Practitioner's Edition

The Endgame Guide to Threat Hunting

Paul Ewing
Devon Kerr

Presented by
ENDGAME.

The Endgame Guide to Threat Hunting

Devon Kerr
Paul Ewing

Foreword by Jamie Butler
Chief Technology Officer, Endgame



CYBEREDGE
PRESS

The Endgame Guide to Threat Hunting

Published by:
CyberEdge Group, LLC
1997 Annapolis Exchange Parkway
Suite 300
Annapolis, MD 21401
(800) 327-8711
www.cyber-edge.com

Copyright © 2018, CyberEdge Group, LLC. All rights reserved. Definitive Guide™ and the CyberEdge Press logo are trademarks of CyberEdge Group, LLC in the United States and other countries. All other trademarks and registered trademarks are the property of their respective owners.

Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of the publisher. Requests to the publisher for permission should be addressed to Permissions Department, CyberEdge Group, 1997 Annapolis Exchange Parkway, Suite 300, Annapolis, MD, 21401 or transmitted via email to info@cyber-edge.com.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on CyberEdge Group research and marketing consulting services, or to create a custom *Definitive Guide* book for your organization, contact our sales department at 800-327-8711 or info@cyber-edge.com.

ISBN: 978-0-9990354-2-9 (paperback); ISBN: 978-0-9990354-3-6 (eBook)

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgements

CyberEdge Group thanks the following individuals for their respective contributions:

Editor: Susan Shuttleworth
Designer: Debbi Stocco
Publishing Coordinator: Jon Friedman

Foreword

The purpose of threat hunting is to reduce the time between a breach and its discovery. Shortening that time can make the difference between spending a few thousand dollars on remediation and millions to deal with a full-on compromise.

Yet many organizations still have not implemented hunt teams. That is evident when we consider that 38 percent of breaches are detected by a third party or external entity rather than the victim organization. To bring down that number, and lower the overall cost of security, we must assume that breaches have occurred and hunt for their evidence before damage occurs.

I talk to organizations every day about threat hunting. Some think that threat hunting is taking known indicators of compromise (IoCs) from threat intelligence feeds and searching for them. But while it is good to know if you are susceptible to previously discovered attacks, that is like driving down the road while looking only in the rearview mirror.

Threat hunting involves hypothesizing about attackers' behavior and verifying the hypotheses in your environment. For example, searching your organization for all remote access tools (RATs) from the past decade is not threat hunting. But you can uncover malicious activity by finding likely persistence locations on systems across your enterprise and measuring how often certain items and certificate signers appear in them.

Since I started in InfoSec, I have seen the pendulum swing between "We trust our security software to stop everything" and "I believe my security software will usually fail." Neither opinion is wise. Threat hunting is the best way companies have to stop the damage and loss that occurs when we have to rely upon third parties to notify us that we have a problem.

Happy Hunting,

Jamie Butler
Chief Technology Officer, Endgame

Table of Contents

Foreword iii

Introduction v

Chapters at a Glance v
Helpful Icons vi

Chapter 1: Be the Hunter 1

What Is Hunting? 1
The Hunt Team 2
What Hunt Teams Look For 4
Categorizing Unknowns 7
Building Environmental Awareness 10

Chapter 2: Structuring Hunts 11

Selecting a Framework 11
Structuring a Hunt Process 13
Transitioning to Incident Response 17
Measuring Your Hunt 18

Chapter 3: Hunting for Fileless Attacks 21

Two Forms of Fileless Attack 21
The Detection Arms Race 23
Anatomy of an In-memory Attack 24
Approaches to Hunting for Fileless Attacks 25
Technique-based Detection 26

Chapter 4: Hunting for Persistence – Basics 29

Why Adversaries Need Persistence 29
The Windows Registry 31
Technique-based Detection 32

Chapter 5: Hunting for Persistence at Scale 35

Taking It to the Enterprise 35
Visualization 39
Example: WMI 40

Chapter 6: Hunting for Lateral Movement 43

Why Adversaries Need Mobility 44
An Example: Hunting for Suspicious Use of PsExec 45
Examine Event Logs 46
Analyze Metadata 47
Analyze Process Events 48
Analyze Command Line Arguments 49

Chapter 7: Credential Theft 51

Survival by Any Means Necessary 51
Example: KERBEROASTING 52
Two Techniques for Hunting Credential Theft 54

Appendix A: Getting Started 56

Appendix B: A Hunt Cheat Sheet 61

Introduction

Threat hunting can seem intimidating at first. How can you come to grips with threats that don't use known malware or indicators of compromise? How can you deduce the presence of "fileless" attacks that leave no files or malicious tools on a hard drive?

Don't worry. If you are new to threat hunting, this guide will show you how to get started. We begin with an overview of threat hunting, then introduce techniques you can use today to stop unknown suspicious activity in your network. You will learn how to find ongoing attacks by proactively searching for signs of fileless attacks, persistence mechanisms, evidence of lateral movement, and credential theft. We include a guide to getting started with free and open source resources (Appendix A), and a cheat sheet to remind you of key information and hunt methods until they become second nature (Appendix B).

Looking to go into depth? We discuss frameworks and tools for finding and analyzing different types of evidence. We also point you toward sources on the web where you can find in-depth discussions and regular updates on important topics.

If this sounds interesting, please read on...

Chapters at a Glance

Chapter 1, "Be the Hunter," reviews basic concepts of threat hunting, the knowledge and experience hunt teams need, and the kinds of behaviors that teams search for.

Chapter 2, "Structuring Hunts," discusses threat modeling frameworks, steps to structure hunts, and metrics for assessing hunt efficiency and efficacy.

Chapter 3, "Hunting for Fileless Attacks," defines two forms of fileless attacks and suggests techniques for discovering them.

Chapter 4, "Hunting for Persistence - Basics," explains methods of persistence and basic techniques for analyzing data to find evidence.

Chapter 5, "Hunting for Persistence at Scale," reviews techniques for working with enterprise quantities of data and explains how visualization can reveal key information quickly.

Chapter 6, "Hunting for Lateral Movement," describes attackers' need for mobility and some of their most common methods, and presents an example hunt for movement using the Sysinternal PSExec remote execution utility.

Chapter 7, "Credential Theft," explores why attackers need to capture credentials and how to detect the KERBEROASTING technique of credential theft.

Appendix A, "The Hunt Environment," outlines technologies that can be used to collect, visualize, and analyze hunt data.

Appendix B, "A Hunt Cheat Sheet," provides a handy summary of key steps in a hunt.

Helpful Icons



Tips provide practical advice that you can apply in your own organization.



When you see this icon, take note as the related content contains key information that you won't want to forget.



Proceed with caution because if you don't it may prove costly to you and your organization.



Content associated with this icon is more technical in nature and is intended for IT practitioners.



Want to learn more? Follow the corresponding URL to discover additional content available on the Web.

Chapter 1

Be the Hunter

In this chapter

- Learn basic concepts about threat hunting
- Review the knowledge and experience hunt teams need
- Understand the kinds of indicators and behaviors that hunters search for

In this chapter we introduce the concept of threat hunting and summarize some of the key ideas. By the end you will be ready to learn how to conduct hunts, the subject of the remaining six chapters of this guide.

What Is Hunting?

Most IT security activities are reactive. They focus on implementing security controls that detect attacker actions, then on responding when indicators of a threat are identified on the enterprise network. This is a central element of security. However, it is also an essentially defensive approach that leaves room for sophisticated adversaries to "dwell" undetected on the network for weeks, months, or even years: ample time to find and steal valuable data, or to disrupt business operations.

In contrast, threat hunting is a proactive approach to securing your organization's systems. It is the process of actively looking for signs of malicious activity within enterprise networks, without prior knowledge of those signs. It allows you to uncover threats on your network without signatures or known indicators of compromise (IOCs).

In this guide we outline effective approaches to finding the

types of malicious activity that don't trigger alerts or use the kind of malware you can catch with signatures. We also tell you where to begin looking for them.

The primary goal

The primary goal of threat hunting is to reduce dwell time. By finding compromises as early as possible, you can remove malicious activity from your systems and networks before attackers can accomplish their objectives. By detecting adversaries early in the intrusion, organizations also reduce the level of effort to scope and remediate, substantially lowering the cost of compromise.

Human processes aided by technology

Threat hunting relies primarily on human processes. Hunters apply the scientific method: defining a problem to be solved, stating a hypothesis to solve it, proposing a procedure to gather and analyze evidence, and measuring the result.

Threat hunting also depends on knowing what to look for and where to look. Members of the hunt team must use their knowledge and experience, as well as this guide and other resources describing adversary tradecraft, to make those decisions.

Technology does not drive hunting processes, but it is a critical enabler. Hunt-specific technology solutions allow hunters to capture, identify, correlate, enrich, measure, and analyze thousands of pieces of data needed to conduct effective and efficient hunts.

The Hunt Team

Responsibilities for threat hunting can be organized many ways.

Smaller organizations typically make threat hunting a part-time job. Hunters may take turns hunting on a designated set of assets, and sometimes are responsible for remediation actions after compromises are discovered.

Larger organizations often have full-time hunters, sometimes

dozens of people distributed around the world, explicitly working as a dedicated hunt team. The hunt team is usually comprised of a team lead and several hunters. The hunt team focuses on quickly identifying the presence of advanced adversaries and compromises and prioritizing their handling. It then hands off responsibility for incident response and remediation.

Knowledge

The hunt team needs people with a wide range of experience. It is essential to include people with knowledge of:

- ☑ Security - methods commonly used by adversaries, incident analysis and response techniques, and tools for detecting and analyzing attacks, malware, and other exploits.
- ☑ Threat detection - techniques and tools, including those typically used for red teaming and penetration testing, and those used by adversaries.
- ☑ IT operations - applications, networks, and enterprise architectures, and the inner workings of the operating systems used by the organization's assets.
- ☑ Communication – how to communicate effectively with other hunters and security professionals, system administrators, upper management, the legal team, and human resources staff.

Roles

Hunt teams should include people who can fulfil a variety of roles, including:

- ☑ Incident manager – a technical person who understands how each threat exposes the business to risk, and has the authority to direct the team to investigate and address signs of malicious activity.
- ☑ Response analyst – a person who can analyze data from one or more sources, then document and communicate findings.

- ☑ Malware analyst – a person who can use automated malware analysis tools to understand and reverse-engineer malware samples, and who can produce decoders and other utilities for response analysts.
- ☑ Operations staff – people with knowledge of system and network administration, application development, and other IT functions.



Membership on a hunt team can be a great way to learn new skills and explore new career directions.

What Hunt Teams Look For

You can't find what you don't look for (pardon our shaky grammar). *What hunt teams look for is very different from the known malware samples and previously identified IOCs that occupy the time and attention of the security operations center (SOC).*

Hunt teams focus on uncovering behaviors and other evidence of attacker techniques and activities.



We don't mean to say that your hunt team should ignore "known bad" indicators such as file hashes, IP addresses, and DNS records. Instead, the team should create automated searches and queries to find them. Automated matching frees up your most precious asset: human beings. Treat the results of this matching process as alerts, triage them, and use the findings as input for your hunts.



As a general practice, hunt team members should research the offensive tools, techniques, and procedures (TTPs) of attackers in a threat-agnostic manner, documenting those that target the enterprise's industry and geographical locations. Note that although techniques and tactics are sometimes treated as synonyms, you should think of techniques as aspects of a threat actor's skill set, and tactics as *applications* of those techniques.



