Defense Against the Lateral Arts: Detecting and Preventing Impacket's Wmiexec

August 31, 2022 | Stephan Wolfert | From The Front Lines

- Impacket, an open source collection of Python modules for manipulating network protocols, contains several tools for remote service execution, Windows credential dumping, packet sniffing and Kerberos manipulation.
- CrowdStrike Services has seen an increased use of Impacket's wmiexec module, primarily by ransomware and eCrime groups.
- Wmiexec leaves behind valuable forensic artifacts that will help defenders detect its usage and identify evidence or indication of adversary activity.

Introduction

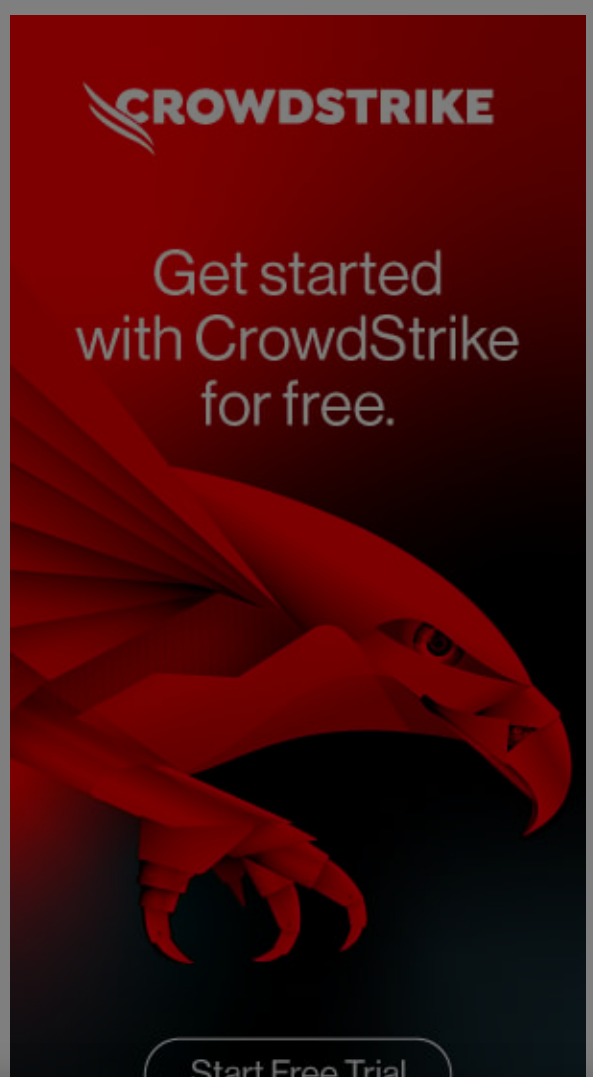
Impacket's wmiexec.py ("wmiexec") is a popular tool used by red teams and threat actors alike. The CrowdStrike Services team commonly sees threat actors leveraging wmiexec to move laterally and execute commands on remote systems as wmiexec leverages Windows native protocols to more easily blend in with benign activity. CrowdStrike has also identified threat actors packaging wmiexec using PyInstaller to run it as an executable on Windows systems, remotely executing data exfiltration tools such as Rclone, and Cobalt Strike beacons for lateral movement and command-and-control operations.

Impacket's suite of tools is extremely versatile and is low impact, making

CATEGORIES

Cloud & Application Security 104
Counter Adversary Operations 184
Endpoint Security & XDR 307
Engineering & Tech 78
Executive Viewpoint 162
Exposure Management 84
From The Front Lines 190
Identity Protection 37
Next-Gen SIEM & Log Management 91
Public Sector 37
Small Business 8

