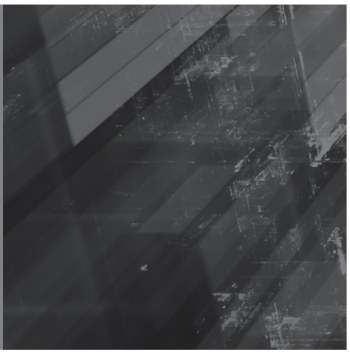
FIRST EDITION | 2019

red canary

Threat Detection Report

An in-depth look at the most prevalent ATT&CK™ techniques according to Red Canary's historical detection dataset



Threat Detection Report

CONTENTS

red canary

REPORT OVERVIEW

A Roadmap for Improving Detection Coverage

MITRE ATT&CK™ has become the security industry's de facto standard for measuring detection coverage and visibility. In 2017, Red Canary adopted the ATT&CK framework across our operations and platform to standardize the way we communicate about threats and detection coverage. We now have multiple years of detection data mapped back to ATT&CK.

HOW TO USE THIS REPORT

This report includes:

- **The most common ATT&CK techniques** observed in our detection dataset
- **Analysis** on why these techniques are so prevalent
- **Detection strategies** for these techniques
- **Industry breakdowns** according to the endpoints we monitor

Whether you're just getting started with ATT&CK or working to improve detection coverage, we hope this report serves as a roadmap to guide your efforts. Security teams of all sizes and industries can turn to this research to focus their efforts on the techniques that adversaries are most likely to leverage and the data sources and detection strategies associated with them.

WHERE THE DATA IN THIS REPORT COMES FROM

Red Canary examines endpoint data at scale in search of adversaries in our customers' environments. Over the span of five years, we have analyzed tens of millions of potentially malicious events. This report is based on a dataset of 10,000 confirmed threats excluding low-severity detections for unwanted software like adware. Each confirmed threat is tagged with the corresponding ATT&CK technique. This data includes information about threats detected from companies of all sizes and in nearly every industry.

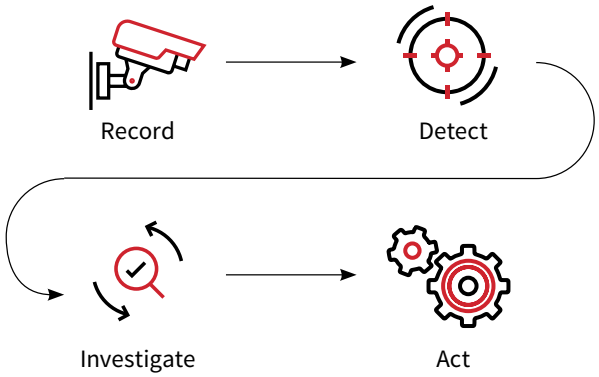
Ultimately, our extensive experience rooting out a diverse array of threats in an equally diverse array of customer environments offers us a unique vantage point from which to produce this report.

Keep in mind that the data in this report is shaped by the nature of our customer engagements. Our friends at MITRE made their own list of the most prevalent ATT&CK techniques, and it differs from ours because it encompasses publicly available threat intelligence reporting that often examines the entire cyber attack lifecycle.

MITRE's findings included many collection and discovery tactics that are absent from our data. Alternatively, our data comes from environments where endpoint telemetry is either closely monitored—where you'd expect to see a lot of execution and defense evasion tactics—or from engagements with our incident response partners where an incident has already occurred—and in which you'd expect to find later stage activity such as lateral movement.

Our platform collects and processes hundreds of terabytes of telemetry from endpoint detection and response (EDR) sensors on a daily basis. We then perform comprehensive analysis of that telemetry to identify and notify customers of malicious software, suspicious behavior, and potential risks such as unwanted software.

How Red Canary Works



About the Authors

Keith McCammon, Chief Security Officer & Co-Founder, Red Canary
@kwm
Keith leads Red Canary's product development and is responsible for the company's security strategy, operations, and threat research. He has spent nearly two decades identifying and solving complex problems related to information security and risk, including more than a decade of service to the United States Department of Defense and Intelligence Community.

Michael Haag, Director of Advanced Threat Detection and Research, Red Canary
@M_haggis
Michael has more than a decade of experience in security architecture and operations. His specialties include advanced threat hunting and investigations, testing, and technological evaluations and integrations. At Red Canary, he works alongside customers to address their organization's unique security needs with strategic vision, research, and technical expertise.

Casey Smith, Director of Applied Research, Red Canary
@subTee
Casey leads research and testing efforts at Red Canary, continually working to understand and evaluate the limits of defensive systems. He led the development of Atomic Red Team, an open-source testing platform that security teams can use to assess detection coverage. His background includes security analysis, threat research, penetration testing, and incident response.

Kyle Rainey, Detection Engineering Lead, Red Canary
@verri3r
Kyle has been providing proactive and reactive incident response and forensics services to Fortune 500 companies for more than five years. He has extensive experience working with organizations to strengthen their security postures and security operations. At Red Canary, he helps lead the development and improvement of detection strategies.

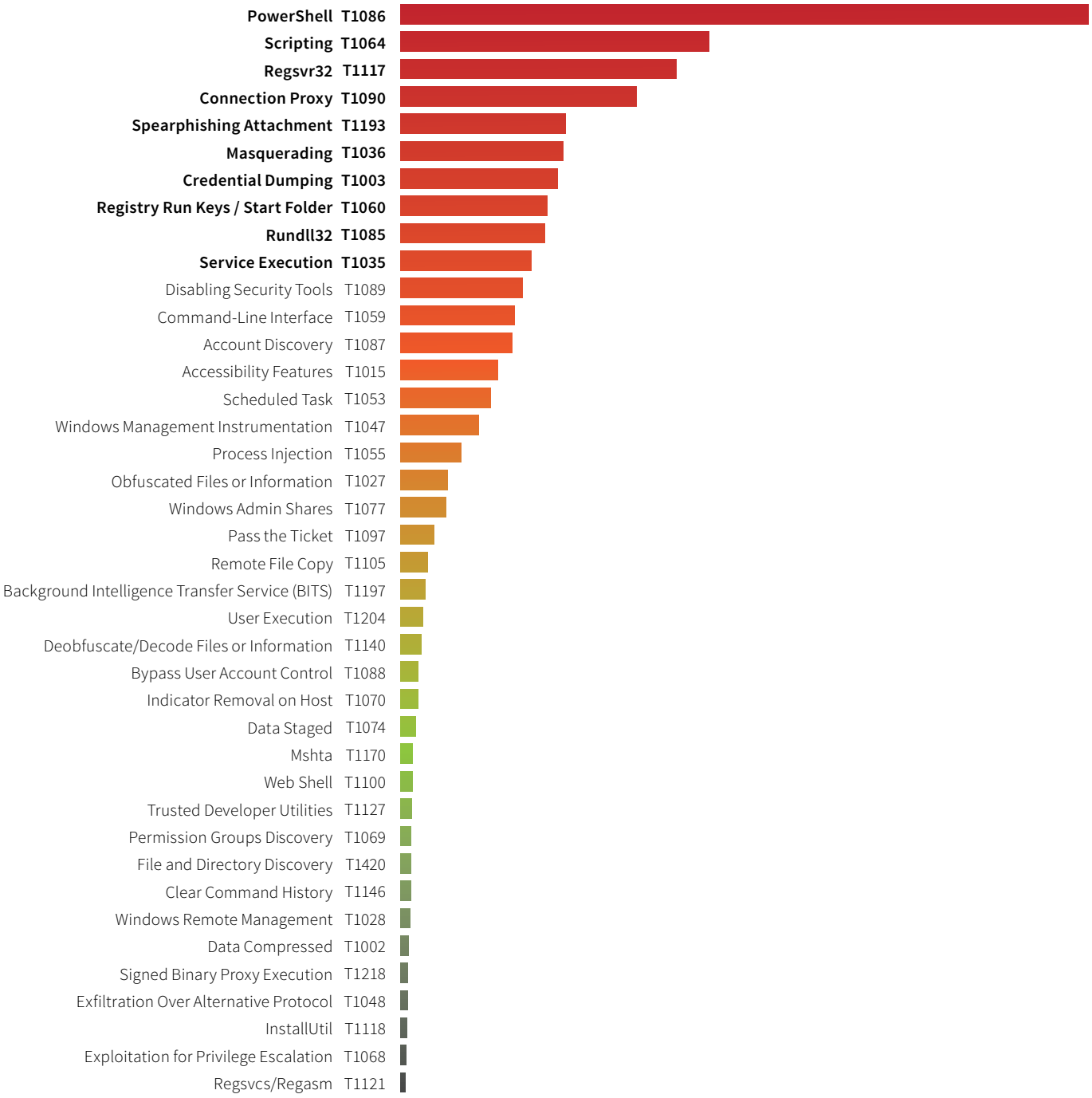
Brian Donohue, Research Production Manager
@TheBrianDonohue
Brian has been writing about and researching information security for the last decade. He started his career as a journalist covering security and privacy. He later consulted as a threat intelligence analyst, researching adversaries and techniques for a variety of major banks, retailers, and manufacturers. At Red Canary, Brian helps guide research production efforts and technical messaging.