Most organizations are targeted by more than one kind of adversary. Financially motivated actors, "hacktivists," and state-sponsored operators each have different motivations, targets, and TTPs. Study all of those who might target your enterprise. Keep in mind that adversaries are not constrained in the ways defenders are, and some individuals operate in more than one threat category.

Artifacts and indicators of behaviors

As attackers interact with the environment, their behaviors produce artifacts that we can use to identify the intrusion. These artifacts come in several varieties, including but not limited to:

- ☑ Filesystem metadata
- ☑ Network metadata (such as NetFlow and DNS queries and responses)
- ☑ Application data and metadata
- ☑ User and authentication records

For example, if a threat actor is using an Internet server application programming interface (ISAPI) filter of Microsoft's Internet Information Services (IIS) to persist malware on a webserver, it might generate:

- ☑ One or more malicious files
- ☑ Modifications to the IIS configuration file
- ☑ User session artifacts in the registry and filesystem
- ☑ Authentication from remote access attempts

Even if the threat actor took very few actions, those actions would leave behind a trail you can follow if you collect the right data and examine it regularly. This is an example of Locard's exchange principle of forensic science.

Locard's Exchange Principle

"Every contact leaves a trace"

Dr. Edmond Locard (1877 – 1966).

Restated by forensic scientist Paul L. Kirk as:

"Wherever [the criminal] steps, whatever he touches, whatever he leaves, even unconsciously, will serve as a silent witness against him. Not only his fingerprints or his footprints, but his hair, the fibers from his clothes, the glass he breaks, the tool mark he leaves, the paint he scratches…All of these and more, bear mute witness against him. This is evidence that does not forget."

Traces of attacker tradecraft

According to Merriam-Webster.com, tradecraft is "the techniques and procedures of espionage." Most attackers, just like spies, have to follow one of a few general procedures in order to achieve their goals. If you know these general procedures, the attacker tradecraft, you can look for traces of those activities.

Defenders should be aware that most of their adversaries attempt to take the following steps:

- ☑ Access a victim environment
- ☑ Create and maintain command and control (C2) communication between the attackers on the outside and software under their control on an internal system
- ☑ Obtain and leverage additional privileges
- ☑ Conduct reconnaissance of networks and hosts
- ☑ Identify data, applications, and systems of interest
- ☑ Access one or more systems
- ☑ Collect or destroy sensitive data
- ☑ Exfiltrate the data or otherwise achieve their objectives

If we know that attackers are going to follow these steps, we can hunt in parts of the computing environment that would show traces of these actions.



Let's look at how you might hunt for evidence of attacker tradecraft. Suppose you suspect that a threat actor is using an ISAPI filter to persist malware on a web server. Because you have done research on this persistence technique, you know that it loads a malicious DLL when an adversary-defined file extension (like ".xcfGGj3kBks") is requested from the web server. You could monitor IIS configuration files to detect modifications that would indicate the use of this technique (of course, after you imposed software revision controls to ensure you could roll back to a known good state). You could also check the IIS logs, which might show the requests that triggered the malicious ISAPI filter execution or the processes spawned by the web server. In fact, you can baseline these log entries to streamline analysis.

Intrusion attributes

There are other characteristics of intrusion attempts that can tip you off to malicious activity. Advanced attacks often:

- ☑ Combine multiple discrete techniques
- ☑ Occur over relatively compressed time periods
- ☑ Include both benign and malicious executables
- ☑ Acquire and leverage some type of privileged access
- ☑ Involve one or more endpoints
- ☑ Change the contents of endpoint file systems

Categorizing Unknowns

One of the biggest challenges of threat hunting is that you are examining events and artifacts that are not inherently malicious or benign but *might* be either. However, there are several techniques you can use to point toward one side or the other.

Prevalence

You can make a good judgement about the likelihood that events and artifacts are related to attacks by measuring their prevalence, the frequency with which they occur in an environment.

Sophisticated attacks usually only affect a relatively small number of systems on the network. Therefore, events and artifacts that are scarce are more likely to be malicious than those that are more prevalent.

For example, suppose you find a previously unknown binary file on 85 percent of your Windows systems. That is almost certainly benign software. It is unlikely that an attacker could plant malware on such a high percentage of your systems.

On the other hand, an unknown binary file on 2 percent of your Windows systems could be the result of a successful phishing campaign. As a guideline, low prevalence is more suspicious than high prevalence.

There are rare cases of widespread outbreaks that leave evidence on hundreds of systems. However, these exceptions to the rule are usually obvious, such as ransomware.

Recentness

Recentness is another analytic that enables analysts to make judgement calls, by grouping events and artifacts into the newest and the oldest. Since the attacks you need to worry about most are the ones still going on, recent occurrences are more likely to be malicious than older ones.

For example, an unknown binary file that appeared on endpoints three days ago is more likely to represent an ongoing threat than one that was last installed three years ago. Likewise, a registry setting that hasn't been changed in two years is less suspicious than one changed two days ago.

Patterns of behavior

There are many actions and behaviors that appear innocuous when considered individually, but can show patterns across an enterprise that indicate malicious intent.

For example, endpoints on the network establish connections with servers on the Internet all the time. However, when an endpoint establishes connections with an unknown server at regularly scheduled intervals, it is a tip-off that malware might be "phoning home" to a server controlled by attackers.

In the same way, logons succeed and fail every second of every day, but logons from invalid or disabled user accounts are suspicious.

Anomalies

Departures from standard behaviors can also help you categorize unknown events and artifacts.

The types of deviations that merit investigation include:

- ☑ Unusual volumes and frequencies (uploading 10GB to an external website in under 24 hours; 20 failed logon attempts per minute from one endpoint)
- ☑ Deviations from a standard configuration (executable files that do not appear on the golden image; differences in registry keys between newly created systems and systems that have been present in the environment for months)
- ☑ Departures from convention (host and file names that do not follow the organization's naming conventions)

Other places to hunt for anomalies include:

- ☑ Run and RunOnce keys
- ☑ Windows services
- ☑ Image file execution options
- ☑ Application debuggers
- ☑ Registered COM servers

Building Environmental Awareness

Before your team can start threat hunting for unusual patterns of behavior and anomalies, it needs to build awareness of its environment, determine expected activities, and create baselines. You need to find the answers to questions like these:

- ☑ How do our system administrators interact with servers, and from where?
- ☑ Which accounts have membership in privileged groups, and which ones?
- ☑ What remote execution tools are commonly used in this environment?
- ☑ Which UserAgent strings are common here?
- ☑ Where are the most sensitive pieces of data stored, and how do users normally access that data?
- ☑ What are typical levels of server-to-workstation, server-to-browser, and workstation-to-server lateral movement?

Have you seen The Hunter's Handbook?

The rest of this guide will give you a lot more detail about where and how to hunt. But if you want a manager's perspective on threat hunting and how to prepare your organization for it, then don't miss *The Hunter's Handbook: Endgame's Guide to Adversary Hunting*. That guide discusses:

The power of hunting: the basic concepts of threat hunting, the motivations for hunting, and the benefits of hunting.

The hunt process: the major components of the hunt, including the technical details of what's involved in executing each component.

The challenges of hunting: common roadblocks to successful threat hunting and how these challenges can best be addressed.

Hunt readiness: how to get your organization ready to adopt and use hunting practices.

The hunt experience: a hunt case study based on a fictional enterprise and situation.

Hunt technology: practical considerations to keep in mind when choosing an automated threat hunting solution.

Chapter 2

Structuring Hunts

In this chapter

- See suggestions for selecting an attack lifecycle framework
- Learn how to structure a hunt process in six steps
- Understand different types of metrics for assessing your hunts

Although you can take an unstructured approach here and there, for most of your threat hunting activities you should establish some structure. Defining a process for each hunt helps make the steps repeatable. This makes it easier to measure outcomes and assess the performance of your tools and your team's methods. It also enables you to present findings to management in a consistent manner across hunts.

The best approach to defining a hunt process is to apply the scientific method:

- ☑ State the problem to be solved and a hypothesis for solving it
- ☑ Propose a procedure to gather and analyze evidence
- ☑ Define metrics to evaluate the success of the effort

Selecting a Framework

Before defining individual hunts, it is useful to select an attack lifecycle framework that breaks down the phases of a typical cyberattack and the techniques it might use in each phase.

There are many frameworks available within the InfoSec community, including Lockheed Martin's Cyber Kill Chain® and

Mandiant's Attacker Lifecycle Model. Our favorite is MITRE's *Adversarial Tactics, Techniques and Common Knowledge (ATT&CK™) framework*. It goes into more depth than most of the others, and its details on post-compromise attacker techniques provide useful guidance on where, what, and how to hunt for evidence of them. ATT&CK isn't limited to one type of adversary, but instead provides threat-agnostic techniques that are relevant to every threat.

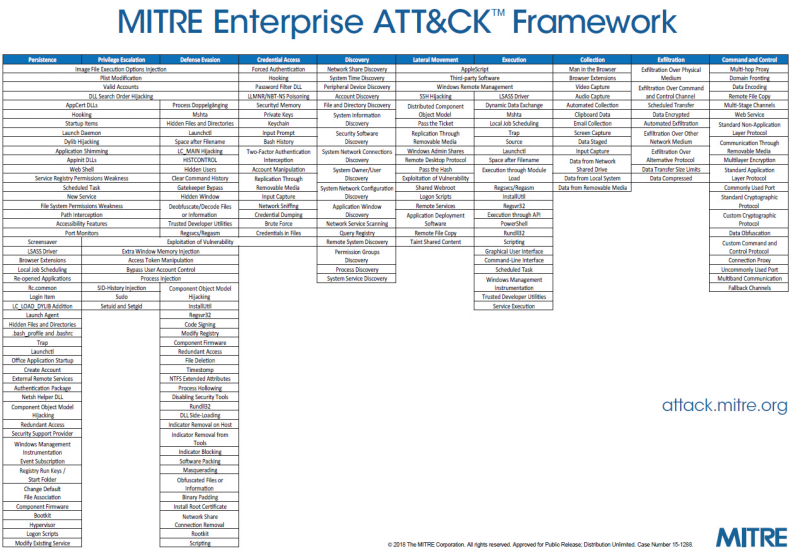

Figure 2-1: Attack lifecycle frameworks like MITRE's ATT&CK™ can help your team think like attackers and focus on likely attack techniques.

ON THE WEB

You can find information on the web about the *MITRE ATT&CK framework*, as well as the *Lockheed Martin Cyber Kill Chain framework* and the *Mandiant Attack Lifecycle Model*.

Custom frameworks

Although we strongly recommend starting with one of the standard frameworks, you can build on it or combine elements of several frameworks to create a custom framework of your own. No one knows your environment like you do.

For instance, if your organization spans a wide geographic area, you could add time zone-based hunts to your rubric. If certain servers contain key intellectual property, you may want to view their logs to hunt for logins during unusual time periods, or from specific network ranges. By adding awareness of your specific environment into your threat framework, you can enrich other searches and reduce the time necessary to identify malicious activity.

It is worth taking time to select or build an attack lifecycle framework to guide your hunt, because it will help your team think like attackers and focus on the attack techniques that are most likely to be used against your organization.



Don't fall into the trap of looking only for threats associated with your industry. While this may feel like a logical way to prioritize, you don't know what you don't know. Techniques developed by state-sponsored actors focused on one type of victim can inspire threats that target broad audiences, and vice versa. Frameworks like MITRE's ATT&CK can help you apply a threat-agnostic approach.