CONNECT WITH US



ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. Cookie Notice

Accept All Cookies Reject All Cookie Settings

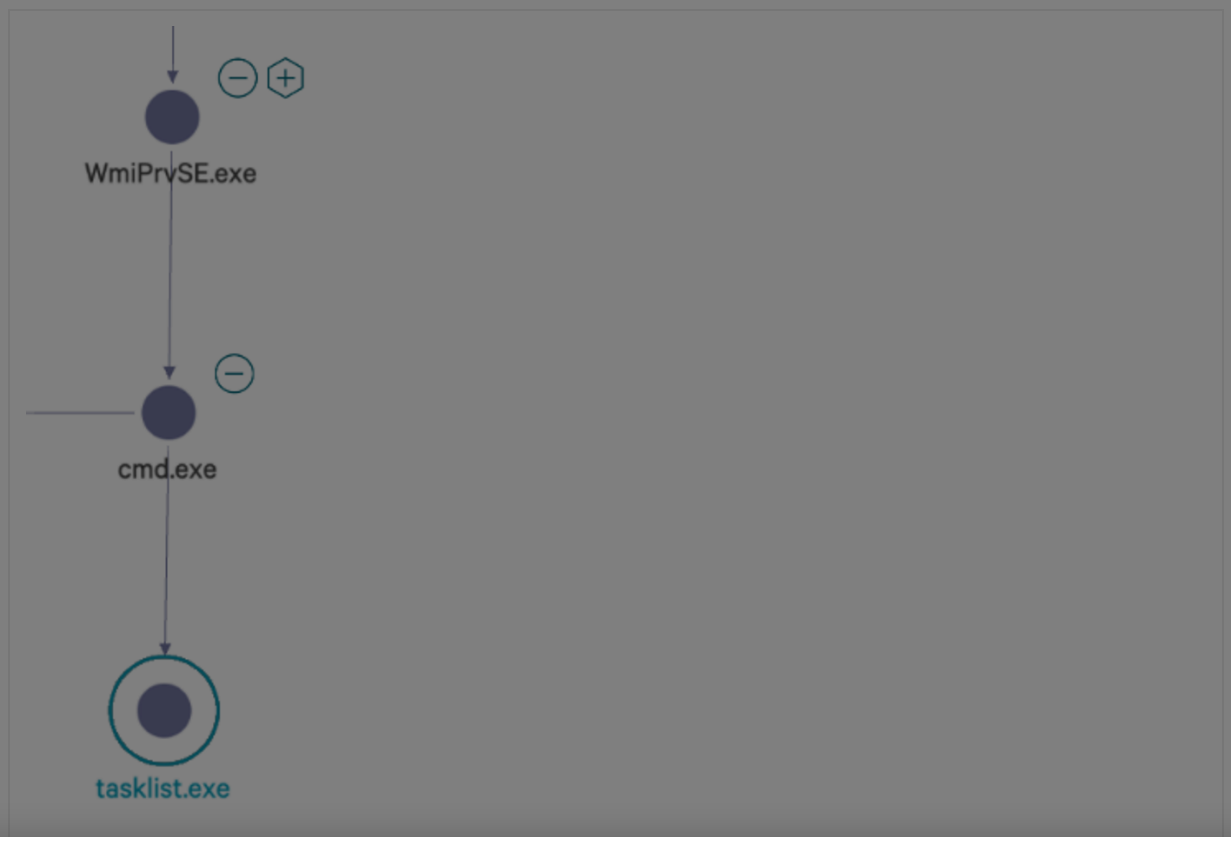
operating systems." While WMI has legitimate use-cases, threat actors commonly use WMI to move laterally.

Wmiexec allows a threat actor to execute commands on a remote system and/or establish a semi-interactive shell on a remote host. The remote connection and command execution requires using a valid username and password or an NTLM hash. Usage of wmiexec does not require the remote service installation that similar lateral movement techniques require, such as smbexec.py by Impacket.

Wmiexec uses Distributed Component Object Model ("DCOM") to connect remotely to a system. DCOM allows COM objects to communicate over the network. The threat actor's execution of wmiexec.py will establish their connection with DCOM/RPC on port 135, the response back to the threat actor system is sent via the Server Message Block ("SMB") protocol.

Initial Indicators

When hunting for wmiexec, defenders should look for WMI usage. A defender's first step should be to analyze the process relationship involving a parent process known as `WMIPRVSE.EXE`. Suspicious processes such as `CMD.EXE` or `POWERSHELL.EXE` running as a child process to `WMIPRVSE.EXE` are a red flag. Most commonly, and by default, wmiexec will use a child process of `CMD.EXE`. A common indicator of wmiexec is the command line switches of the `CMD.EXE` process, which is somewhat unique. An example of executing a tasklist using wmiexec would establish a process relationship similar to the image in Figure 1. Throughout this blog we will often refer to the publicly available source code on Impacket's GitHub repository.



September 25, 2024

Recognizing the Resilience of the CrowdStrike Community

September 25, 2024

CrowdStrike Drives Cybersecurity Forward with New Innovations Spanning AI, Cloud, Next-Gen SIEM and Identity Protection

September 18, 2024

SUBSCRIBE

Sign up now to receive the latest notifications and updates from CrowdStrike.

Sign Up

See CrowdStrike Falcon® in Action

Detect, prevent, and respond to attacks—even malware-free intrusions—at any stage, with next-generation endpoint protection.

See Demo

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. Cookie Notice

is set to stop after the command specified by the string is carried out.

```
127                 self.__shell = 'cmd.exe /Q /c '
```

Figure 2. Code example calling of cmd.exe

Identifying commands issued with the default `cmd.exe /Q /c` arguments are an indicator that wmiexec may be in use but note that these are parameters which can be used for legitimate purposes. To further validate the identification of wmiexec usage, another indicator is command redirection. During execution of *wmiexec*, the command is redirected to a file created on the remote host's local ADMIN$ share by default. The ADMIN$ share aligns with the file path `C:\Windows\`.

```
295                 command += ' 1> ' + '\\\\127.0.0.1\\%s' % self.__share + self.__output + ' 2>&1'
```

Figure 3. Code example of redirected output to a file (Click to enlarge)