ACKNOWLEDGEMENTS

As always, thanks to the MITRE ATT&CK team for their collaboration and dedication. The framework they've created has helped the community take a giant leap forward in understanding and tracking adversary behaviors. The analysis and findings here wouldn't be possible without them.

Top ATT&CK Techniques by Prevalence

This chart illustrates how often each ATT&CK technique is leveraged in a confirmed threat in our customers' environments. To provide a degree of scope to this chart, the top technique is PowerShell, which was a component of 1,774 confirmed threats.



#1 TECHNIQUE: T1086

PowerShell

T1086 Definition:

PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the Start-Process cmdlet, which can be used to run an executable, and the Invoke-Command cmdlet which runs a command locally or on a remote computer. PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

For more, visit attack.mitre.org/techniques/T1086/

WHY IS T1086 PREVALENT?

PowerShell has been included in essentially every Windows operating system by default since 2009.[1] Like a number of techniques on this list, the ubiquity of PowerShell has contributed to its popularity among adversaries. An incredibly powerful framework, PowerShell gives adversaries the ability to perform a multitude of administrative and automation tasks using a utility that is rarely constrained and very unlikely to be blocked outright by system policy.

PowerShell provides full access to Windows API, including hundreds of functions intended for developers and system administrators[2] but leveraged heavily by adversaries alike. Like many core platform utilities, PowerShell libraries are readily available, leading to implementations that expose the full functionality of PowerShell within arbitrary processes.

By virtue of PowerShell becoming available as an open source project and cross-platform utility, we are now encountering highly customizable offensive tooling that can create payloads targeting Windows, macOS, and Linux in ways that are increasingly unpredictable and difficult to fingerprint. Combine all of these factors with an explosion of PowerShell-related offensive suites, such as PSAttack, PowerShell Empire, and dozens of others, and it's easy to understand why PowerShell tops our list of the most prevalent ATT&CK techniques.

PROMINENT EXAMPLES

Turla Group

The Turla Group is renowned for having breached the Department of Defense in 2008, but the suspected state-sponsored espionage group has also targeted defense contractors and government organizations, with a particular focus on embassies and consulates in Eastern Europe. Since about mid-2018, the group has been conducting man-in-the-middle attacks somewhere between Adobe's servers and their victim's machines, injecting otherwise legitimate Flash Player updates with a Metasploit shell and subsequent Meterpreter payload that, in turn, downloads a loader and a backdoor. After infection, the group leverages a custom executable that loads malicious PowerShell scripts directly into memory.[3]

Cobalt Group

Cobalt is the commonly used name for a cybercriminal group famous for targeting banks in and around Eastern Europe. The group is thought to have been responsible for numerous attack campaigns that vary widely in methodology, but the initial infection vector is nearly always an email message containing either a malicious attachment or a URL that leads to the payload. The group uses multiple instances of PowerShell in the later stages of its attacks.[4]

DETECTION STRATEGIES

Data Sources:

- Windows registry
- File monitoring
- Process command-line parameters
- Process monitoring

PowerShell can be leveraged in multiple ways:

- Direct execution of a local script
- Encoded payloads passed via the command line
- Retrieving and executing remote resources using various network protocols
- Loading PowerShell into other processes

There are several indicators that provide detection opportunities with respect to PowerShell. Process monitoring is the most generally effective technique. First, process monitoring allows defenders to baseline the use of PowerShell within their environment. Understanding where it is being used, how, and by whom, is invaluable organizational intelligence that can be used to build simple but highly effective detection criteria. Further, understanding the relationships that PowerShell should have with other processes can make it easier for analysts to spot instances of the tool that have unique parent and/or child processes.

Process command line monitoring takes this a step further, providing insight into PowerShell instances that attempt to pass payloads and otherwise obfuscate intentions via encoded commands. Command line argument parsing is yet another baselining and runtime evaluation technique that is useful for spotting anomalous use.

In addition to the default host for PowerShell scripts, scripts can also be executed in other processes that load the PowerShell framework libraries. To see this behavior, it is important to observe module loads, as well as analytics to provide additional context to support the detection. For example, some binaries may load PowerShell libraries as a normal part of their operation. Again, taking the time to understand what is expected now will make detection and analysis of potentially harmful events easier down the road.

Microsoft has made a tremendous effort to limit the impact and effectiveness of PowerShell attacks.[5] For example, newer versions of PowerShell provide a rich set of features that defenders can deploy to gain insights into how the tool is being used on their systems. PowerShell logging, Constrained Language Mode, and the Windows Antimalware Scan interface (AMSI) are all good ways to begin detecting malicious instances of PowerShell.

WHAT'S AHEAD?

PowerShell is here to stay for administrators and adversaries alike, and those organizations that learn to defend against malicious uses of it will have a distinct advantage. Defending against PowerShell will require not just baselining and an understanding of changes in the ways adversaries use the tool, but defenders will also have to maintain intelligence related to a wide and changing variety of PowerShell attack tools. Furthermore, security teams will need to conduct ongoing, hands-on testing to ensure that controls and analysis processes remain effective against adversarial tools and techniques over time.

[1] "Windows Management Framework is here!" Windows PowerShell team. October 27, 2009.
[2] "Use PowerShell to interact with the Windows API: Part 1." Windows PowerShell team. June 25, 2013.
[3] "Turla: In and out of its unique Outlook backdoor." Tomáš Foltýn. August 22, 2018.
[4] "Multiple Cobalt personality disorder." No author (staff editorial). July 31, 2018.
[5] "Defending against PowerShell attacks." Windows PowerShell team. October 23, 2017.

#2 TECHNIQUE: T1064

Scripting

T1064 Definition:

Adversaries may use scripts to aid in operations and perform multiple actions that would otherwise be manual. Scripting is useful for speeding up operational tasks and reducing the time required to gain access to critical resources. Some scripting languages may be used to bypass process monitoring mechanisms by directly interacting with the operating system at an API level instead of calling other programs. Common scripting languages for Windows include VBScript and PowerShell but could also be in the form of command-line batch scripts.