Structuring a Hunt Process

Every hunt is different, but as we mentioned earlier, structuring each one based on the scientific method helps you make the steps repeatable and the results measurable. This section focuses on applying the scientific method to hunt methodology development.



Robert M. Lee and David Bianco have written a great white paper on hypothesis-driven threat hunting: *Generating Hypotheses for Successful Threat Hunting*.

Step 1: Propose a hypothesis

State a reasonable assumption about one or more adversaries and the techniques they might use to enter or persist in your environment.

An example might be: "Bad actors will leverage benign and signed Windows binaries to perform malicious activities, which will not be flagged as suspicious by my existing security tools."

Step 2: Identify evidence to prove the hypothesis

Identify the forms of evidence that could prove or disprove the hypothesis and determine how that evidence can be collected.

To continue the example started above, you might enable process execution auditing (Windows EID 4688 or Sysmon EID 1) and begin aggregating:

- ☑ Process names, paths, hashes, and signing information
- ☑ Command line arguments
- ☑ Any network destinations captured
- ☑ DLLs loaded by the executable

Sysmon is capable of returning the same information as detailed process auditing in Windows. It has several advantages as well: Sysmon provides more information about parent processes, can provide multiple types of file hashes, and describes network connections, among many other valuable capabilities.



Don't stop with one form of evidence; usually two or more are needed to prove a hypothesis with a high degree of confidence. A single source of evidence or visibility can turn out, at the most inconvenient time possible, to be misleading.



It is important to document the sources of evidence you have chosen, and to ensure that data collected from each source is consistent. Otherwise you may end up with pieces of data that can't be compared or combined. For example, if you plan to match encoded data against keywords, you should make sure a decoding operation takes place before that data arrives at your console.

Step 3: Develop analytics

Describe how the evidence can be reduced, grouped, and analyzed to reach a conclusion. Typically, this involves identifying anomalous activities or sequences of events associated with attacks.



Techniques such as prevalence and enrichment (pulling in external metadata to help determine if a given artifact can be characterized as benign or malicious) can dramatically reduce the human effort required for analysis. As you develop analytics, consider the tactical and strategic advantages of these techniques in eliminating false positives and reducing the amount of data you need to analyze.

Step 4: Automate

Usually you can automate and schedule the process of collecting events, performing data reduction, and matching keywords against data. Automation replaces manual processes and frees your hunters to apply their judgement to the results. Ideally, you can allocate tasks that don't require knowledge or experience to machines, while reserving those that do require judgement to human analysts.



Learn more about automation by reading Endgame's "*Think Offense: Automate the Hunt*" white paper.

Step 5: Document

In the heat of an ongoing hunt it is very tempting to process yet more data and put off documentation until later – or never. However, this is a tactical miscalculation. Usually it is impossible to recall after the fact the details of the hunt: the evidence collected, the types of analysis performed, and the justification for the conclusions. This information will be lost unless it is documented during the course of the hunt, at least at a basic level.



When possible, have experienced analysts record their methods for identifying suspicious events. This and other documentation can be extremely valuable to less-experienced members of the hunt team by bringing them up to speed on

the tactics of attackers and successful hunt techniques. Consider creating a "training dataset" based on some commonly-performed hunts that you can use to evaluate and train new analysts.



While your analysts are documenting the details of the hunt, ask them to highlight opportunities to reduce the volume of data you collect. Typically this involves filtering out known or otherwise trusted events. This practice pays substantial dividends, because cutting the amount of data retained ultimately improves your ability to answer questions quickly.

Step 6: Communicate and report

You should decide at the beginning of a hunt what to communicate with management and when. This decision includes when to declare that an incident has been detected, and how often to provide updates on the progress of the hunt.

At the conclusion of the hunt you should create a record that includes:

- ☑ The metrics of the hunt (discussed below)
- ☑ The root cause of any compromise detected
- ☑ The scope of affected machines, accounts, and applications
- ☑ A description of the techniques detected
- ☑ IOCs to be used for detecting similar attacks
- ☑ Lessons learned and areas for increasing visibility and improving future hunts
- ☑ Recommendations for changes to the organization's security controls



As your hunt team matures, track when your recommendations are made and how long before they are implemented. These measurements will give you important clues about the influence of the team.



Your organization should have a formal process for sharing the results of hunts with leadership. It is critical to have an executive sponsor with sufficient authority and responsibility to promote the goals of the hunt team. Generally speaking, the greater your executive support, the more influence your team will exercise. Hunt teams should share the results of hunts, improvements to visibility, findings, and resource costs with this sponsor *at least quarterly*, for wider socialization. Hunt teams that skip this step will have trouble obtaining resources and making improvements.

Transitioning to Incident Response

Hunt teams do not operate in isolation. At some point they must decide whether to declare an incident and escalate the evidence of malicious behavior to the incident response team for further investigation and remediation.

Every organization has its own criteria for making this decision. But if you find yourself in a gray area and need a tie-breaker, step back and consider very basic characteristics of your potential incident:

- ☑ Is the evidence rare or widespread?
- ☑ Is the evidence recent or has it been present in the environment for a long time?
- ☑ Do the activities map to adversary tradecraft?



Having made your call and declared an incident, spend some time thinking through the incident's priority. Consider the likelihood that this attack would target your business, industry, or region. Estimate the impact it would have on your enterprise if it were successful. You want the incident response team to address the highest risks quickly, but not be overwhelmed by incidents that are unlikely to have much impact.

Measuring Your Hunt

Unless you can measure the results of your hunts, you won't know which ones are successful, if your team is improving over time, or if you are producing important results for your enterprise. Measuring the outcome of each hunt exercise should be discussed during the development phase and refined each time the hunt is performed.

Different kinds of metrics can serve different purposes. You can measure how long it took to obtain the necessary data, how many human hours were required to complete the analysis, the average number of findings, and the total time to complete the process.

Every hunt gives the team an opportunity to improve on each metric that you have adopted.

Was the hypothesis confirmed?

For some hunts, the evaluation is binary: our hypothesis is confirmed, or it isn't. For example, let's say the hypothesis for a hunt is: "We can identify malicious activity by finding unsigned binaries launched from persistent registry run keys." The result will be "yes" if you find unsigned executables that were loaded from persistent registry run keys and you associate them with incidents. It will be "no" if you are unable to find any unsigned run keys, or if you find some but determine that they are benign.

A hypothesis needs to be sufficiently detailed so that analysts running the hunt can prove or disprove it. Sometimes a hypothesis is too generic to be proven. One common example is: "We can hunt for and find persistent malware." That is too vague to produce useful results. A more meaningful hypothesis would be something like: "We can find common registry-based malware persistence that leverages Run, RunOnce, ActiveSetup Installed Components, AppInit_DLLs, and Services registry keys."



If you're just getting started, consider a hunt for each cell in the ATT&CK matrix. You can combine related cells, such as those related to persistent techniques, for group analysis.

How effective was the hunt at finding issues?

One measure of the effectiveness of a hunt is the number and severity of incidents and issues it uncovered. You can measure and report the number of:

- ☑ Vulnerabilities discovered, and recommendations provided
- ☑ Incidents discovered, grouped by severity
- ☑ Compromised systems discovered, grouped by severity of the compromise
- ☑ Days of dwell time for each incident discovered
- ☑ New attacker tactics uncovered



Some hunts may not produce findings of any kind–but that does not mean they were failures. Quite the contrary; sometimes the only way to detect a technique is by collecting routine data to use as a baseline of normal behavior. When the results of the hunt appear to be negative, it is all the more important to document the process and outcomes.

How effective was the team at improving security?

You can also try to measure the effectiveness of the team and its impact on the organization, and how those factors change over time. Possible metrics include:

- ☑ New sources of evidence integrated into hunt processes
- ☑ New methods of data reduction and improvements in searching technology
- ☑ Reductions in false positives of incidents
- ☑ Hunts escalated to investigations
- ☑ Investigations resulting in ongoing attacks being blocked and remediated

> ☑ Vulnerabilities and bad behaviors corrected as a result of hunt reports

Endgame and the MITRE ATT&CK™ Matrix

To be effective against today's sophisticated attackers, security programs must operate with a comprehensive model that covers the full scope of techniques used by adversaries.

At Endgame, we leverage an open source framework, the MITRE ATT&CK™ matrix, to help enterprise teams transform their security programs.

1. Comprehensive scope of protection

Endgame covers the breadth and depth of the MITRE ATT&CK™ matrix, with prevention, detection and response, and automated hunting. At Endgame, we recognize that focusing on a few tools or attack vectors is not enough, but rather think about the techniques and tactics of the attacker. The MITRE ATT&CK™ matrix provides a comprehensive landscape that Endgame leverages in its preventions and detections.

In addition to building capabilities across the ATT&CK™ matrix, the Endgame team has collaborated with MITRE to detect new tactics such as COM Object Hijacking. This tactic is often used by attackers to execute code by manipulating Microsoft's Component Object Model (COM), specifically its software classes in the current user registry hive, and enabling their persistence on an endpoint.

2. Validating the efficacy of our platform

Endgame is the first endpoint security vendor to collaborate with MITRE to validate the efficacy of its platform beyond malware-based attacks. To measure Endgame's performance against sophisticated attacks, MITRE simulated the tactics used by APT3, a prolific Chinese APT group responsible for intellectual property theft costing companies over £9.2 billion ($12.5 billion) a year. This attack used more than a dozen techniques to gain and maintain access, including PowerShell misuse, credential dumping, scripting, and persistence.

You can learn more about Endgame's commitment to the MITRE ATT&CK™ matrix by reading the white paper: Redefining Endpoint Protection: *What's Your Attack Model?*

Chapter 3

Hunting for Fileless Attacks

In this chapter

- Understand the two forms of fileless attacks and how they work
- Learn how to detect in-memory attacks
- Discover techniques for determining if administrative tools are being used by real administrators or by attackers

Recently hackers have dramatically increased their use of fileless attacks, also known as non-malware or zero-footprint attacks. Fileless attacks can seem particularly intimidating to beginning hunters because there are no recognizable malware files or malicious tools that can be located on a hard drive. There are techniques for detecting them, though.

Two Forms of Fileless Attack

The term *fileless attack* is a blanket term to describe two different adversary techniques: using tools and applications already present on a host ("living off the land"), and malware that is memory resident without a filesystem component. Let's examine the difference and demystify the terms.

Living off the land

As its name implies, living off the land describes techniques used by attackers to conduct their operations with tools already on a host. Often these are administrative tools or operating system features which, unfortunately, are often more powerful than any custom malware the attackers might build themselves.

One very popular example is PowerShell. PowerShell is both a language and an administrative framework built into Windows. It exposes hundreds of commands, called cmdlets, to attackers. Like WMI, it extends all the built-in functions of Windows system programs to allow attackers to enumerate, move laterally, persist, and execute. These activities and others can be performed via a simple script or using the PowerShell console.

Some PowerShell methods allow an adversary to grab the content of a script from an Internet location and execute it in memory – without saving the content of the script to the victim system. In this sense the technique is both living off the land with the native PowerShell framework and executing malware (in the form of a script) in memory.



This behavior isn't that unusual. There are plenty of built-in tools that can interpret a script in this way. For example, the Windows Script Host (WSH) executables wscript.exe and cscript.exe interpret Jscript or VBScript; HTA scripts (JavaScript) can be loaded by the mshta.exe binary.