As shown in Figure 3, on line 295 of the wmiexec code, the command variable has a few variables that are appended with additional data, concatenating the `/Q /c` switches with the command being run and the redirection. While this full command line is a great indicator of wmiexec usage, the variable `__output` (shown in Figure 3 as `self.__output`) is the name of the temporary file written to disk and creates additional forensic artifacts on disk. When the tool concatenates the command parameters together, an example final resulting command is shown in Figure 4 where the threat actor is attempting to execute `hostname`.

```
cmd.exe /Q /c hostname 1> \\127.0.0.1\ADMIN$\__1645635365.8908157 2>&1
```

Figure 4. Full wmiexec command result (Source: Windows Event Log)

The output variable containing the name of the file that is to be written to disk is declared in two places, shown in Figure 5 of the wmiexec code. First on line 46, the output of the command is given a filename of two underscores, `__`, followed by the current time in EPOCH. This is important for two reasons: the presence of this file indicates execution of (or attempted execution of) wmiexec and also it will give us a time of execution. As previously mentioned, this file will reside in `C:\Windows\`, which is remotely accessible using the ADMIN$ share. An example of the file written to disk is shown in Figure 6. The output file stores the results of the command executed to be sent back to the threat actor, which can also be useful to defenders (more on this later).

```
46  OUTPUT_FILENAME = '__' + str(time.time())
```

```
125                 self.__output = '\\' + OUTPUT_FILENAME
```

Figure 5. Code example of the output filename

```
69 __1645636159.6177979
```

Figure 6. Example of output file written to disk

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

command.

```
285             self.__transferClient.deleteFile(self.__share, self.__output)
```

Figure 7. Code example of file cleanup after execution

An example of a failed cleanup operation is if an interrupt command such as CTRL+C was issued while wmiexec is running, which would leave behind a file on disk. In Figure 8, execution of wmiexec is performed using the NTLM hash of a user `winadmin` to the host `INFOSEC-PC1` in a test environment. Executing wmiexec without a command will establish a semi-interactive shell — from an threat actor perspective, this means a shell is established that appears interactive, but each command executed will be sent through the wmiexec channel and include the standard process execution `cmd.exe /Q /C <command> 1> \\\\127.0.0.1\ADMIN$\__<EPOCHTIME> 2>&1` command format. In Figure 8, after running nslookup an abort command is issued (CTRL+C) and the nslookup process is interrupted. The result is a file on disk containing the output of the command executed, shown in Figure 9. Note: In Figure 9, an abort command needed to be issued since nothing was supplied to nslookup, and although this was done in a test environment, CrowdStrike Services has seen this exact occurrence happen during real incident response engagements.

```
impacket$ python3 examples/wmiexec.py -hashes :58a4[redacted]fdb71 winadmin@INFOSEC-PC1
Impacket v0.9.25.dev1+20220218.140931.6042675a - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>nslookup
^C[-]
```

Figure 8. Execution of wmiexec.py to interrupt nslookup execution (Click to enlarge)

```
-a----         2/23/2022  12:09 PM             69 __1645636159.6177979


PS C:\Windows> cat __1645636159.6177979
Default Server:  [redacted].com
Address:  192.168.1.1

>
PS C:\Windows>
```

Figure 9. Examining the leftover file contents

Another example of failure to clean up and artifacts left on disk is shown in Figures 10 and 11. In Figure 10, wmiexec is executed similarly as before, but this time `powershell.exe` is executed. Since running the command powershell.exe would open a new PowerShell prompt, issuing another abort command (CTRL+C) will result in the output file not being cleaned up. Figure 11 shows the header of a new PowerShell window in the contents of the output file, `__1645636397.851114`.

```
impacket$ python3 examples/wmiexec.py -hashes :58a4[redacted]fdb71 winadmin@INFOSEC-PC1
Impacket v0.9.25.dev1+20220218.140931.6042675a - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>powershell.exe
^C[-]
```

Figure 10. Execution of wmiexec.py to interrupt PowerShell execution (Click to enlarge)

```
-a----         2/23/2022  12:09 PM             69 __1645636159.6177979
-a----         2/23/2022  12:13 PM              0 __1645636397.851114


PS C:\Windows> cat __1645636397.851114
Windows PowerShell
```

ABOUT COOKIES ON THIS SITE

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

As shown in Figure 12, contextual information can be parsed from Prefetch using a tool such as PECmd. This contextual information includes Dynamic Link Libraries (DLLs) and other files used by the process that was executed. In this example, a variety of DLLs are referenced in the Prefetch file for `whoami.exe` as well as the temporary files associated with wmiexec. The temporary files previously discussed cannot be extracted from the Prefetch file — but, this example does show that Prefetch data contains evidence that `whoami.exe` was executed with wmiexec.

```
14: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\UCRTBASE.DLL
15: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\WS2_32.DLL
16: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\SHLWAPI.DLL
17: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\COMBASE.DLL
18: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\BCRYPTPRIMITIVES.DLL
19: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\VERSION.DLL
20: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\AUTHZ.DLL
21: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\SSPICLI.DLL
22: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\WKSCLI.DLL
23: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\NETUTILS.DLL
24: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\BCRYPT.DLL
25: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\EN-US\WHOAMI.EXE.MUI
26: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\GLOBALIZATION\SORTING\SORTDEFAULT.NLS
27: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\__1658938368.1822846
28: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\__1658938223.9829855
29: \VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\IMM32.DLL

---------- Processed C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf in 0.14051020 seconds ----------
```