For more, visit: attack.mitre.org/techniques/T1064

WHY IS T1064 PREVALENT?

Advances in security tooling and human analysis have made it far more difficult for adversaries to introduce overt binary payloads and execute them from disk. As a result, adversaries need to find alternative methods for executing payloads and performing other malicious activities, which accounts for much of the rising prevalence of scripting-related techniques. Add to this the same principles that apply to all techniques relying on native platform utilities such as PowerShell and regsvr32: the runtime environment, libraries, and executables leveraged by this technique are core components of every modern computing platform, cannot be easily disabled, and are not consistently and closely monitored.

Circumstance, however, only tells part of the story. There have also been breakthroughs in methods for escaping script-host constraints, particularly on Windows platforms but also on macOS, which have presented new options for adversaries. What was once difficult via scripting is now trivial; for example, it's very straightforward for an adversary to make an arbitrary API call rather than having to rely exclusively on sanctioned scripting objects.[1]

Beyond the normal scripting hosts such as WScript and CScript, which are the default binaries to host and execute scripts on Windows systems, there are also a number of trusted applications that can execute scripts, including MSXSL and WMIC. This means that scripts can be used by binaries that one may not expect to load and execute scripts. Make no mistake: scripting languages are often as performant and powerful as any binary application. As such, while our wariness about binaries may have necessitated a transition to scripting, scripting has proven a more than sufficient replacement for traditional, native code and the corresponding traditional delivery mechanisms.

PROMINENT EXAMPLES

APT1

The group known as APT1, perhaps the most well-known cyberespionage group of all, was said to have deployed batch scripts in the early reconnaissance phase of its attack campaigns. Known as the subject of a sweeping Mandiant report from 2012, APT1 is thought to have compromised the networks of, and stolen proprietary and other information from, more than 150 organizations in the private and public sectors in primarily English-speaking countries. The group leverages scripting to conduct reconnaissance, including custom batch scripts designed to gather system configuration information, enumerate running services and processes, list accounts with administrative privileges, and gather other important data.[2]

Smoke Loader

From the world of cybercrime, the Smoke Loader trojan, which installs additional malware on the machines it compromises, leverages scripting to launch its payload. More specifically,

Smoke Loader saves a Visual Basic script that automates the execution of its payload in the startup folder for Windows, thereby executing itself each time a user logs in.[3]

DETECTION STRATEGIES

Data Sources:

- Process monitoring
- File monitoring
- Process command-line parameters

Adversaries can leverage scripting in multiple ways:

- Direct execution of a local script via default scripting harnesses. Examples might include Windows Script Host (wscript.exe, cscript.exe) or Python
- Executing within a process that can execute scripts
- On Windows, rundll32.exe with a script host scheme
- Identification and exploitation of vulnerable scripts such as pubprn.vbs

On Windows, the simplest detection use cases for Windows Script Host (WSH) are based on process ancestry. This Includes monitoring for wscript.exe or cscript.exe being spawned from command shells (cmd.exe, powershell.exe), Office applications, web browsers, and web service handlers. It is also advisable to monitor for scripts executing from non-standard locations, such as user-writable paths including appdata\local*, others like it, and temporary directories.

Monitoring process metadata, process command lines, and file modifications are invaluable strategies. In addition, instrumenting systems to observe suspicious module loads of binaries related to hosting scripts, such as vbscript.dll, are worthwhile strategies.

You may choose to disable Windows Script Host, or you can force scripts to be signed, ensuring that only approved scripts are executed. Tools like AppLocker also provide additional constraints related to script execution. These are prevention strategies, but are also useful for detection purposes, as attempted execution of an unauthorized script should produce a higher-quality signal.

WHAT'S AHEAD?

Scripting as an adversarial technique will get worse before it gets better. The dynamic nature and performance of scripts put them on par with nearly any binary resource an adversary could use. As more organizations adopt application control solutions, scripting will become more attractive, and new techniques will evolve.

[1] GitHub: https://github.com/tyranid/DotNetToJScript

[2] "APT1: Exposing One of China's Cyber Espionage Units." Mandiant. February 18, 2013.

[3] "Smoke Loader -- downloader with a smokescreen still alive." Malwarebytes Labs. August 5, 2016.

#3 TECHNIQUE: T1117

Regsvr32

T1117 Definition:

Regsvr32.exe is a command-line program used to register and unregister object linking and embedding controls, including dynamic link libraries (DLLs), on Windows systems. Regsvr32.exe can be used to execute arbitrary binaries . . . [and] can also be used to specifically bypass process whitelisting using functionality to load COM scriptlets to execute DLLs under user permissions.

For more, visit: attack.mitre.org/techniques/T1117/

WHY IS T1117 PREVALENT?

Regsvr32 offers a simple and elegant way for adversaries to execute native code or scripts, either by staging resources locally or by loading them from a remote location.[1] Because the technique leverages a trusted component of the Windows platform that cannot be easily disabled or constrained and detection depends on close inspection of process-level telemetry, this technique remains effective and popular with everyone from purveyors of unwanted software to high-profile actors.

In addition to evading detection by most protection products for well over a year, this technique remains effective due to derivative attack vectors that allow for execution of VBScript and JScript via regsvr32. As a result, these scripts can be used to craft and execute payloads without calling the native wscript.exe and cscript.exe handlers, circumventing detection that relies on these processes and also bypassing Windows Script Host controls.

PROMINENT EXAMPLES

Ocean Lotus Group

Ocean Lotus is a suspected state-sponsored espionage group known to target private companies, government agencies, journalists, and dissidents, with a particular interest in organizations and individuals with ties to Vietnam. The group typically leverages spearphishing emails that social engineer their targets into enabling macros that create scheduled tasks, ensuring that a pair of backdoors can persist through reboots. One of the scheduled tasks used by Ocean Lotus leverages regsvr32 to bypass Windows application whitelisting controls every 30 minutes, ultimately launching a COM scriptlet that downloads later-stage Meterpreter and Cobalt Strike payloads.[2]

APT19

The espionage group APT19 leveraged regsvr32 in a phishing campaign that targeted a handful of law firms and financial services companies around the world in mid-2017. The adversaries developed a macro that leveraged regsvr32 to launch a Windows script component (SCT) file. The SCT file, in turn, launched what appeared to be a Cobalt Strike payload. [3]

DETECTION STRATEGIES

Data Sources:

- Loaded DLLs
- Process monitoring
- Process command-line parameters
- Windows Registry

The regsvr32 technique can be executed by loading a local or remote resource that can be either a DLL or COM Scriptlet. **Thus, an adversary exercising this technique may exhibit all or some of the following behaviors:**

- File modification in the user's profile, either during staging of a local resource or as an artifact of remote resource load

- Network connections initiated by regsvr32.exe processes
- Module loads for scrobj.dll, in the event that the resource is a COM Scriptlet

Detection of this technique requires observation of module loads and the process command line at a minimum. Other valuable data types include process and binary metadata and network connection metadata correlated to process. These data types are available via commercial EDR tools or native monitoring tools such as Sysmon.

This technique and others like it also require an understanding of T1036 or Masquerading. Adversaries have been known to deliver their own copy of regsvr32.exe, copy the local binary to another location, and rename it prior to runtime to evade fragile detection logic that looks explicitly for standard paths and filenames.

WHAT'S AHEAD?

Adversaries are almost certain to continue abusing regsvr32 for the foreseeable future. However, considering general trends in operating system hardening, techniques like regsvr32 are bound to become less effective—even if they don't disappear entirely—in the coming years. In particular, Microsoft is continually adding security mitigations to the Windows operating system, and these are certain to diminish the utility and prevalence of regsvr32 among adversaries.