To learn more about how attackers misuse PowerShell, just run a quick Internet search for "PowerShell + hacking." You will see numerous blogs, videos, presentations, and news headlines. The MITRE page for PowerShell Misuse is: https://attack.mitre.org/wiki/Technique/T1086.

PowerShell isn't the only tool at risk of being exploited by hackers. Any local application or piece of software that allows arbitrary code execution is in danger of being found and leveraged by hackers. Benign tools like Sysinternals BgInfo, which are meant to print system information to the desktop background as wallpaper, can be abused by threat actors to execute malicious VBScript. When hunting these types of techniques, remember that the native tools are likely to show up as trusted, so to find evil you need to look at the things they're interpreting.

In-memory malware

Another type of fileless attack represents a more literal definition of "fileless": memory-resident malware. Attackers inject a malicious payload into applications that are already running.

This technique can be used to evade controls like some application whitelisting and antivirus solutions, because the attacker's code executes in applications that have been approved by the organization.

Part of the reason why the use of in-memory attacks is growing so rapidly is that they are no longer the province of sophisticated attackers alone. Off-the-shelf offensive frameworks freely available on the Internet enable entire categories of such attacks. These frameworks have dramatically reduced the barrier to entry for threat actors of all experience levels.

The Detection Arms Race

Why do attackers employ fileless techniques? That's simple: because they evade detection by many existing cybersecurity tools, which attempt to identify attacks by finding malware. Many antimalware products, enterprise and open-source alike, struggle to stay at the forefront of this arms race because adversaries are able to find circuitous ways to evade them. A product that can inspect a downloaded PowerShell script, for example, may not be able to inspect one that is heavily obfuscated, or is stored in a registry key and piped as input to a running process.

Fortunately, even fileless attacks create artifacts that provide evidence of techniques used by attackers.

Be a Blue Team Champion

Does your company have red team-blue team exercises? Are you a blue teamer?

If so, learn how to detect living off the land techniques. Red teamers are notorious for using these methods. Many of them are incredible administrators because they know Windows systems so well. By mastering hunt techniques to detect these methods, you can become a famous blue team champion!

Anatomy of an In-memory Attack

Initial infection

The initial stages of an in-memory fileless attack are not too different from those of a conventional malicious campaign. Fileless attacks often utilize spear phishing and drive-by downloads to compromise their victims and gain a foothold on the network.

Stagers and cradles and droppers! Oh my!

When talking about malware, practitioners often refer to malicious payloads using the terms "stagers," "cradles," and "droppers." While these aren't interchangeable, we can discuss them generically as a class of malicious files used to introduce additional files or scripts.

Some of these can be described as outright malware, where the malicious payload is embedded within another file type, such as a DLL, and extracted on a victim system before being executed. Other droppers might be a script that simply contains a request to download additional malware or scripts. Still others download a stream of shellcode and execute it in memory, either whole or in parts.

In-memory execution

Ultimately, a malicious payload will be executed. An in-memory attack may take the form of process injection, process hollowing, or side-loading.

In the case of process injection, the malware creates or allocates some space in process memory, then creates a remote thread to a section of memory within a legitimate process. This process is illustrated in Figure 3-1.

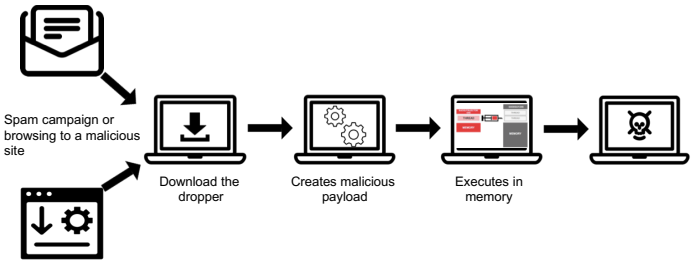

Figure 3-1: Fileless attacks often use a dropper to deliver a malicious payload that executes in memory.

ON THE WEB

For an in-depth look at process injection techniques, see the Endgame blog post: *Ten Process Injection Techniques: A Technical Survey of Common and Trending Process Injection Techniques*.

Approaches to Hunting for Fileless Attacks

Isolated memory forensics

One approach to finding in-memory attacks is to examine a capture of system memory. There are plenty of tools available to assist in the forensic analysis of memory. For example, you can use tools like Volatility, which comes with a suite of built-in plugins, to find injected code residing within legitimate applications.

Investigators have successfully used tools like Volatility, margaritashotgun, and PowerForensics to acquire process memory and apply analytics at scale. For hunting purposes, you should parse metadata from memory. Usually that only requires collecting a few hundred megabytes of metadata per system.

Aggressive approaches like collecting full memory captures are impractical when you are working at enterprise scale, because they might require collecting as much as 16GB from each workstation and 128GB from every server.

Drowning in noise

What about living off the land techniques?

You can start by looking for "typical" usage of administrative tools. You will probably generate mountains of data. You will find lots of anomalies to investigate (oh boy!). Then you will spend days discovering all the bizarre and unexpected ways that users and admins operate while doing their legitimate jobs.



An anomaly is not automatically suspicious. Some anomalies are just noise. If you find too many false positives, either reclassify what you consider anomalies or find another behavior to monitor. Hunt teams should establish relationships with IT and network operations groups. Working together saves time and helps analysts put anomalies in context.

Technique-based Detection

Threat hunting in memory

There are some open sources tools to help you examine memory to find evidence of malicious behavior. One is the PowerShell library Get-InjectedThreads, developed by Joe Desimone of Endgame and Jared Atkinson of SpecterOps. In a relatively low-noise approach, this tool scans active threads on the system for suspicious start addresses that may indicate process injection has occurred.



For example, an attacker might call VirtualAllocEx to allocate space for malicious code to execute, and then utilize CreateRemoteThread or another API call to execute the malicious code within another application. Get-InjectedThreads will retrieve the start address of each active thread, then determine the associated section properties. If there is an observed executable running within this section, it is deemed to be injected. But keep in mind that some legitimate applications perform process injection (and you might also run across an injected thread and alert).



For a presentation describing threat hunting in memory in detail and explaining how to use Get-InjectedThreads, watch this video from the SANS Threat Hunting Summit 2017 conference: *Taking Hunting to the Next Level*.

Timing is everything

A theme throughout this guide is how you can use time to assist you in your hunt, or more precisely, how determining a sequence of events can point to malicious activities.

In a living off the land scenario, the attacker wants to launch a native admin tool to execute malicious commands. But while the tool may be nothing out of the ordinary, like PowerShell, the way it is launched may indicate malicious intent. If you examine the parent process lineage of PowerShell or other admin tools, you might find some interesting artifacts.

For example, you might observe a local admin tool being executed as child of your email application a few minutes after an email is received. This is a good indicator of something suspicious, perhaps part of a spear phishing attack.

Another example would be seeing many enumeration commands (such as ipconfig, net *, whoami, systeminfo, or netstat) being run in a very short time. This behavior would be consistent with an attacker's attempt to discover more about a network quickly by running these commands manually or from a script. System administrators use the same commands, but not all of them, and not within a few seconds.

Searching for intent

As we noted earlier, attackers may try to look like administrators in your network, but they do different things. You can look for running admin applications that are exhibiting non-admin behaviors.

See if you can find unusual command line parameters. For example, you might turn up PowerShell being executed with encoded parameters, hidden windows, or other unusual execution parameters such as "-e *," "-en *," and "-ep *."

You can also search for native Windows applications that allow code execution, for instance installUtil.exe, regsvr32, regasm.

exe, or even rundll32.exe. Look for these signed and trusted applications being executed, and closely examine the command line parameters and contents of memory sections allocated.

Finally, you can hunt for suspicious network activity associated with local tools. If you see a lot of outbound connections from tools designed for internal system maintenance, it is probably not the work of legitimate systems administrators.

Endgame Threat Hunting in Memory Techniques

The increasing popularity of memory-resident malware reflects its success evading detection by security products and practitioners, as well as the proliferation of code and knowledge related to in-memory techniques.

Endgame employs layered protection to prevent fileless attacks. We do not simply rely on naïve approaches like monitoring well-known system call sequences for process injection, but efficiently analyze memory to find all known evasion capabilities. Endgame's endpoint protection platform has built-in detections for techniques including shellcode injection, reflective DLL injection, memory module, process and module hollowing, Gargoyle (ROP/APC), and many more, offering the best available capabilities for locating in-memory threats. Combining pre-attack and ongoing attack prevention at the kernel and user levels of the operating system, Endgame ensures complete protection against fileless attacks, regardless of when in the attack lifecycle the agent is deployed to endpoints.

Pre-attack prevention: Endgame's patent-pending technology prevents fileless attack techniques like shellcode injection and DLL injection. Kernel-level analysis, performed on every executing thread, stops fileless attacks before an adversary can gain a foothold in memory. Once a fileless attack is blocked, the analyst gets an alert providing complete visibility of the origin and the full extent of the attack.

Ongoing attack prevention: To find adversaries resident in memory, Endgame automates in-memory analysis and identifies techniques such as memory modification, memory injection, hidden modules, and packed and encrypted areas in memory. It provides coverage across unlimited endpoints in minutes, with no end-user impact. Unlike other endpoint detection and response (EDR) tools, Endgame allows analysts to proactively root out advanced attackers before any data theft and loss. With a few clicks of a button, analysts can stop fileless attacks at scale across the enterprise.

Chapter 4

Hunting for Persistence – Basics

In this chapter

- Learn why attackers need persistence, and why it can be their Achilles' heel.
- Review basic techniques for hunting for evidence of persistence.

Why Adversaries Need Persistence

Some intruders can reach their targets during the first rush of their attack. They gain temporary access, just enough to complete their mission, and then withdraw, cleaning up as they go.

Much more often, however, adversaries have to play a long game: establishing a beachhead on one host, maintaining contact with an external server they control, obtaining new credentials, moving laterally to other systems, locating targets, obtaining data, and then exfiltrating the data. To do this, they must be able to maintain a presence in the victim environment that survives reboots and access interruptions.

For this reason, persistence is usually one of an attacker's first objectives. After all that work to access the victim's system, why risk losing control because of a power outage or a software update that forces a restart?

In some complex intrusions, establishing persistence is a cyclic process that ebbs and flows based on factors like shifting adversary goals, changes to the environment, and interaction with the security team. As attackers compromise additional

systems, they may install more tools and malware to afford themselves more persistent footholds. Some organized threat actors even install multiple types of malware, with different forms of persistence and capabilities, so that even if one or two are discovered they retain a presence on other hosts.



It can be perilous to try to clean up infected systems. If a threat group has deployed multiple forms of malware with different persistence mechanisms, you may miss some. It is safer to rebuild infected systems from a trusted image. If you do decide to clean up, make certain the affected hosts are restricted to minimal access from the local network and can communicate only with trusted destinations.

Their need is your opportunity

But persistence is the Achilles' heel of many attackers. You can find persistence mechanisms and use them to begin unravelling the chain of techniques used in the attack. While a multitude of persistence techniques exist, the majority of threats leverage just a handful, so in your hunts you can start out by prioritizing the most common ones.



If you want to understand the tremendous number of persistence options, look no further than the Beyond good ol' Run key series on the Hexacorn.com blog. Started in 2012, it covers dozens of the most common, uncommon, and rare methods of achieving persistence. You should take the time to understand how each persistence method functions and how it can be detected. Another technical blog with good material on persistence is enigma0x3.