Figure 12. Examining parsed Prefetch data to identify a relation between whoami and temporary wmiexec files

Key visibility into potential wmiexec execution comes from command-line process execution. Process execution will give a defender more definitive evidence and visibility into the exact commands being executed regardless of actions taken by the threat actor. Relying primarily on local logging, the Windows Security Event Log can provide granular data on the command line execution with the right settings enabled. Unfortunately, Security Event ID 4688 is **not** enabled by default. After enabling Event ID 4688, defenders must also configure it for full command line auditing, which is described below. Note: When introducing any recommendations for security or visibility enhancements to your environment, proper testing should be conducted to ensure it will not affect business operations negatively. To enable Event ID 4688, open the Local Group Policy Editor and do the following (shown in Figure 13): Local Group Policy Editor > Open Computer Configuration > Open Windows Settings > Security Settings > Advanced Audit Policy Configuration > System Audit

ABOUT COOKIES ON THIS SITE

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

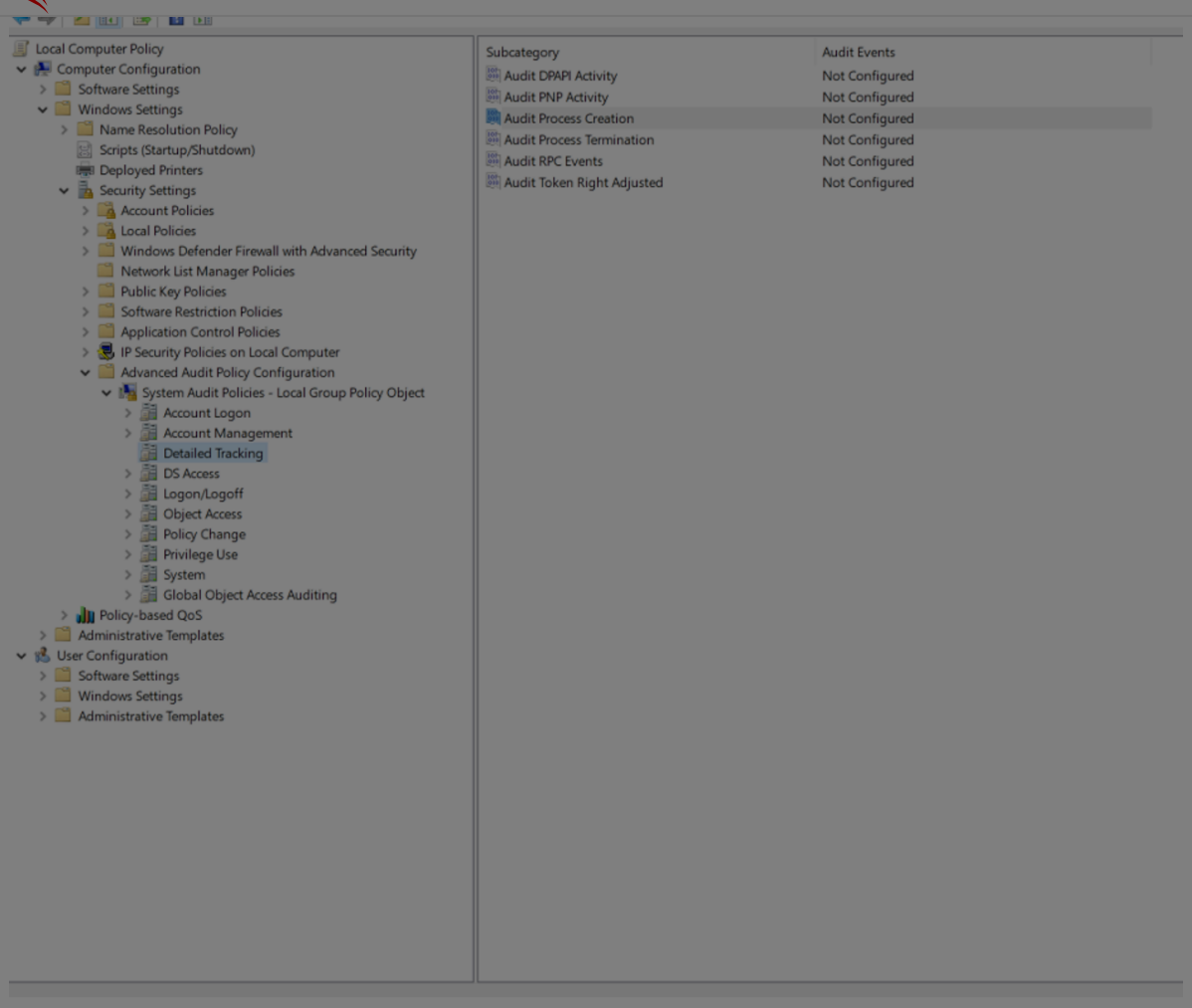


Figure 13. Local Group Policy Editor to enable Event ID 4688

In addition, the box for configuring both success and failure events should be checked, and then click apply, as shown in Figure 14.