[1] "Application Whitelisting Bypass: regsvr32.exe." Conscious Hacker. November 17, 2017.

[2] "Cyber Espionage is Alive and Well: APT32 and the Threat to Global Corporations." Nick Carr. May 14, 2017.

[3] "Privileges and Credentials: Phished at the Request of Counsel." Ian Ahl. June 6, 2017.

#4 TECHNIQUE: T1090

Connection Proxy

T1090 Definition:

A connection proxy is used to direct network traffic between systems or act as an intermediary for network communications. Many tools exist that enable traffic redirection through proxies or port redirection, including HTRAN, ZXProxy, and ZXPortMap. … Adversaries could use [proxies] to manage command and control communications, to reduce the number of simultaneous outbound network connections, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between victims to avoid suspicion.

MITRE ATT&CK™

For more, visit: attack.mitre.org/techniques/T1090

WHY IS T1090 PREVALENT?

Connection proxies serve to obscure the identity or location of an adversary. While that's important for a few reasons, it also simplifies a technique that has a wide variety of other, equally important functions.

In addition to providing cover from law enforcement or defensive strategies, proxies also increase adversarial resilience. It's a relatively simple matter to trace an attack back to an IP address and then block it internally, have it blocked by a hosting or Internet service provider, or have it sinkholed through a variety of means. In this way, proxies enable the adversary to simply pack up and move their attack infrastructure to a new location should their original infrastructure become compromised.

Proxies can also serve as discrete methods for adversaries to access and remove information from networks of interest. Adversaries use a wide variety of proxy methods to hide their command and control traffic, including PuTTY/SSH forwarding, Dynamic DNS, domain fronting, fast flux, Tor, i2p, SOCKS, STUN, and host firewall forwarding. We'll examine this in more depth below.[1,2]

PROMINENT EXAMPLES

Duqu

One of the most prominent examples of adversaries using a connection proxy comes to us from Duqu, which first emerged in 2011 and has been attributed to the same actor responsible for Stuxnet. Considering the overwhelming volume of research and analysis that's been written about Duqu and its predecessors, it's difficult to succinctly summarize the threat. However, Duqu was primarily an information-stealing trojan, the main purpose of which was espionage. In terms of connection proxies, Duqu's command infrastructure was set up to forward traffic from compromised machines to proxy servers not affiliated with, and thereby cloaking, the actual C2 server(s).[3]

APT10

The group commonly identified as APT10 is a long-standing threat group known for conducting espionage attacks targeting defense, aerospace, and telecommunications organizations in the United States, Europe, and Japan. While proxy activity probably has very little to do with APT10's notoriety, the attack group has taken something of a novel approach to proxying its attack traffic. They are known to compromise, and subsequently route their traffic through the systems of their target's service providers. In this way, their espionage activity appears to be the legitimate network traffic of companies that work closely with their victims. [4]

DETECTION STRATEGIES

Data Sources:

- Process use of network
- Process monitoring
- Netflow/Enclave netflow
- Packet capture

Adversaries most commonly use connection proxies in the following ways:

- Using proxies for internal or external communication
- Injecting into trusted processes to make connections
- Routing connections through less attributable access points

There are many ways that an organization can get a handle on proxy connections. They should begin by performing a network baseline of egress traffic by geolocation, port, and frequency by endpoint. This provides the visibility required to understand what is normal and abnormal in a given environment. This is achievable through network monitoring. It's also possible to observe this activity in proxy logs—specifically unexpected egress ports.

At the endpoint level, security teams should begin by identifying normal process execution around netsh.exe, PuTTY, Telnet, SSH and other proxy methods. Most of the access to internal or business-related systems will be benign. Therefore, it makes sense to build out use-cases for extraordinary process execution. For example, most employees have never used PuTTY, therefore, if it is executed, it's probably worth examining.

Among the most prevalent forms of connection proxying is the abuse of trusted, core system processes by compromised processes. On Windows systems, malware will tend toward injection into processes such as svchost.exe and others like it, as these are likely to have elevated privileges and thus have or can access explicit proxy configurations that would otherwise prevent an arbitrary process or user from establishing an outbound connection. The best detection approach in these cases is to understand how platforms operate at a lower level, what processes are authorized to communicate via the network, and with which remote endpoints.

Security teams may also want to trace back and identify the source of traffic—specifically the processes that are generating it. It's also a good idea to build out use-cases for identifying or even preventing the use of Tor, Dynamic DNS, and other network services that route traffic to or through less attributable access points. One of the most draconian, but also most effective, approaches is to lock down egress traffic by whitelisting what is needed as opposed to permitting everything.

WHAT'S AHEAD?

Proxies are a necessary part of the internet that are bound to become more popular for both legitimate and illegitimate reasons. Therefore, it's absolutely necessary that security teams develop strategies that allow for benign uses of proxies when needed while also finding methods for preventing adversarial proxying.

[1,2] "Malware Trending: STUN Awareness." Rob Downs. September 30, 2013.

[3] "W32.Duqu: the precursor to the next Stuxnet." Symantec. November 23, 2011.

[4] "APT10 (MenuPass Group): New Tools, Global Campaign Latest Manifestation of Longstanding Threat." FireEye iSIGHT Intelligence. April 6, 2017.

#5 TECHNIQUE: T1193

Spearphishing Attachment

T1193 Definition:

Spearphishing attachment is a specific variant of spearphishing. Spearphishing attachment is different from other forms of spearphishing in that it employs the use of malware attached to an email. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon user execution to gain execution.

For more, visit: attack.mitre.org/techniques/T1193

WHY IS T1193 PREVALENT?

There are a relatively small number of techniques available to most adversaries who seek to gain execution on an endpoint. Spearphishing is among the most popular ATT&CK techniques because it is simple and effective. While technique prevalence varies from one organization or industry to another, T1193 and the broader variations of phishing are among the most commonly observed and most effective techniques in use by adversaries year-over-year. This is due to a number of factors, including human psychology, low cost, target availability, and the ease with which adversaries can improve targeting through open source research.

Spearphishing via attachment, as opposed to similar techniques, allows for the use of a wide variety of file types, and adversaries may exploit the application that handles a given document type or leverage features of the document format, such as scripting or macro languages. Because malicious code, or a bootstrap mechanism that enables retrieval of later stage payloads, is placed directly on the target, the use of attachments has many advantages over other spearphishing variants.

> Phishing succeeds at the intersection of human psychology, technology, simplicity, and target availability.

Phishing succeeds at the intersection of human psychology, technology, simplicity, and target availability. Nearly everyone has an email address, and sending a spearphishing email requires almost nothing of the sender. To that point, the obscurity of an email address is the only meaningful barrier that prevents an adversary from sending a phishing email to their target. Furthermore, the basic design and function of email systems are not well-equipped to prevent anything but the most basic phishing attempts. Psychologically, there is a tendency to implicitly trust email messages, and recipients are accustomed to opening attachments and complying with the directives therein.

Considering these factors, there is a nearly unlimited array of potential targets that are primed to become victims and a similarly deep pool of capable attackers. The barrier of entry is low and the potential value is high. That's a solid value proposition for nearly any attacker.

There are of course complications for an adversary seeking to conduct a spearphishing attack. For one, there is a whole sub-industry of products—discussed in the detections section below—designed to prevent malicious documents from ending up in your inbox. There's also been a major drive to educate end-users. And while it's true that anyone can send an email with a malicious attachment to a specially selected target, not everyone can do it well.