You should also review the MITRE ATT&CK™ matrix section on persistence at: https://attack.mitre.org/wiki/Persistence. Studying the tactics outlined there regularly will improve your ability to identify and analyze persistence techniques at scale.

Even fileless attacks use persistence

Some security professionals are intimidated by fileless attacks. But as we discussed in Chapter 3, you shouldn't be. They are simply a class of techniques that don't depend on malware or any other type of file stored on the host's file system.

Techniques for fileless attacks include:

- ☑ Storing shellcode within a registry key value, executed by a generally benign Windows application
- ☑ Storing a script within a data structure like the WMI CIM or another database, executed by a script processor such as the Windows Script Host (WSH)
- ☑ Using a PowerShell cmdlet to download malicious scripts from an Internet location and passing them to one of several utilities
- ☑ Using stored procedures to perform inline compilation of C# or other code

But don't confuse "fileless" with "undetectable." Although there are no tell-tale files pointing to an attack, there are many types of evidence you can use to hunt for malicious execution. Many of the best involve finding evidence of persistence mechanisms. You can examine the registry, audit running processes, and audit process lineages of ancestors and their descendants.

For example, if you decide to hunt for shellcode stored in a registry key, it is a fairly trivial task to search all registry keys for the shellcode representing a compiled binary. By performing a content search of registry keys for the executable file header, you can identify suspicious keys in short order.



A compiled binary with an intact PE header will begin with the hexadecimal string "0x5a4d" (for "MZ" – the initials of Mark Zbikowski, one of the developers of the MS-DOS operating system).

The Windows Registry

Most organizations begin hunting for persistence in the Windows registry.

For those who haven't been exposed to it much, the registry is an improvement on host and application configuration management. Before the introduction of the registry to maintain centralized configurations on a per-host basis, these settings used to be stored in configuration files!

The popularity of the registry in threat hunting is due in part to the array of free and low-cost methods for querying it.

Those include PowerForensics and Sysinternals Autoruns, powerful free tools that increase your visibility into a multitude of persistence techniques.

You can start your search by investigating registry locations that commonly contain evidence of persistence, such as Run and RunOnce keys, and Windows Services keys. Although these are only a handful of the keys in the registry, they are used by a substantial percentage of malicious malware families and samples.



If you subscribe to a threat feed that includes IOCs, you can search for the registry keys mentioned in that feed. Set up an automated process to collect the data, match it against IOCs, and present the results to analysts as alerts. Manual IOC matching is inefficient and unnecessary; those hours are better spent advancing coverage of ATT&CK, implementing controls, and performing meaningful analyses.

Technique-based Detection

While it may be tempting to just match IOCs against your available evidence and call it a success, that's not really threat hunting. This section focuses on techniques hunters can begin with to detect persistence techniques. For most organizations, this process begins by prioritizing sources of evidence, conducting analyses, and using environmental awareness to help reduce the volume of data.



You can greatly reduce the volume of data you collect and retain by profiling the base image of each system type in your organization for default persistence locations. Knowing the profile of workstations and servers when they are deployed for the first time (including the known good metadata of executables, scripts, and other files) makes changes from the baseline more obvious to analysts.

Data collection

Before you start collecting event data that can be used to detect persistence techniques, it can be helpful to assess which sources of evidence capture the metadata most comprehensively.

Consider, for example, the challenges of monitoring changes to the registry using Windows event logs. Many applications use a registry key of some kind for persistence and overwrite it every time the application starts and stops. Imagine this occurring across a universe of 10,000 systems, around the clock, for several years. The important changes can be captured with tools like Event Tracing for Windows (ETW) and Sysmon.



When you consider the costs of using Windows and Sysmon events for persistence hunting, be aware of the following EIDs and events:

–Sysmon ID–	*–Tag–*
12 RegistryEvent	Registry object added or deleted
13 RegistryEvent	Registry value set
14 RegistryEvent	Registry object renamed

–Windows events–	
4663(S):	An attempt was made to access an object.
4656(S, F):	A handle to an object was requested.
4658(S):	The handle to an object was closed.
4660(S)	An object was deleted.
4657(S):	A registry value was modified.
5039(-):	A registry key was virtualized.
4670(S):	Permissions on an object were changed.



Unfortunately, relying solely on these events for hunting persistence presents considerable challenges. If you don't have Sysmon or an easy way to parse Windows events, then Sysinternals Autoruns is a great place to start. This is a free tool (with command line!) to pull artifacts from your environment. The tool is configured to run automatically. You can task it to run on your endpoints and retrieve persistent artifacts. We recommend storing the results in Elastic Stack or another database system because it will make your hunting and querying much more efficient.



Windows isn't the only operating system with a notion of persistence. In fact, every operating system has various forms of persistence. However, gaining visibility into persistence mechanisms at enterprise scale outside of the Windows world can be challenging. One useful tool is OSQuery. Discussed in Appendix A, OSQuery was developed by Facebook and released to the public for use inspecting Windows, Linux, and MacOS systems. OSQuery supports the ability to query a variety of persistence mechanisms.

Simple hunts for persistence

Once you have some events and artifacts, you can begin conducting hunts based on simple questions.

Ideally, each question should be focused on a single form of persistence, for example:

- ☑ Across the enterprise, which persistent objects (executables, scripts, etc.) are using Run or RunOnce keys?
- ☑ Which daemons are running on Linux web servers? Are the hashes custom or part of the National Software Reference Library (NSRL)?
- ☑ Which accounts are modifying persistence mechanisms on MacOS systems?
- ☑ Which persistent objects are signed versus unsigned?
- ☑ Do any persistent objects have a history of initiating network connections (or even the ability to do so)? Where to?
- ☑ Across the enterprise, which logon scripts are being used when authentication succeeds?
- ☑ Which device drivers are persistent?

When you're establishing your first hunts for persistence mechanisms, tackle one cell of the Enterprise ATT&CK matrix at a time. That enables you to measure discrete techniques. After a while you will be able to move beyond these simple questions and use more sophisticated techniques to search and analyze the data, refine your processes, and make them more efficient.

Chapter 5

Hunting for Persistence at Scale

In this chapter

- Review techniques for collecting and analyzing data across the enterprise to find evidence of persistence
- Understand how visualization can reveal key information quickly and at scale

Taking It to the Enterprise

Many organizations begin with simple queries or playbooks, then progress to more complex workflows. In this section, we'll look at a few types of analyses that can be applied at enterprise scale. These include statistical approaches like least-frequency analysis, and differential analyses like baseline comparisons.

Questions to ask at scale

When you take an enterprise-wide perspective, you can start asking questions like these:

- ☑ Of all scheduled tasks, named and unnamed, what are the hashes of the JOB files themselves, and what payloads have they kicked? Sort the results in descending order by frequency.
- ☑ Which persistent binaries make network connections outside the environment, and are there any seemingly unrelated persistent objects communicating with the same destinations?

- ☑ What is the distribution of certificate authorities (CAs) associated with persistent objects across the enterprise? Do any of those CAs have a weak or poor reputation based on open source research?
- ☑ Which Windows persistence objects may be vulnerable to search-order hijacking techniques? (*Quick hint:* DLLs listed in the Known_DLLs key can be eliminated first!)



The Endgame blog is a good source of information on advanced techniques for persistence and how to find them. See, for example: *How To Hunt: Detecting Persistence & Evasion With The COM*.

Frequency and outlier analysis

Frequency-of-occurrence and outlier analyses are two common statistical approaches to finding evil, and they are good ways to begin assessing your environment at enterprise scale. It is a general principle that if the same persistent object is on every system it is too common to be malicious.

Note that this principle is a guideline, not a rule. Self-propagating malware like crypto-miner malware can spread across an unguarded enterprise very quickly, creating persistence mechanisms as it expands its foothold.

In addition to the statistically-based analyses mentioned earlier, you can consider answering questions like these:

- ☑ What payloads associated with any Windows-based persistence mechanism occur least frequently? Most frequently?
- ☑ What is the distribution of network connections associated with previously unknown scripts and executables on Linux systems?
- ☑ What rare and unique (based on hashes) volume boot records (VBRs) exist across all endpoints? What network locations and running processes are unique to those endpoints?



Don't be discouraged if frequency and outlier analysis sometimes produces inconclusive results. Environments are heterogeneous, and you are probably dealing with (relatively) small data sets. But don't give up – sometimes these techniques have a big payoff!

Comparative analysis

What better way to find suspicious persistence items than comparing registry items to a baseline image? (Assuming you have one!) Persistence is one area (among many) where comparisons with baselines can be a very effective method for data reduction.

If you compare persistence artifacts from your baseline (e.g., autoruns) against production systems, you can identify differences between those datasets. This isn't a foolproof hunt, and you may want to dig a little deeper before sounding the alarm, but it narrows data for further inspection.

Some second-order analyses you can perform include examining signing certificates and SSL certificates used in network communication, stacking DNS queries by time-to-live (a shorter TTL value means the DNS record is configured to change IPs quickly), and looking at when associated fully-qualified domain names (FQDNs) were registered.



More suggestions for comparative analysis are available in this Endgame blog post: *How To Hunt: Finding The Delta*. For a detailed discussion on how to create a Windows baseline and use it for comparative analysis, see the SANS Institute white paper: *Quick and Effective Windows System Baselining and Comparative Analysis for Troubleshooting and Incident Response*.

Temporal proximity

You can tell a tremendous amount from looking at the sequence and timing of events: what we might call "temporal proximity" or "contemporariness." It is helpful to know whether other interesting events occurred when persistence mechanisms were created or changed.

For example, if a key was created or modified, what was the order of the operations? Were there corresponding process

events preceding the creation or modification of the key in the registry? If so, the process event is suspicious, and you should investigate that key. You might want to ask:

- ☑ Was a file created and then executed before the key change?
- ☑ Did the executable change the DNS servers configured on the endpoint before resolving an FQDN to an IP address?
- ☑ Did the executable immediately download and execute scripts that communicated directly with IP addresses we've never seen?

If these events aren't consistent with patterns you recognize, you should treat the process event as suspicious, and you should investigate that system.

Data enrichment

It can be incredibly difficult to deal with the staggering volumes of data found in modern enterprises. One way to gain a little control is to improve your data quality through enrichment. You can use popular methods such as:

- ☑ Setting up automatic searches of MD5 hashes in VirusTotal
- ☑ Checking signer information (untrusted files in the registry could be malware)
- ☑ If you have a strict software install process, looking for installed applications in the registry that don't appear on the approved list. (The list should be small if you examine a specific category such as run keys.)
- ☑ Using a sandbox to determine the behaviors of executables based on function imports and dynamic execution
- ☑ Ingesting DNS history and searching for persistence mechanisms that perform DNS lookups

During enrichment, try to capture and compare both local and enterprise conditions. For example, if you observed that HOSTA performed a DNS query for *www.my-evil-domain.org*, make sure you capture what that FQDN resolved to at the endpoint, and at a central location in your enterprise. Except in a few exceptional situations, these should use the same authoritative name server and resolve to the same address.

You are likely to find that applications using distributed DNS will pop up here and there, but as you identify those exceptions, simply document them.

Visualization

Visualization tools can greatly strengthen your ability to understand and interpret hunt data. They are especially important when dealing with large datasets (and autorun data sets quickly become very large).