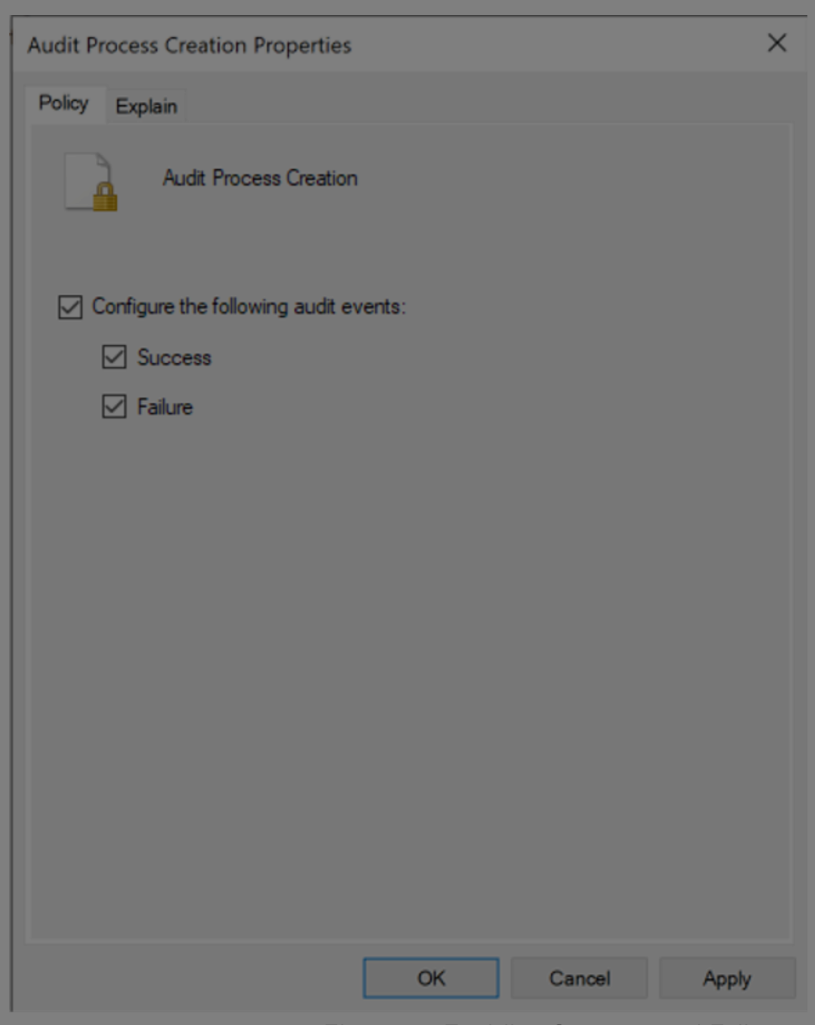


Figure 14. Enabling Success and Failure events

Finally, check to make sure auditing is enabled, as shown in Figure 15.

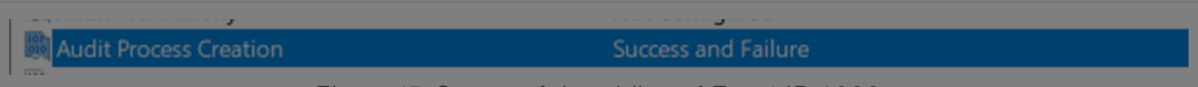


Figure 15. Successful enabling of Event ID 4688

After enabling Event ID 4688, the Windows Security Event Log will log created

ABOUT COOKIES ON THIS SITE

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

General Details

A new process has been created.

Creator Subject:
- Security ID: NETWORK SERVICE
- Account Name: INFOSEC-PC1$
- Account Domain: WORKGROUP
- Logon ID: 0x3E4

Target Subject:
- Security ID: NULL SID
- Account Name: winadmin
- Account Domain: INFOSEC-PC1
- Logon ID: 0x40C6A82

Process Information:
- New Process ID: 0x7dc
- New Process Name: C:\Windows\System32\cmd.exe
- Token Elevation Type: %%1936
- Mandatory Label: Mandatory Label\High Mandatory Level
- Creator Process ID: 0x894
- Creator Process Name: C:\Windows\System32\wbem\WmiPrvSE.exe
- Process Command Line:

Figure 16. Event ID 4688 enabled showing process creation of wmiexec

Although visibility into parent/child processes on a system helps defenders, the full process command line in Figure 16 is blank because it must be enabled separately. In order to enable the full process command line, open Local Group Policy Editor and navigate to the following location (as shown in Figures 17 and 18): Local Group Policy Editor > Computer Configuration > Administrative Templates > System > Audit Process Configuration > Include command line in process creation events > Enable