PROMINENT EXAMPLES

Surveiling Tibet
The technique has been a particularly prolific tool among governments seeking to surveil supposed dissidents. The digital and human rights advocates at Citizen Lab showcased a campaign in early 2018 where unidentified adversaries—relying on attack infrastructure thought to cost little more than $1,000—conducted a months-long series of attacks targeting Tibetan activists.[1] Purporting to come from the Central Tibetan Administration, the attackers sent email messages containing attachments that, when opened, redirected to a fake Google login page prompting users to enter their Google account credentials.

Leviathan
In a more sophisticated example, researchers from Proofpoint drew the curtain back on a multi-year campaign in which adversaries targeted defense contractors, universities with military research ties, law firms, and government agencies with email messages containing malicious attachments that exploited recently patched security vulnerabilities.[2] Unlike the previous example, which focused exclusively on access to one of the victim's online identities, this campaign installed malicious payloads and offered the adversary a foothold from which they could leverage multiple post-exploitation tools and techniques, including at least two that are prominently featured in our top ten: PowerShell and Regsvr32.

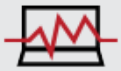
DIG DEEPER
See our PowerShell analysis on page 6, and Regsvr32 on page 10.

Carbanak
The vaunted cybercriminal group known as Carbanak is believed to have used spearphishing attachments as the initial infection vector in some of its attacks as well. According to research from Kaspersky Lab, the criminal group may have stolen as much as $10M from banks around the world in a campaign that began with targeted emails containing malicious Windows Control Panel applets (CPL).[3]

These campaigns were designed to execute malicious shellcode and install backdoors on many thousands of systems and ended in millions of dollars worth of remote ATM cashouts.

DETECTION STRATEGIES

Data Sources:



- File monitoring
- Packet capture
- Network intrusion detection system
- Detonation chamber
- Email gateway
- Mail server

The most common phishing mechanisms are:

- Delivery of malicious software (less common)
- Delivery of malicious documents
- Delivery of a URL lure in the message body or in an otherwise benign attachment
- Simple requests for information or assistance

Note the significant number of data sources associated with this technique. These data sources correlate with each of the above mechanisms and across various stages of a phishing attack. Because of the wide variety of mechanisms and objectives associated with phishing, detection strategies vary widely and defenses should be layered to the extent feasible.

Advances in mail transport policy, phishing intelligence, and local system policy continue to make delivery of an overt software payload difficult. Most mail providers and systems will refuse to transport a message containing any executable software payload, most scripts, and even archive files that cannot be effectively inspected. This is true to a lesser degree of malicious documents, but intelligence and controls continue to improve. And in both of these cases, where a file has been successfully delivered to a user and thus an endpoint, endpoint telemetry is invaluable for detection.

The most common detection strategies for file-based phishing mechanisms include understanding the relationships between file types and the processes with which they interact. This includes:

- Looking for executable (binary or script) files written to disk by browsers, email clients, and other processes associated with the local storage and/or execution of files that are delivered via email.

- Investigating document handlers that have spawned child processes. For instance, Word spawning a command shell, a scripting executor such as PowerShell, and a variety of similar execution harnesses

There have also been recent advances in runtime inspection of document macros, which are valuable controls for prevention, detection, and incident response.

The latter techniques are much more challenging. In many cases, no malicious payload is delivered to the endpoint, and thus looking at process relationships or other overt behaviors yields little fruit. The best detection mechanism in these cases is well-trained people with a keen sense of awareness. People aside, however, there are strategies that leverage one or more of these data sources in novel ways–for early detection or for investigation and scoping.

Leveraging the mail gateway to detect, sanitize, or block URL lures in email based on rules, intelligence, or other attributes can be effective in detecting and mitigating many attacks. A URL that displays as https://www.google.com but that actually points to http://www.ev1l.co is an early and easy detection, and thus prevention, win. Network-based controls, or network metadata collected on the endpoint, can then be used to take the intelligence gleaned from these detections and apply it retrospectively to ensure that messages in the campaign weren't missed.

Similarly, network metadata is extremely valuable for detecting and scoping successful later-stage activity. An organization can follow this standard investigative flow to differentiate between potential and confirmed victims:

1) Identify every employee that received the phishing email

2) Identify those that clicked on the attachment

3) Isolate the subset that provided credentials

4) Look for any misuses of those credentials

WHAT'S AHEAD?

If there's a future for phishing, we can expect that it looks much more like the latter mechanisms than the former. Endpoint platforms are evolving such that native code execution simply isn't an option, and rich document handlers are cloud-based and much less susceptible to the class of attacks that macros have introduced. Instead, the likely trend is toward increasingly clever social engineering, coupled with an increased focus on identity platforms, and the technical means by which adversaries can assume the identity of the victim without needing to traditionally infect the victim's endpoint.

[1] "Spying on a Budget: Inside a Phishing Operation with Targets in the Tibetan Community." Masashi Crete-Nishihata, Jakub Dalek, Etienne Maynier, and John Scott-Railton. January 30, 2018.

[2] "Leviathan: Espionage actor spearphishes maritime and defense targets." Axel F. and Pierre T. (pseudonyms). October 16, 2017.

[3] "The Great Bank Robbery: the Carbanak APT." GReAT (pseudonym). February 16, 2016.

#6 TECHNIQUE: T1036

Masquerading

T1036 Definition:

Masquerading occurs when the name or location of an executable, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. Several different variations of this technique have been observed.

For more, visit: attack.mitre.org/techniques/T1036

WHY IS T1036 PREVALENT?

Adversaries leverage masquerading as a means to evade defensive technology and as a method of deception. Adversaries use the technique in an attempt to trip up machine and human analysis by making malicious executables and software look legitimate or expected. Implementations of masquerading range from simply renaming executables so that they resemble innocuous system processes to more involved methods like command line spoofing.

Masquerading is prevalent among adversaries because it satisfies the simple need to evade defensive technology and human analysis, and it's relatively easy to carry out.

PROMINENT EXAMPLES

SWIFT

The Society for Worldwide Interbank Financial Telecommunication (SWIFT) is a global network that facilitates bank-to-bank financial transactions. In 2016, SWIFT was involved in multiple cybersecurity incidents, the most prominent of which led to the theft of some $80M from Bangladesh Bank. Around that same time, there was another campaign against SWIFT, in which adversaries leveraged masquerading by renaming malware so that it appeared to be the Microsoft binary svchost.exe.[1]

Calisto Trojan

Calisto is the name for a macOS backdoor that emerged in 2016. Designed to steal a variety of information from macOS users, certain versions of the malware masqueraded as a well-known security product vendor to compromise its targets.[2]

DETECTION STRATEGIES

Data Sources:

- File monitoring
- Process monitoring
- Binary file metadata

Masquerading most commonly manifests in the following ways:

- Renaming or relocating files
- Binary metadata manipulation

Detection for masquerading falls into three specific categories: binary metadata modification, expected location, and usernames and process ancestry.

One strategy is to leverage binary metadata, such as the original filename at the time of file creation and/or signing. For example, if we intend to look for wscript.exe, we should look for binaries with this name but also any binary with an original filename of WScript. This enables detection of masquerading where a file has been renamed.

Similarly, we may want to detect any binary with a name or metadata purporting to be wscript.exe that is not trusted based on its signature, hash, or other identifiers.

Establishing a baseline for the location of files is a powerful compliment to the above, though this detection technique can also stand alone. If we understand the path from which a given binary should execute, we can raise a flag if it is observed anywhere else. This technique in particular has proven useful against malware families that bring runtime dependencies with them. These dependencies are often trusted libraries or executable files, and thus abnormal execution paths can help to find what file trust alone cannot.

WHAT'S AHEAD?