There are many free tools for visualizing data. Here we discuss some approaches to visualization using D3.js, a JavaScript library that can be used to visualize data.

Here are examples of how you can use visualization to support outlier analysis and category chaining to parse large amounts of autorun data.

Visualization to find outliers

Figure 5-1 is a D3.js radial plot of a type that we jokingly call "the Persistence Flower." The image in this guide is small, but a larger version makes it easy to see which data points appear as outliers.

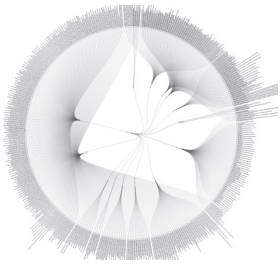

Figure 5-1: A radial plot makes it easy to see which data points appear to be outliers.

You can also use bar graphs and histograms to show data; choose the visualization that makes it easiest for you to identify anomalies.

Category chaining

You can also present data in a hierarchical view. Sysinternals autoruns provide a lot of useful artifacts, and with a visualization tool we can assign levels to all data objects.

In Figure 5-2, D3.js creates a collapsible tree based on a hierarchy we defined as having three levels: category, value, and arguments. This visualization highlights a rogue PowerShell script, and provides insight into scheduled tasks.

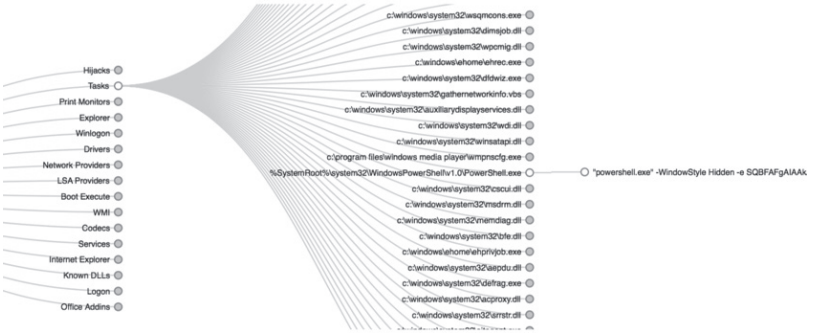

Figure 5-2: A collapsible tree provides a hierarchical view of data.



Information about D3.js is available on the Data Driven Documents website.

Example: WMI

Windows Management Instrumentation (WMI) is Microsoft's implementation of web-based enterprise management (WBEM). WBEM can be described as a way to manage your enterprise using common interfaces. For practitioners who have used WMI, this framework exposes a SQL-like command line utility that is incredibly powerful for both administrators and adversaries.



You can find more information about WMI and how it can be maliciously used in two whitepapers: *Abusing WMI to Build a Persistent, Asynchronous, and Fileless Backdoor* and *WMI for Detection and Response*. You can also see one of the authors of this guide giving a presentation: *There's Something About WMI*.

Without leaving the WMI console, a person can query just about any aspect of a Windows environment. You can use it to gather data for hunting, and attackers can use it just as easily to perform enterprise reconnaissance. WMI can also be used to give malware persistence.

There are a few ways to accomplish this, the most popular being the use of _EventConsumers and _EventFilters. There's an easy way to differentiate between them:

- ☑ An _EventFilter is a condition you test for, or a trigger
- ☑ An _EventConsumer is the result of meeting that condition

If you used an _EventFilter that tested for a specific user to login, you could then configure an action like executing an application.

This may sound obscure, but it's really not that daunting. There are many ways to query changes to WMI, such as WMI itself, Sysinternals Autoruns, PowerShell cmdlets, and several open source tools. If you want to query your environment for _EventFilters and _EventConsumers, you could use the following commands on each system:

```
Get-wmiobject -namespace root\subscription -query
"select * from _EventFilter"

Get-wmiobject -namespace root\subscription -query
"select * from _EventConsumer"
```



If this is your first time looking at common WMI persistence mechanisms, you may need to put in a little work understanding exactly what should be present. Most important, every Windows system should have at least the BVTFilter or the BVTConsumer, which are designed for use with Windows servers. At least one of these should be present on systems with

Windows 7 and above, and they should launch a VBScript called "kern-cap.vbs." You should always confirm the presence of this VBScript and verify that it has not been modified from a known trusted hash. Here's why: threat actors have learned that you can overwrite this VBScript with malicious code on workstations very easily, and with no consequences.

How Endgame Hunts for Malicious Persistence

Without a framework and intelligent automation, hunts can be time-consuming, resource intensive, and unfocused, and produce no results.

MITRE's ATT&CK™ framework provides an array of techniques that can guide hunts in a structured way.

The Endgame platform provides tradecraft analytics to hunt for malicious persistence across all objects within registries, across an entire enterprise environment. These enable hunters to perform their operations quickly and efficiently. The platform automates processes for enumerating all known persistence locations across a network, enriching the data, and performing a variety of analytics that highlight potentially malicious artifacts.

Endgame's persistence hunts provide analysts with a list of applications configured to launch when a system reboots. Our tradecraft analytics for persistence provides unique views that show uncommon and anomalous data for tactics, including COM Hijacking, Search Order Hijacking, Phantom DLL Hijacking, Multiple Hits, Filename Masquerading, and many other tactics that are covered across the MITRE ATT&CK™ matrix.

Chapter 6

Hunting for Lateral Movement

In this chapter

- Understand why attackers need mobility
- Review some of the ways attackers move laterally
- Learn how to determine when PsExec, a tool used by system administrators, is being employed by an attacker for lateral movement and remote execution

Detecting lateral movement is a great way to find evidence of threats. You have to be careful, however, because the same tools and protocols leveraged by threat actors during an intrusion are sometimes used by system administrators for benign purposes.

There are a few ways to classify lateral movement techniques. Here we'll refer to:

- ☑ Protocols that enable remote authentication, such as SSH, SMB, and RDP
- ☑ Frameworks designed for remote execution, such as WinRM, WMI, and RPC
- ☑ Techniques that don't rely on a protocol or framework to support remote access or execution, such as the "Sticky Keys" feature abuse

Why Adversaries Need Mobility

Sometimes threat actors know in advance what users or systems to target to complete their mission. Far more often, though, an attacker who gains a foothold in your enterprise has to undertake a discovery process to gain information about hosts, users, and data of interest. Once a target is identified, the attacker must move across the network to obtain what they need before the environment becomes hostile.

To move laterally, threat actors often employ tools built into operating systems, such as SSH, Windows Management Instrumentation (WMI), and Windows Remote Management (WinRM). Other times the attacker introduces a tool like Windows Sysinternals PsExec. Several of these tools have the option of specifying a target username and password, while others are capable of using the current user context and transparently authenticating to a remote system.



Attackers can even use multi-purpose features of operating systems. An example is the notorious Sticky Keys Attack. This attack has been employed to escalate privileges, enable lateral movement, and provide a poor man's backdoor to hosts. It works because a hidden but well-known accessibility feature allows a user to press the SHIFT key several times to trigger access to an on-screen keyboard. A very minor change to either the registry or filesystem is all it takes to trigger access to the Windows console "cmd.exe." This attack is classified as technique T1015 by MITRE in ATT&CK™ and applies to several other applications.



Some adversaries are capable of exceptionally accurate targeting, precluding the need for lateral movement. This reinforces how important it is for organizations to pursue maximum coverage of their chosen attack framework – such as the MITRE ATT&CK matrix – and the benefits of implementing security controls that inhibit features your organization doesn't use for day-to-day operations.

An Example: Hunting for Suspicious Use of PsExec

At first, hunting for lateral movement can seem like an uphill battle. That's because attackers and system administrators often use the same tools and techniques to move laterally. But the process need not be terribly daunting if you "eat the whale one bite at a time."

In this section we discuss how to detect evidence that someone is using the Sysinternals PsExec tool to interact with a remote system in an unauthorized manner.

About PsExec

PsExec, a utility included in the Sysinternals PsTools suite of software, is one of the more common lateral movement tools associated with remote execution. It is described in product literature as a "telnet replacement" that can be executed using the Windows console or via third-party software.

PsExec has been widely adopted by administrators at organizations of all kinds and is regularly encountered on Windows systems. However, attackers were quick to adopt it for the very same reasons as administrators. In an environment where both admins and adversaries have the same tools, discovering malicious actions can be extremely challenging.



If your organization doesn't employ PsExec, you should implement controls that prevent it from being used, and treat any detected use of PsExec as a security incident.

Technique-based detection

PsExec is a unique tool for lateral movement and remote execution (a) because it isn't native to the operating system, and (b) because of the way it works.

PsExec starts the remote logon process using supplied credentials and performs a quick check to see if it can copy a file and execute it using the hidden $ADMIN share on the target system.

If no errors are received, it unpacks a binary from within itself, "PSEXESVC.EXE," which is executed on the remote host as a

temporary service (PSEXESVC) and then deleted. If $ADMIN isn't available, PsExec will try using another hidden share, $IPC.



There are several hidden shares exposed on every Windows endpoint by default, but they are relatively easy to disable through an update to Group Policy (GPO). If you don't require these hidden shares for a legitimate business purpose, you should disable them so they can't be used by adversaries during the reconnaissance and lateral movement phases of an attack.

Examine Event Logs

You can capture evidence of the use of PsExec on *target* systems by examining several sources of forensic data, which vary according to the target operating system.

Sources of evidence found in Windows events logs include:

- ☑ EID 5145, which contains metadata about requests for access to the hidden $ADMIN and $IPC shares; these logs indicate the responsible process (look for PsExec).
- ☑ EID 5140, which indicates a share was successfully accessed, may confirm that an attempt succeeded, as well as the account used and other supporting evidence.
- ☑ EIDs 4697 and 7045, which record service creation, may capture the installation of the temporary PSEXESVC service.
- ☑ Detailed process execution, captured in EID 4688 events, can identify the use of PsExec on both source and target systems, including full command line arguments.
- ☑ Sysmon, a free logging utility, captures detailed process execution in EID 1 and includes parent process, network, and user metadata.

Teams with access to these sources of evidence should begin by assessing how commonly PsExec appears in the environment, and whether it is known to be used legitimately. From there, they can identify which systems are common sources of PsExec for remote execution, and which accounts are most commonly used for authentication.



Coordinate with IT and network operations to better determine if common tools, accounts, and systems are legitimate. Most security teams don't have the environmental awareness necessary to understand how these resources are used. Relationships with operations groups can also be leveraged during an incident to mobilize a response, support data collection, and implement preventative controls and enhanced logging.



For a primer on investigating the use of PsExec, watch the video of this presentation by Matt Bromiley and Brian Marks: *Skynet Will Use PsExec When SysInternals Go Bad*.

How much data do you need?

Organizations building out a hunt capability may feel compelled to enable excessive logging to cover techniques like lateral movement with PsExec. Retaining the events associated with NTLM and KERBEROS authentication is an excellent decision, but for every minute of system activity many dozens of individual records may be created.

In contrast, the logs associated with share access change much less frequently, so you don't have to search through so much data. For that reason, you can get started by focusing on EID 5145 events and hold off for a while on other sources of evidence. In the next section we'll discuss analysis options for this event type.

Analyze Metadata

We know, based on our understanding of PsExec internals, that it will check the attributes of shares on the target system. We also know, based on our understanding of the Windows operating system, that checking those attributes is something that generates an EID 5145 event.