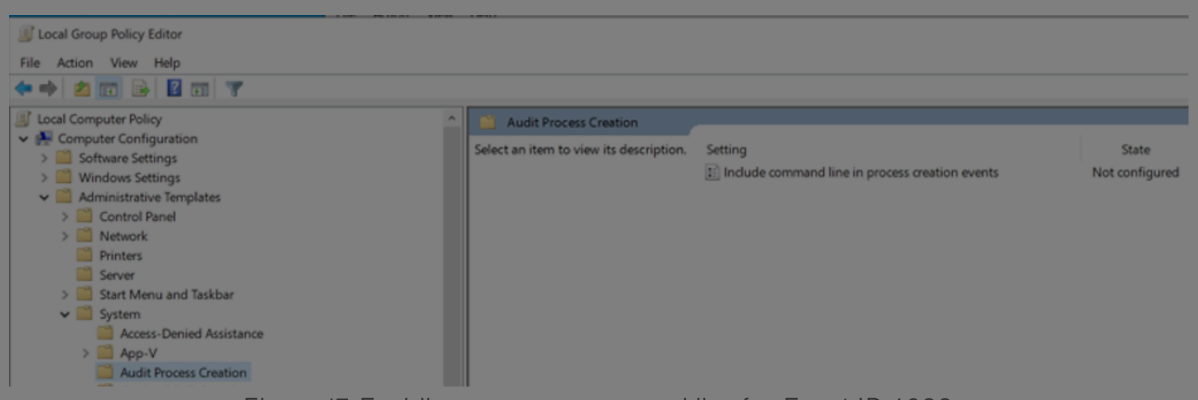


Figure 17. Enabling process command line for Event ID 4688

ABOUT COOKIES ON THIS SITE

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

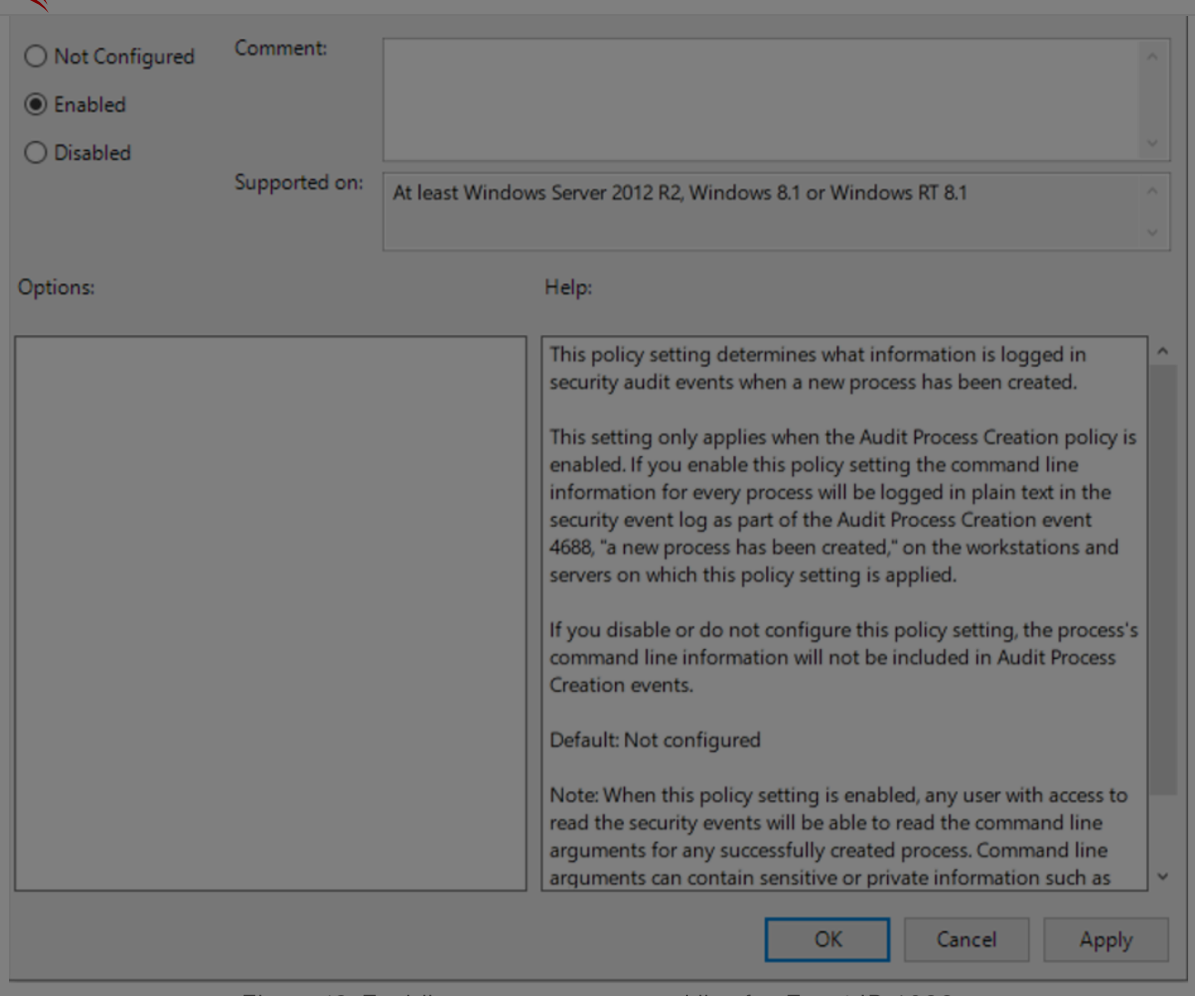


Figure 18. Enabling process command line for Event ID 4688

With process command line auditing enabled for Event ID 4688, more granular detail on the exact commands executed is recorded, rather than just parent/child process execution. Forwarding this data to a centralized security information and event management (SIEM) tool can allow defenders to set up alerts and dashboards to track process executions. As shown in Figure 19, Event ID 4688 is now logging the entire command line, giving a defender the ability to see wmiexec was used to remotely execute the command `hostname`.