Masquerading is an ATT&CK technique that is here to stay. For one, it's not prohibitively complicated to use the technique. To that point, the offensive security community has recently begun integrating techniques such as parent process spoofing and process command line spoofing into open-source and commercial tooling.

These techniques significantly impact detection as they allow the adversary to stitch together an expected process lineage and hide suspicious command lines. These sophisticated implementations of masquerading have the outcome of subverting detection products and human analysts who might think the masked activity is a false positive.

[1] "Cyber heist attribution." Sergei Shevchenko and Adrian Nish. May 13, 2016.

[2] "Calisto Trojan for macOS The first member of the Proton malware family?" Mikhail Kuzin and Sergey Zelensky. July 20, 2018.

#7 TECHNIQUE: T1003

Credential Dumping

T1003 Definition:

Credential dumping is the process of obtaining account login and password information, normally in the form of a hash or a clear text password, from the operating system and software. Credentials can then be used to perform Lateral Movement and access restricted information.

For more, visit: attack.mitre.org/techniques/T1003

WHY IS T1003 PREVALENT?

Credential dumping is a technique derived from the common need among adversaries to access user accounts and other resources within target organizations. Adversaries also leverage dumped credentials to elevate their privilege levels and to move laterally.

Credentials are so important for adversaries that, in many cases, the acquisition of usernames and passwords is not just the means to an end but the entire objective of an attack. Thus, credentials are a salable commodity on a wide variety of criminal forums, and there are websites that track public credential dumps.[1]

Beyond dumping credentials to sell them or use them for initial access, credentials are a critically important part of the post-exploit phase. Once an adversary has gained initial access into an environment, some level of privileged access is often needed to accomplish further objectives in an attack campaign. While there are many ways to elevate privilege levels, one of the most effective and reliable ways is to simply use the legitimate credentials of someone with the requisite privilege level.

PROMINENT EXAMPLES

CosmicDuke

CosmicDuke is the name of an information-stealing trojan that first emerged in 2010. It's thought to have been developed by a suspected, state-sponsored cyberespionage group known by a variety of names, including APT29 and the Dukes. Primarily targeting government-affiliated organizations in the Caucasus region and Central Asia, one of the main modules of CosmicDuke was responsible for stealing credentials from chat programs, email services, and web browsers.[2]

PatchWork

Dating back as far as 2014, PatchWork has been the name of a cyberespionage group that has targeted military and political groups linked to southeast Asia and the South China Sea. The group relies on a long list of techniques in the early phase of its attacks, including spearphishing attachments, scripting, and PowerShell. However, in later stages of the attack, the command and control servers can direct infected hosts to dump login data from the user profile directory and from Google Chrome.[3]

DETECTION STRATEGIES

Data Sources:

- API monitoring
- Process monitoring
- PowerShell logs
- Process command line parameters

Credential dumping comes in various shapes and sizes, but can be broken down into three main implementation categories:

- Accessing hashed credentials
- Accessing credentials in plaintext
- Acquiring key material (most commonly on Linux and MacOS)

Credentials can also be extracted in plaintext from memory. Monitoring for access to specific processes can provide a way for defenders to detect credential dumping. This method of detection is prone to high volumes of false positive events though, as the operating system's built in functionalities accesses these processes as well. Defenders can reduce this noise by focusing their monitoring efforts exclusively on potentially problematic process-to-process interactions, specifically when processes associated with bypass and living-off-the-land techniques, shells and interpreters, or new binaries attempt to access memory stores.

Another opportunity to detect the presence of credential dumping is to profile commonly used tools and develop detection strategies based on the fingerprints left behind using additional data sources as correlation points. Registry keys and file modifications are a good starting point in this regard. Adversaries tend to target not just passwords and hashes but key materials such as certificates, API tokens, and private keys. Detection methods vary in effectiveness depending on how key material is managed from one company to the next, but successful detection of file access is a great place to start.

WHAT'S AHEAD?

Barring a world without passwords, it's hard to imagine a world without credential dumping. However, the technique loses its effectiveness on systems that are protected by multi-factor authentication. Thus, the increased adoption of two-factor authentication may render this technique less effective, even if it remains prevalent based on simple observation.

[1] Pwned Websites. Accessed February 2019. https://haveibeenpwned.com/PwnedWebsites

[2] "The Dukes: Over seven years of Russian cyberespionage." F-Secure. January 29, 2011.

[3] "Unveiling Patchwork - the Copy-Paste Art." Cymmetria. 2016.

#8 TECHNIQUE: T1060

Registry Run Keys / Startup Folder

T1060 Definition:

Adding an entry to the "run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account's associated permissions level.

For more, visit: attack.mitre.org/techniques/T1060

WHY IS T1060 PREVALENT?

Registry Run Keys and Startup Folders have been a long-time target for persistence by adversaries of all flavors. According to Microsoft documentation, support for the registry run keys date back to at least Windows 95. As a persistence mechanism, the technique has a proven, reliable track-record, which, along with ease of implementation, at least partially explains its popularity among adversaries. An adversary needs only user level permissions and the ability to write to the registry or drop a payload to the startup folder.[1]

PROMINENT EXAMPLES

BlackEnergy

BlackEnergy is a famous malware toolkit that's been used for years, most prominently in espionage attacks targeting organizations in and around Ukraine. BlackEnergy's infection vectors have included malicious email attachments and trojans. Later versions of the malware dropped the main DLL into the local application data folder before loading a Windows Shortcut File (LNK) file into the the startup folder to maintain persistence after executing that primary DLL via rundll32.exe.[2]

Lazarus Group

Lazarus Group has been blamed for a long list of attacks in recent years, most notably the major 2014 breach of Sony Pictures Entertainment. While the group is often said to enjoy state sponsorship, it's also apparently financially motivated. In an attack campaign targeting banks and cryptocurrency users, the group sent Word documents containing a malicious VBScript macro that installed implants on the endpoints they compromised. A second stage payload then gained persistence by creating a shortcut link in the the compromised endpoint's startup folder.[3]

DETECTION STRATEGIES

Data Sources:

- Windows registry
- File monitoring

T1060 is most commonly used for persistence. The adversary's goal is to put in place a mechanism that allows their tooling to survive reboots or remediation attempts.

There are two simple methods used by attackers to gain persistence leveraging this technique:

- Installation of run keys in the Windows registry
- Addition of executables or scripts to the startup folder

The implementation, while relatively simple, has proven to be very effective over a long period of time. The technique has evolved over time from referencing executable payloads to loading dynamic libraries and leveraging other techniques such as regsvr32 and script execution.

Detection can be effectively achieved at three different points in the lifecycle of a persistence mechanism: when it is installed, when it is sitting dormant, and when it is triggered.

Detecting run keys and startup folder items at the time of installation requires monitoring changes to specific registry and filesystem paths. These paths can be enumerated via platform documentation, or by referencing utilities that exist solely to report on the presence of these configurations. Additionally, you may find success inspecting any file type, such as LNK, that is known to be used in conjunction with these paths.

To detect persistence that is installed and dormant, the contents of the same registry and filesystem paths can be examined for suspicious entries. Create a baseline and periodically monitor drift from that baseline to narrow the amount of investigation required over time.

Persistence never occurs in isolation, and is always a means to an end. Because of this, it is also effective to monitor the relationship between processes that you expect to make these types of changes and those that have not yet been observed. This plays to the final detection opportunity: understanding what you expect to execute and investigating execution that is statistically less prevalent or not explicitly trusted.

WHAT'S AHEAD?