Begin by analyzing EID 5145 events. These include the following types of metadata:

- ☑ The time the event was recorded (will vary)
- ☑ The source of the request (Service Control Manager)
- ☑ The name of the service (PSEXECSVC, but note that this is configurable)
- ☑ The service executable (%systemroot%\psexecsvc.exe, also configurable)

The service name and executable created on the target are configurable by an adversary. That shouldn't be a major stumbling block when it comes to accessing hidden shares, though. Analysts need to understand *all* the valid services that query share attributes and lead to the generation of these events. In your environment, you should be documenting these valid services. That way, when you see a service with an odd name like "WjjNnsdsd12sdkj" trying to access the $IPC share, you can be certain something is fishy.

Analyze Process Events

In addition to logs of share access and service creation events, evidence of process execution (either native or via Sysmon) can help to reveal the questionable use of PsExec and other malicious executables.

For analysis of running processes, a generic approach can be helpful for getting started. This might consist of asking a few important questions about each process you observe, for example:

- ☑ What was the application and the application metadata (filename, path, hash, size, PE version information, command line arguments, etc.)?
- ☑ Was it recorded during a known operational window?
- ☑ Was a valid account associated with the execution, and was this account used during it's normal operational window?
- ☑ Was the process associated with network activity?



It is a good practice to capture detailed process execution data for all systems. However, some enterprises may find capturing share access events is sufficient for hunting. By forwarding only specific events from endpoints to a central location, you might be able to have your cake and eat it too. Retain the most important events in a central location for hunting, while keeping several days' worth on endpoints for investigative purposes.

New process creation auditing can be enabled on Windows, which causes 4688 events to be recorded. These logs contain a wealth of valuable information about processes but can generate a substantial amount of data. A 4688 event, generated on the source *and* target, contains:

- ☑ The time the event was recorded
- ☑ The user context (account ID, name, domain, session ID)
- ☑ Process metadata (ID, full path to executable, privilege token, parent process ID, parent process full path, full command line)

Think how much faster you could assess whether PsExec is being used for legitimate purposes if you could see exactly how it is used (a thought that applies to any process, not just PsExec!).



Find out how your system administrators and other authorized personnel use tools like PsExec. Taking this information into account will dramatically reduce the number of false positives you need to sort through. If you're using a specific version, with a specific hash and other metadata you've documented, it is even easier to detect legitimate use of these tools.

Analyze Command Line Arguments

For analysis, helpful metadata includes details associated with the process itself. Of these, command line arguments are the most useful. Adversaries can change some of the attributes of PsExec to hide their tracks, but they can't alter the command line arguments it accepts.

Let's look at a quick example where the PsExec utility on the source has been renamed "termsvr.exe":

termsevr.exe \\HOSTA -u
HOSTA\administrator -p
probably.shared.admin.pw –accepteula -s -r
TerminalServiceManager

This command executes PsExec (termsevr.exe) to run on a remote system (HOSTA) using the local administrator credentials (local administrator) and the password ("probably.shared.admin.pw"). The "—accepteula" flag automatically accepts the EULA to prevent interrupted execution, while modifying the registry on the target. The "-s" flag escalates privileges to SYSTEM when possible. Finally, the "-r" flag instructs PsExec to run as a different service on the target (TerminalServiceManager).

If you were looking for the "PSEXESVC" service or the "PSEXESVC.exe" service executable, you'd be sorely disappointed. However, you could look for any executable that uses the "—accepteula" flag and the "\\" network resource prefix.

You can combine this information with the event log metadata we've already discussed to develop an understanding of both normal and abnormal use of PsExec for lateral movement and remote command execution.

Endgame Point of View

Endgame is focused on providing coverage across the techniques of the MITRE ATT&CK™ matrix. The Endgame platform helps organizations pinpoint evidence of lateral movement, including:

- Remote scheduled task (at.exe, schtasks.exe) on SOURCE
- Net share access on SOURCE
- PsExec on SOURCE (assumes default executable names) and TARGET
- Remote service (sc.exe, services.exe) on SOURCE
- WMI (wmic.exe) on SOURCE and TARGET
- PowerShell Remoting on TARGET.

Chapter 7

Credential Theft

In this chapter

- Understand why attackers need to capture and exploit user credentials
- Explore an example of a credential theft technique – KERBEROASTING – and how it can be detected

Credentials are to attackers what bearer bonds are to bank robbers: whoever holds them, owns them. The threat actor who captures the right credentials can access systems, data, networks, and applications almost at will.

In this chapter we discuss why capturing credentials is so important for threat actors, the most common methods they use, and typical targets. We also look at a common KERBEROS-based attack as an example, and at techniques you can use to detect credential theft.

Survival by Any Means Necessary

As reluctant as security professionals are to admit the fact, attackers rarely face much of a challenge gaining a foothold in victim environments.

Once inside, however, the threat actor faces a series of hurdles. The first of these is understanding the configuration of the environment. Like a burglar in a dark house at night, the attacker needs to quickly discover the layout, including the people and the targets (in this case the users, data, and systems of interest).

Even with the discovery process complete, however, the job is far from over. The attacker must find ways to escalate privileges, acquire the ability to move around freely, and map the interesting places.

You should not be surprised to find that there are many ways to capture valid credentials. Among the multitude of options are:

- ☑ Cracking NTLM hashes (which still works in many environments)
- ☑ Dumping clear-text credentials from the Local Security Authority Subsystem Service, LSASS
- ☑ Using Silver and Golden Ticket attacks (both very popular)
- ☑ Cracking KERBEROS service tickets (KERBEROASTING) with weak passwords

To illustrate how attackers work, we examine *KERBEROASTING*, a common way to obtain credentials.

Example: KERBEROASTING

The basics of KERBEROS

KERBEROS is used in Active Directory environments to authenticate users. It is one of the most popular security support providers (SSPs), otherwise known as authentication protocols, available for Windows.

When users on HOSTA want to log onto HOSTB, they type in a domain username and password, and immediately find out if the authentication is successful. Behind the scenes, though, Windows takes a number of steps (illustrated in Figure 7-1):

1. The password is hashed, and an authentication request is sent to the domain controller, which validates the user and hash material.
2. The domain controller sends back a ticket-granting ticket, or TGT.

3. With the TGT, a request is sent to the domain controller on behalf of the user for a ticket-granting service (TGS) ticket.
4. The domain controller validates the TGS request and sends back a reply with the TGS ticket.
5. The TGS ticket is handed off to HOSTB from HOSTA.
6. The user is able to access HOSTB from HOSTA.

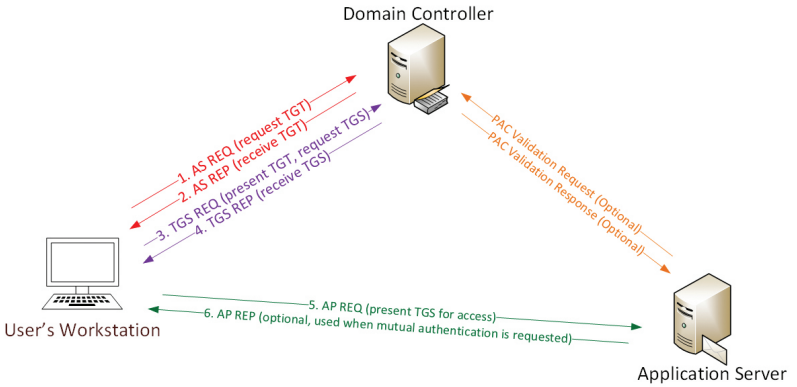

Figure 7-1: The KERBEROS authentication process (diagram courtesy of Sean Metcalf and adsecurity.org)

How attackers KERBEROAST

What is the problem? It concerns service principal names (SPNs). Some SPNs are assigned to privileged groups such as domain administrators. But domain controllers don't care if you query some or all SPNs without actually requesting access to a remote system. As a result, an attacker can look up SPNs with any valid account and use that information for targeting.

Here is how KERBEROASTING works. The attacker:

1. Phishes into the environment and gains a foothold on the workstation of a domain user who is also in the local admins group.
2. Locally escalates privileges using the domain account, and uses a tool to obtain credentials (Benjamin Delpy's Mimikatz does this very well).

3. Uses a native Windows tool that doesn't trigger alerts to query the domain password policy and to query the SPNs of all service accounts (because in the victim environment those don't ever expire, and they have the same rights as a domain administrator).
4. Requests a TGS for one of the SPNs – and the domain controller responds with an encrypted TGS ticket.
5. Uses the Mimikatz output obtained earlier or a dictionary wordlist, and a tool like Hashcat, to begin cracking and obtaining the plaintext password of the target SPN.

Two Techniques for Hunting Credential Theft

Technique-based detection

So, how can we uncover credential theft? The first approach is our old friend, technique-based detection. In the case of KERBEROASTING, you can enable auditing of KERBEROS service ticket operations. This will record an event (EID 4769) for each TGS request. You can then monitor these events and look for patterns that don't make sense for legitimate users. For example, it is highly unusual for a single account to submit multiple requests in a short timeframe. Yet you might observe an event as noisy as a dozen or more records per day for one user. Bingo!

For each TGS request, the record will also list the account, the domain, the hostname from which the request was made, and the encryption metadata. That information allows you to follow up, confirm that credential theft has taken place, and take appropriate actions to disable captured credentials.

Detecting host:user anomalies

Another method of detecting KERBEROASTING involves looking for encryption flags that didn't match the domain default. This technique is not always reliable against sophisticated attackers who have more powerful tools to support KERBEROASTING, but it can still be effective in older and smaller-scale environments.

How Endgame Prevents Credential Theft

The Endgame platform protects enterprises from credential theft attacks that access passwords in memory. For example, attackers often use free, open source tools like Mimikatz to dump and extract cleartext passwords from LSASS. Endgame stops credential access pre- and post-execution by stopping credential dumping, credential manipulation, and credential theft.

Appendix A

Getting Started

For organizations building their hunt capability from the ground up, it can be challenging to get started. Endgame has collected the following free and open source resources into this appendix to help lower the barriers to entry. Readers should be aware that commercial solutions may provide additional features and capabilities.

This section has been broken into two parts. The first describes resources applied to endpoints, and the second discusses solutions for event aggregation.

Endpoint assets

The definition of endpoint assets has expanded recently to include mobile and embedded devices in addition to Windows, Linux, and MacOS systems. Here we focus on traditional operating systems, which represent the biggest challenge for organizations just starting out. For each type of endpoint, we look at native logging applications and share log configuration resources. This information will help you streamline your data collection process and begin hunting faster.

Generally speaking, the default logging facility and settings of an operating system will need changes to become effective.

Windows

Microsoft Windows includes some of the most popular client and server software in production today. At the time of this writing, Windows versions 7 and 8 are slowly losing ground on desktops to Windows 10 – which includes a number of unique security features and enhancements such as Device and Credential Guard. Windows Server 2016 is considered popular, though 2008 and 2012 Windows Server versions are still around.

The default audit policy for Windows is not sufficient for organizations that need to proactively identify threats. Fortunately, for those who don't have the means to deploy other logging tools, this policy can be easily and quickly updated.

The following resources should help you quickly deploy a powerful advanced auditing policy on Windows:

- ☑ Jessica Payne's blog post on Windows Event Forwarding
- ☑ Additional WEF resources
- ☑ Sean Metcalf's article on *Securing Active Directory*

A well-known alternative to the default logging capability is Sysinternals' Sysmon utility. Sysmon provides visibility into DLL loads, detailed command line arguments, parent process metadata, network connections, changes to the registry, and WMI interaction. The "sysmon-dfir" github repository contains some of the most comprehensive information about deploying Sysmon for hunting.