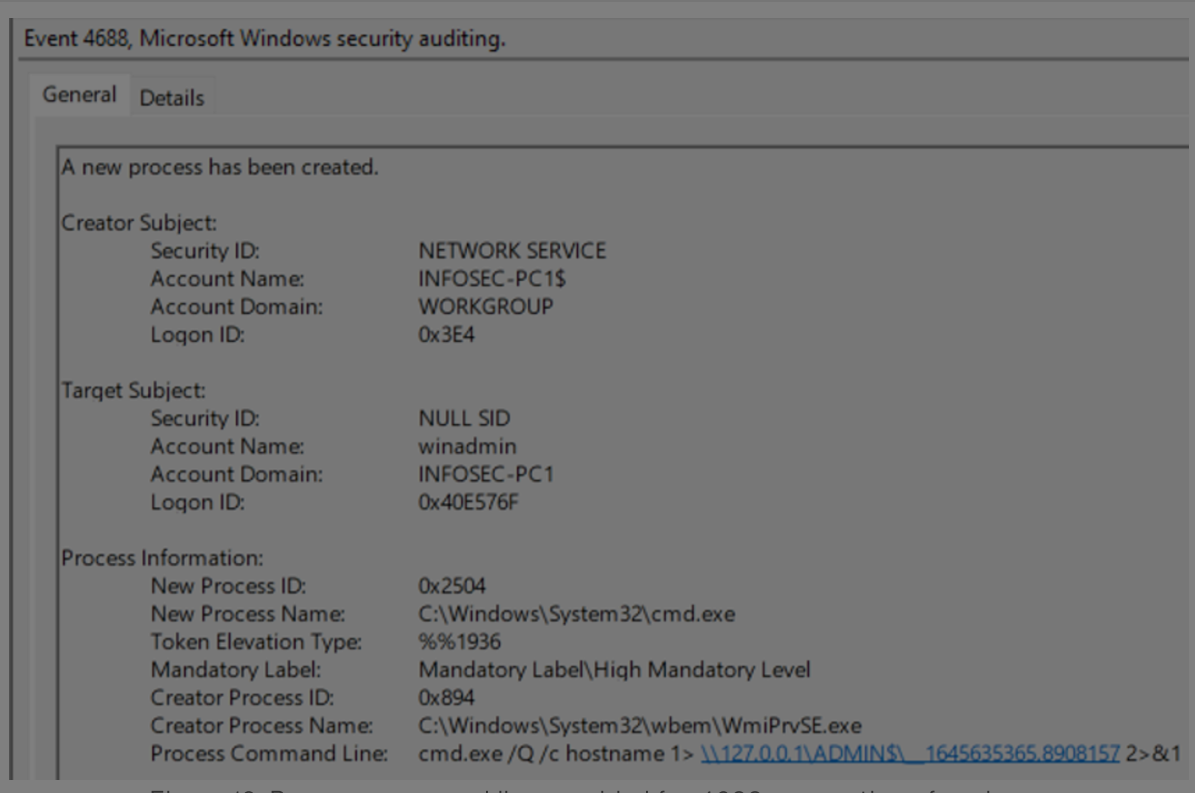


Figure 19. Process command lines enabled for 4688 – execution of wmiexec

Other resources such as antivirus detection logs and the Microsoft Protection Log ("MPLog") are also good resources for gaining command line visibility into wmiexec usage. This CrowdStrike blog discusses how to leverage MPLog in greater detail.

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

ability to run commands and while visibility into these attacks is critical, mitigating them is the ultimate goal.