While there are certainly more complex methods for establishing and maintaining persistence, the simple and effective persistence offered by loading payloads into registry run keys and startup folders is here to stay. As others have suggested, the most cautious and deliberate adversaries will shun overt persistence mechanisms in favor of lesser known techniques and vectors. But persistence being a legitimate function of software will continue to provide cover for even these well-known and traditional approaches.[4]

[1] "Run, RunOnce, RunServices, RunServicesOnce and Startup." Microsoft Support Help article. April 16, 2018.
[2] "Blackenergy & Quedagh: the convergence of crimeware and ATP attacks." F-Secure. No publication date.
[3] "Lazarus Resurfaces, Targets Global Banks and Bitcoin Users." Ryan Sherstobitoff. February 12, 2018.
[4] "Draw me like one of your French APTs – expanding our descriptive palette for cyber threat actors." Martjin Grooten. January 7, 2019.

#9 TECHNIQUE: T1085

Rundll32

T1085 Definition:

The rundll32.exe program can be called to execute an arbitrary binary. Adversaries may take advantage of this functionality to proxy execution of code to avoid triggering security tools that may not monitor execution of the rundll32.exe process because of whitelists or false positives from Windows using rundll32.exe for normal operations.

For more, visit: attack.mitre.org/techniques/T1085

WHY IS T1085 PREVALENT?

Rundll32.exe is a native process installed by default on essentially every Windows operating system since Windows 95. You can use it to call scripts, load DLLs (legitimate or otherwise), and perform network communication. Since it plays an integral role in overall functionality of the Windows operating system, it is not something you can simply turn off, disable, or block.

In other words, given its ubiquity and its capabilities, rundll32 was bound to become a popular target for malicious abuse. Problematically, since rundll32 is responsible for a wide variety of legitimate actions, it offers the attackers who use it the ability to hide their malicious behaviors in plain sight. Considering its necessity, the best thing we can do is monitor its activity for indications of evil and restrict its capabilities accordingly.

PROMINENT EXAMPLES

Poweliks

Back in 2014, Poweliks was one of the major malware families using rundll32.exe. A so-called "fileless malware," it was also noisy, suspicious, persistent, and lived off the land. Poweliks had a way of bypassing preventative security products, and it used rundll32.exe to perform that and other actions, whether it sought to collect system information, download other malware, or perform click-fraud, among other activities.[1]

Powerduke

Reportedly part of an espionage campaign targeting think tanks and non-governmental organizations (NGOs) with spearphishing email messages in the follow-up to the 2016 U.S. presidential election, Powerduke used rundll32.exe to load a malicious backdoor DLL into memory. Upon infection, the backdoor gathered various system information and downloaded, uploaded, and deleted files, to name some of its features.[2]

Flame

Flame is widely considered one of the most advanced malware families ever created. It used rundll32.exe to load its main module early in the infection process, and would later use it in the post-compromise phase to propagate around infected networks. Targeting Windows systems primarily in the Middle East, Flame was a highly modular malware family designed for espionage.[3]

DETECTION STRATEGIES

Data Sources:

- File monitoring
- Process monitoring
- Process command line parameters
- Binary file metadata

The most common ways that adversaries leverage rundll32 are:

- Loading DLLs from non-standard locations
- Performing network communication through rundll32.exe
- Misusing command line parameters

As is commonly the case, establishing a baseline for normal usage of rundll32.exe in an environment will help security teams understand what normal module loads look like. This baseline should include the modules being loaded, the command line parameters being passed to rundll32.exe, and the processes that are calling rundll32.exe in the first place.

To complement or backstop this baseline, look for instances of rundll32.exe establishing network connections, particularly those leveraging protocols that are most likely to transit the perimeter, including HTTP or HTTPS.

While it's true that most teams cannot unilaterally prevent use of rundll32.exe since it performs certain critical operating system functions, use can be constrained in various ways, and reliance on the utility by software developers can be discouraged as it violates various engineering guidelines.[4]

WHAT'S AHEAD?

It seems likely that we will continue seeing evolutions in the way that adversaries leverage rundll32.exe. In particular, we expect to see new methods emerge where adversaries use rundll32 to load additional DLLs directly or via injection.

[1] "Poweliks – Command Line Confusion." Benkow_ (pseudonym). August 20, 2014.
[2] "PowerDuke: Widespread Post-Election Spear Phishing Campaigns Targeting Think Tanks and NGOs." Steven Adair. November 9, 2016.
[3] "Flame: Bunny, Frog, Munch and BeetleJuice." Alexander Gostev. May 30, 2012.
[4] "What's the guidance on when to use rundll32? Easy: Don't use it." Raymond Chen. January 4, 2013.

#10 TECHNIQUE: T1035

Service Execution

T1035 Definition:

Adversaries may execute a binary, command, or script via a method that interacts with Windows services, such as the Service Control Manager. This can be done by either creating a new service or modifying an existing service. This technique is the execution used in conjunction with New Service and Modify Existing Service during service persistence or privilege escalation.

For more, visit: attack.mitre.org/techniques/T1035

WHY IS T1035 PREVALENT?

Service execution is a technique whereby adversaries rely on a native service such as PsExec or WMIC to execute malicious code or to start, stop, or replace another service or program. As such, it's another of the techniques in this list where ubiquity on target systems breeds popularity among adversaries. Adversaries have proven time and again that it is preferable to leverage a native system utility rather than introduce new software, risking detection of the change.

Service execution is as utilitarian as it is ubiquitous, enabling everything from execution to persistence. In more advanced iterations of the technique, adversaries modify active services to execute payloads upon launch, rendering malicious service executions seemingly normal and, at times, difficult to detect. The confluence of these three factors—ubiquity, utility, and normality—make it a highly attractive technique among adversaries.

PROMINENT EXAMPLES

If we were to sort ATT&CK techniques by broadness or narrowness, service execution would certainly land on the broader side of the spectrum. So once again, there are no shortage of examples of adversaries using service execution, especially in recent years as so-called "living-off-the-land" attacks have become increasingly en vogue.

Shamoon

Shamoon is one high-profile example of an attack campaign that leveraged service execution in a destructive strain of malware called Disttracker, which famously targeted and destroyed some 30,000 workstations at Saudi Aramco in 2012. In a later variant of Disttracker, adversaries leveraged service execution to create a process called ntssrv.exe in the system32 folder, which would later execute the malicious wiper payload.[1]

Honeybee

Honeybee is a threat group known to target humanitarian groups with interests in Korean affairs. The group's malware is primarily used to gather system information and exfiltrate data from infected systems. In a campaign from late 2017 and early 2018, the group was using VBScript-laden Microsoft Word attachments to deliver a dropper (called the "MaoCheng Dropper") that used svchost.exe to run a malicious DLL as a service.[2]

DETECTION STRATEGIES

Data Sources:

- Windows registry
- Process monitoring
- Process command line parameters

The most common service execution mechanisms among adversaries are:

- Start and stop services
- Utilizing PsExec (and variants) to spread laterally
- Registry modification for service persistence

In theory, security teams could monitor for all executions of net.exe or sc.exe, but they would end up inundating their team with entirely too much noise. Establishing environmental baselines will help to identify all expected process execution paths. From there, security teams can differentiate between usual and unusual execution paths. Baselining will also help establish how often net.exe is being used to start and stop services. Once it's clear what is normal in an environment, teams can whitelist those normal behaviors that seem benign while looking for the abnormal behaviors that are more likely to be malicious.

Monitoring for PsExec is generally a good idea as well, since not everyone needs or uses it. In certain circumstances, vendor applications will use third-party utilities (e.g., RemCom, PAExec, CSExec) that are similar to PsExec, potentially muddying the water when it comes to detection.[3] However, if an environment uses no such software, these can be blocked or removed.