Linux

The Linux operating system achieved a large and very stable base of support many years ago, and represents one of the largest classes of endpoint software for desktops, directory services, web servers, proxies, and numerous other applications.

However, few resources exist for gaining visibility into Linux systems. By default, Linux logging includes authentication data and *cron* events, but doesn't provide much other visibility into historical process or network events. Modern Linux systems may have one or more of the following installed:

- ☑ Syslog, an older and established logging service
- ☑ Rsyslog, which includes some improvements on Syslog
- ☑ Systemd and journald, the logging facilities installed on systems implementing Systemd for system management

To achieve some of the same results that Sysmon delivers for Windows, Linux systems can benefit from the Integrity Measurement Architecture (IMA) system. IMA works with other Linux auditing services to capture and provide process execution metadata based on its configuration.

MacOS

While older version of the MacOS operating system included large collections of logs in different formats, Sierra (as well as the iOS10, WatchOS and tvOS versions) introduced unified logging to replace syslog. The default logging, however, is more useful for developers than for threat hunters and analysts.

Mobile

Organizations are slowly realizing that mobile devices are essential to their success, and are permitting employees to supply their own. These dual-purpose devices are rarely managed at the enterprise level, even though they freely migrate between corporate and non-corporate networks. This means an adversary has a much easier time gaining a foothold through a mobile device than through dealing with your layered corporate security stack, at least in environments where bring your own device (BYOD) is allowed.

At the time of this writing, we have not found a default mechanism for auditing authentication, process, or network events on mobile devices. Further, we know of few open source enterprise solutions – OSQuery does not provide support for Android or iOS devices.

OSQuery and multiple operating systems

Yet another option, which supports several operating systems, is Facebook's OSQuery. This agent-based application logs almost 200 different types of events across Windows, Linux, FreeBSD, and MacOS. The schema and query language uses a familiar, SQL-like format.

OSQuery even offers options for searching containers, file events, DNS requests, Amazon EC2 instances, hardware-based events, browser extensions, software installs, and installed patches.

Aggregation and Storage

Generating the right events to the right level of detail is just one part of an effective hunting capability. It is also essential to have powerful, scalable capabilities to gather those logs in a central place. Many organizations are choosing to implement Elasticsearch with Logstash and Kibana (ELK for short) over transactional databases that may struggle to scale.

ELK stack

Elasticsearch with Logstash and Kibana is a collection of complementary systems that gather data for later searching. Each component handles different functions:

- ☑ Elasticsearch – text searching at scale
- ☑ Logstash – log ingestion
- ☑ Kibana – an analytics engine designed to work with Elasticsearch

An excellent example of the power of the ELK stack is the "Hunting ELK" distribution – created by Roberto Rodriguez.

HELK includes additional features designed by and for hunters, such as support for Jupyter notebooks and various methods of performing enterprise searches. One of the most powerful implementations of HELK relies on the ability of analysts to configure custom dashboards to visualize log data – Mr. Rodriguez has helpfully provided a few on the HELK web site that display detailed summaries of network and Sysmon activity.

Data Collection

After an organization has chosen endpoint software for generating the right data, and settled on a centralized place to collect it, the task of getting the right data *to* the right place takes center stage. There are many free and open source tools to help with this, such as:

- ☑ Syslog, Syslog-NG, and Rsyslog for MacOS and Linux
- ☑ Windows Event Forwarding, NxLog, and WinLogBeat for Windows

Considerations

Data quality is commonly overlooked by organizations developing a hunting capability. But data quality is important if hunters are meant to efficiently and effectively identify threats using it – especially if one objective is automation. To paraphrase Roberto Rodriguez, author of the HELK distribution, data must be:

- ☑ Complete – data for every necessary field
- ☑ Consistent – reliably accurate and complete
- ☑ Timely – delivered in an up-to-date manner

For each type of endpoint, organizations should understand what visibility is provided to them and should adopt a scoring system for measuring those attributes. Organizations cannot fall into the seductive trap of scoring partial or undefined coverage as complete coverage – this cannot be overstated. Instead, refer to the following blog post for a scoring methodology: Ready to hunt? First, Show me your data!

Appendix B

A Hunt Cheat Sheet

This hunt cheat sheet is a resource you can use to look up key information and ideas during your hunts.

Platform

@Cyb3rWard0g, https://github.com/Cyb3rWard0g/HELK

Hardware

Use an Elastic Stack to store your hunting data (DNS, Sysmon, ETW, etc.). System Requirements: OS: Ubuntu-16.04.2 Server amd64, Network: NAT/Bridge, RAM: >=4GB

IPV4 header format (network hunt)

		0	1	2	3
Offsets	Octet	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
Octet	Bit	Version IHL	DSCP ECN	Total Length	
0	0	Identification		Flags	Fragment Offset
4	32	Time to Live	Protocol	Header Checksum	
8	64	Source IP Address			
12	96	Destination IP Address			
16	128	Options (If HL>5)			
20	160				
24	192				
28	224				
32	256				

DNS record (network hunt)

Domain Name: google.com

Updated Date: 2015-06-12...

Creation Date: 1997-09-15...

Ref: elastic.co, whois, docs.microsoft.com

Sysmon event (host hunt)

Event ID	
1	Process creation - provides extended information about a newly created process
2	A process changed a file creation time - file creation time is modified by process
3	Network connection - logs TCP/UDP connections on the machine
4	Sysmon service state changed - state of the Sysmon service (started or stopped)
5	Process terminated -reports when a process terminates
6	Driver loaded - driver being loaded on the system
7	Image loaded - logs when a module is loaded in a specific process
8	CreateRemoteThread - a process creates a thread in another process
9	RawAccessRead - detects reading operations from the drive using the \\.\
10	ProcessAccess - reports when a process opens another process
11	FileCreate - file is created or overwritten
12	RegistryEvent (Object create and delete) - registry key/value create and delete
13	RegistryEvent (Value Set) - registry value modifications
14	RegistryEvent (Key and Value Rename) - registry key/value rename operations
15	FileCreateStreamHash - file stream is created
16	n/a - Sysmon configuration change (cannot be filtered)
17	PipeEvent - Named pipe created
18	PipeEvent - Named pipe connected
255	Error

Analysis Techniques

Search

Searching involves using indicators of compromise to detect malicious activity. These can be file attributes (hashes, filenames, import hashes), network artifacts (domains, IP addresses), registry keys (key values, key sources), and known compromised user accounts and machines. This is a weak approach, because signatures have short life spans.

Frequency and outlier analysis

Frequency and counts of artifacts help discover anomalies. Anomalies do not necessarily represent suspicious activity, but when used correctly they provide leads for investigation. For example, DNS request counts show the occurrences of a registry key, or the least occurring scheduled tasks and WMI objects in the environment.

Comparative analysis

Comparative analysis uses a gold or baseline image to find deltas. The gold image is the clean slate prior to any user interaction. You can compare workstations to the baseline image. This is especially important if your users are unable to install new software or don't commonly do so. Any deviation from that baseline gold image might be an anomaly worth investigating.

Temporal proximity

Using time can be very powerful because it relates to network and event data. For instance, small packets being sent on a routine time interval may be indicative of malware beaconing or showing Windows events in a sequential order. This can illuminate malicious activity, e.g., process create, process execute, DNS request, network connection, process terminate, and file delete.

Data enrichment

Public data sources and threat intel feeds are immensely powerful for data enrichment, not for searching. For instance, you can search file attributes in VirusTotal and search network artifacts in WHOIS databases and tools like Domain Tools or Central Ops.

Quick Wins

How do you detect persistence techniques?

Ref: sysinternals/downloads/autoruns

- ☑ Look for files set to run automatically
- ☑ Pay close attention to outliers

What forensics data should you look for?

Ref: powerforensVics.readthedocs.io

- ☑ Check the Prefetch and Shimcache
- ☑ Get-ForensicPrefetch: file execution forensics
- ☑ Get-ForensicShimcache: AppCompatCache forensics

How do you look for evasion techniques?

Malware files may be named to pose as native Windows files. Compare filenames within %system% to files on disk. Be suspicious when a name matches but the file path does not.

How do you look for injected code?

Look for remote thread creation (e.g. Sysmon thread injection detection), for example:

<CreateRemoteThread onmatch=”include”>

<TargetImage condition=”image”>lsass.exe</TargetImage>

</CreateRemoteThread >

Are your files trusted?

Ref: sysinternals/downloads/sigcheck

Examine certificate information by looking for untrusted processes. Enrich your findings by looking specifically for:

- ☑ Persistent untrusted files
- ☑ Running untrusted processes
- ☑ Running untrusted processes generating network traffic (e.g., netstat)

How do I find credential theft, like KERBEROAST?

Ref: adsecurity.org

- ☑ Frequency of Eventid 4769 - A Kerberos service ticket was requested
- ☑ Alert for KerberosRequestorSecurityToken
- ☑ Search for use of invalid accounts

What file properties are interesting?

Ref: msdn.microsoft.com

- ☑ Examine signer/certificate information
- ☑ Don't trust the file name on disk – compare it to FileVersion Info.OriginalFilename
- ☑ Look for files running out of %temp% or %downloads%

Is this an administrator?

Living off the land techniques use legitimate tools. Monitor PowerShell, WMI, InstallUtil, MSBUILD, RegAsm, and other tools that allow code execution.

What is the IDS rule syntax? [Network searching help]

Ref: Snort Manual

alert tcp any any -> 192.168.1.0/24 111 (content: "|00 01 86 a6|"; msg: "mountd access";)

WHAT IS THE YARA RULE SYNTAX [FILE SEARCHING HELP]

What is the YARA rule syntax? [File searching help]

Ref: yara.readthedocs.io

rule Example { strings: $string = { } condition: $string }



You can check the latest version of the ATT&CK Matrix on the MITRE website.

Don't wait until it's too late!
Hunt down attackers before they cause damage.

Threat hunting is the process of actively looking for signs of malicious activity within enterprise networks without prior knowledge of those signs. It is a proactive approach to uncovering bad actors before they can steal your data or disrupt your business. This one-of-a-kind guide is full of step-by-step instructions and practical advice on how to hunt. Learn how to stop targeted attacks before damage and loss.

- **Threat hunting basics —** review the core concepts of threat hunting in real enterprises
- **Structuring hunts —** learn how to structure hunts and use metrics to assess hunt efficiency
- **Hunting for fileless attacks —** examine techniques for detecting threats that hide in memory
- **Hunting for persistence —** explore how attackers achieve persistence, and how to uncover their techniques
- **Hunting for lateral movement —** find out how to hunt for lateral movement across the enterprise
- **Hunting for credential theft —** discover how to uncover whether attackers have been stealing user credentials

About the Authors

Devon Kerr is a Principal Researcher at Endgame, focusing on adversary simulation, detection, and response technologies. Formerly a Mandiant incident response and remediation lead, he helps Fortune 500 companies detect and contain advanced threat actors.

Paul Ewing is a Senior Threat Researcher at Endgame, where he leads the adversary hunt efforts. He formerly lead hunt teams in government. He specializes in prototyping analytics to detect malicious behaviors and techniques.

CYBEREDGE PRESS

Not for resale

ISBN 978-0-9903543-6

9 780999 035436