Problematically, there are numerous locations in the registry where an adversary can persist and modify legitimate services to execute malicious binaries. There are excellent repositories that list these and offer many Linux and Windows artifacts that security teams should monitor to detect adversaries leveraging service execution techniques. [4]

WHAT'S AHEAD?

While service execution is a well-established technique that adversaries have been leveraging for many years, and while it is not going away, there is some reason to believe that its prevalence may fade. Windows 10 S, for instance, has protections to prevent adversaries from loading unsigned binaries or from launching certain executables within the context of a system service.

This and other protections in upcoming operating system releases might limit the effectiveness of certain adversary uses. However, the technique will still offer adversaries a very wide variety of methods for quietly conducting malicious activity.

[1] "FireEye Responds to Wave of Destructive Cyber Attacks in Gulf Region." FireEye. November 30, 2016.
[2] "McAfee Uncovers Operation Honeybee, a Malicious Document Campaign Targeting Humanitarian Aid Groups." Ryan Sherstobitoff. March 2, 2018.
[3] "Threat Hunting for PsExec, Open-Source Clones, and Other Lateral Movement Tools." Tony Lambert. November 19, 2018.
[4] GitHub on ForensicArtifacts: https://github.com/ForensicArtifacts/artifacts.

Industry Breakdowns

What follows is a breakdown of the top ten ATT&CK techniques by industry. We've excluded certain industries from this report because they did not meet our criteria for detection volume. In tandem with the overall top ten list above, we hope security teams can look at the list for their specific industry and use it to guide their detection efforts where reasonable.

COMMUNICATION

T1086: PowerShell
T1035: Service Execution
T1117: Regsvr32
T1064: Scripting
T1003: Credential Dumping
T1090: Connection Proxy
T1036: Masquerading
T1060: Registry Run Keys / Start Folder
T1193: Spearphishing Attachment
T1053: Scheduled Task

EDUCATION

T1086: PowerShell
T1077: Windows Admin Shares
T1053: Scheduled Task
T1064: Scripting
T1003: Credential Dumping
T1060: Registry Run Keys / Start Folder
T1055: Process Injection
T1036: Masquerading
T1089: Disabling Security Tools
T1090: Connection Proxy

ENERGY

T1086: PowerShell
T1193: Spearphishing Attachment
T1059: Command-Line Interface
T1047: Windows Management Instrumentation
T1085: Rundll32
T1035: Service Execution
T1064: Scripting
T1036: Masquerading
T1097: Pass the Ticket
T1003: Credential Dumping

FINANCIAL

T1086: PowerShell
T1064: Scripting
T1117: Regsvr32
T1090: Connection Proxy
T1085: Rundll32
T1089: Disabling Security Tools
T1036: Masquerading
T1193: Spearphishing Attachment
T1015: Accessibility Features
T1060: Registry Run Keys / Start Folder

GOVERNMENT

T1117: Regsvr32
T1060: Registry Run Keys / Start Folder
T1086: PowerShell
T1003: Credential Dumping
T1036: Masquerading
T1193: Spearphishing Attachment
T1027: Obfuscated Files or Information
T1089: Disabling Security Tools
T1015: Accessibility Features
T1105: Remote File Copy

HEALTH

T1086: PowerShell
T1003: Credential Dumping
T1117: Regsvr32
T1059: Command-Line Interface
T1105: Remote File Copy
T1053: Scheduled Task
T1193: Spearphishing Attachment
T1036: Masquerading
T1064: Scripting
T1090: Connection Proxy

HOSPITALITY

T1086: PowerShell
T1064: Scripting
T1036: Masquerading
T1140: Deobfuscate/Decode Files or Information
T1100: Web Shell
T1047: Windows Management Instrumentation
T1193: Spearphishing Attachment
T1090: Connection Proxy
T1015: Accessibility Features
T1003: Credential Dumping

INDUSTRIAL

T1086: PowerShell
T1064: Scripting
T1117: Regsvr32
T1090: Connection Proxy
T1036: Masquerading
T1060: Registry Run Keys / Start Folder
T1085: Rundll32
T1193: Spearphishing Attachment
T1015: Accessibility Features
T1059: Command-Line Interface

MEDIA

T1086: PowerShell
T1193: Spearphishing Attachment
T1090: Connection Proxy
T1064: Scripting
T1036: Masquerading
T1060: Registry Run Keys / Start Folder
T1059: Command-Line Interface
T1003: Credential Dumping
T1015: Accessibility Features
T1089: Disabling Security Tools

NON-PROFIT

T1086: PowerShell
T1059: Command-Line Interface
T1064: Scripting
T1089: Disabling Security Tools
T1117: Regsvr32
T1036: Masquerading
T1003: Credential Dumping
T1193: Spearphishing Attachment
T1060: Registry Run Keys / Start Folder
T1105: Remote File Copy

PHARMACEUTICALS

T1003: Credential Dumping
T1086: PowerShell
T1064: Scripting
T1036: Masquerading
T1193: Spearphishing Attachment
T1035: Service Execution
T1047: Windows Management Instrumentation
T1059: Command-Line Interface
T1055: Process Injection
T1085: Rundll32

RETAIL

T1193: Spearphishing Attachment
T1086: PowerShell
T1064: Scripting
T1059: Command-Line Interface
T1060: Registry Run Keys / Start Folder
T1090: Connection Proxy
T1036: Masquerading
T1015: Accessibility Features
T1089: Disabling Security Tools
T1055: Process Injection

SERVICES

T1086: PowerShell
T1117: Regsvr32
T1047: Windows Management Instrumentation
T1003: Credential Dumping
T1064: Scripting
T1087: Account Discovery
T1193: Spearphishing Attachment
T1060: Registry Run Keys / Start Folder
T1036: Masquerading
T1090: Connection Proxy

TRANSPORTATION

T1086: PowerShell
T1035: Service Execution
T1085: Rundll32
T1064: Scripting
T1117: Regsvr32
T1059: Command-Line Interface
T1060: Registry Run Keys / Start Folder
T1090: Connection Proxy
T1089: Disabling Security Tools
T1036: Masquerading

TECHNOLOGY

T1087: Account Discovery
T1035: Service Execution
T1086: PowerShell
T1090: Connection Proxy
T1089: Disabling Security Tools
T1064: Scripting
T1193: Spearphishing Attachment
T1003: Credential Dumping
T1036: Masquerading
T1117: Regsvr32

CLOSING THOUGHTS

Building comprehensive ATT&CK detection coverage is a massive, multi-year endeavor. We hope that sharing the data we've collected and analyzed over the course of five years arms security teams and the community with a useful roadmap to prioritize detection and analysis where it matters most.

Additional Resources

Join the Community

Sign up to receive a monthly snapshot of Red Canary's best resources, blogs, and staff picks.

JOIN NOW

Get Started with ATT&CK

Just getting started with ATT&CK? Check out our blogs, webinars, research papers, and much more.

BROWSE

Test with Atomic Red Team

Our open source tool makes it fast and easy to test your detection coverage against ATT&CK.

START TESTING

One of many collaborative working sessions that drove the development of this report.

Clockwise, from left: Keith McCammon, Michael Haag, Casey Smith, Adam Mathis, Brian Donohue. Not shown: Kyle Rainey (who was buried in a spreadsheet at the time), and the blood, sweat, and tears leading up to this moment.

red canary

About Red Canary:

Red Canary was built to serve one core purpose: fix the broken information security system. As a security operations ally, we arm businesses of all sizes with outcome-focused solutions to quickly identify and shut down attacks from adversaries.

www.redcanary.com