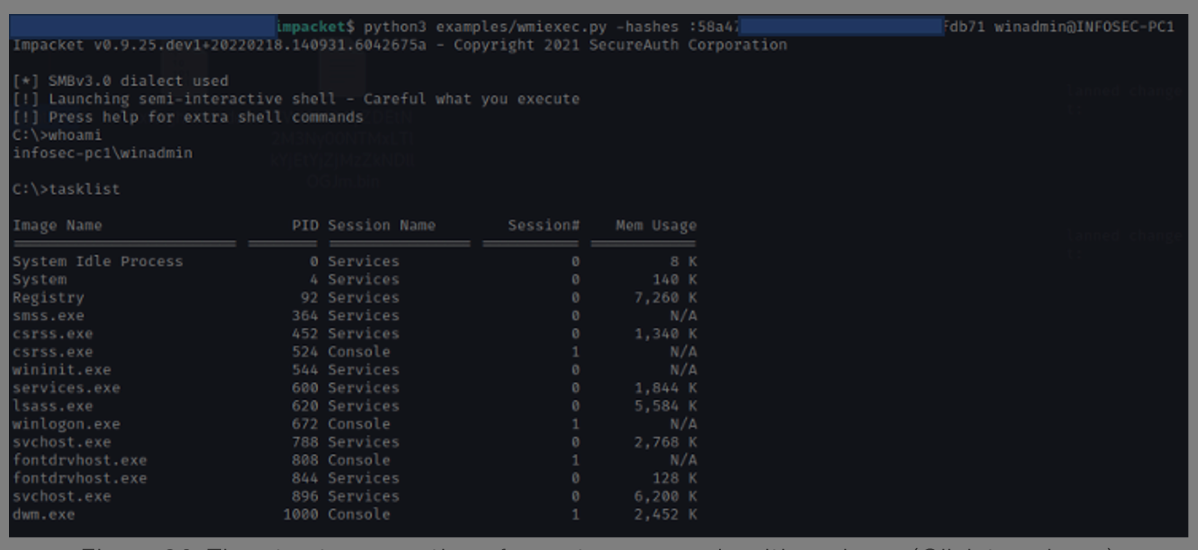


Figure 20. Threat actor execution of remote commands with wmiexec (Click to enlarge)

CrowdStrike Falcon® Insight™ endpoint detection and response will collect granular data on process execution in real time, going into more detail than the standard process creation logging available in Windows. Indicators of attack (IOAs) are a method used to create detections and preventions for wmiexec. Commonly on CrowdStrike Services Red Team/Blue Team assessments, CrowdStrike experts will assist internal teams to configure an IOA that will allow detection and prevention of wmiexec. In Figure 21, having a proper IOA and then attempting to execute `tasklist` again shows not only the ability to see the full process parent relationships but also how to prevent it.

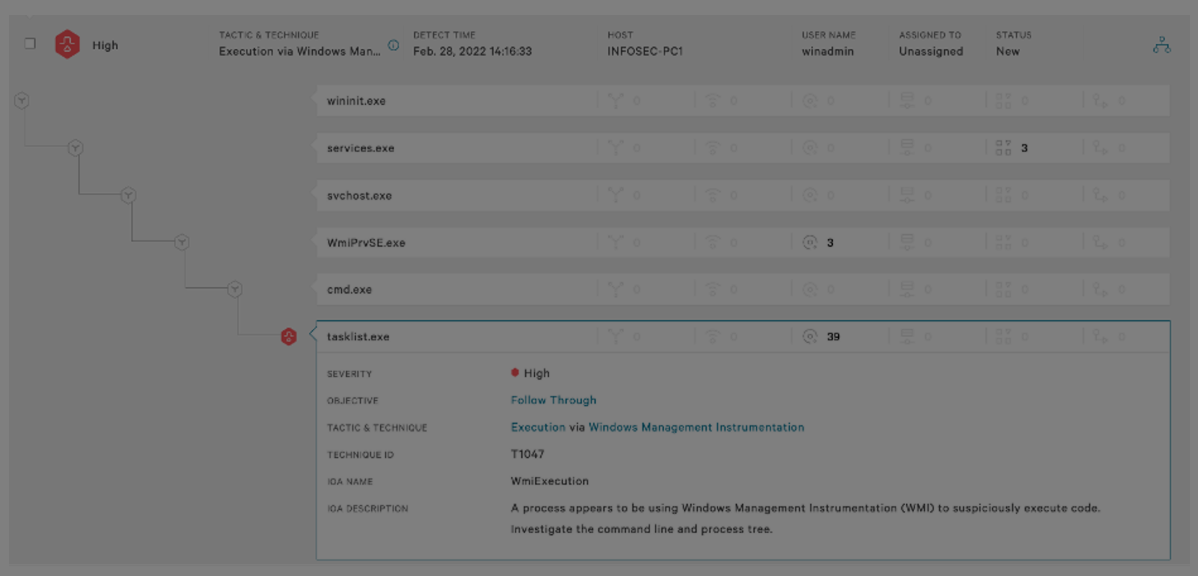


Figure 21. CrowdStrike Falcon® IOA for wmiexec (Click to enlarge)

Each of the processes that spawned the suspected malicious process can be examined to identify a gold mine of information, including the host where the request originated and full command line parameters, as shown in Figure 22.

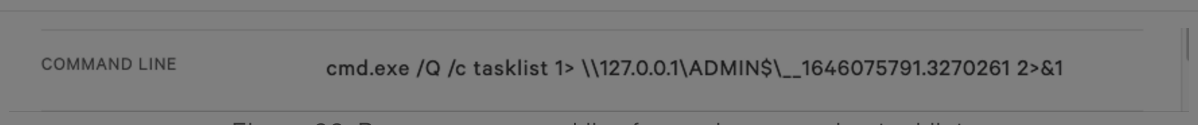


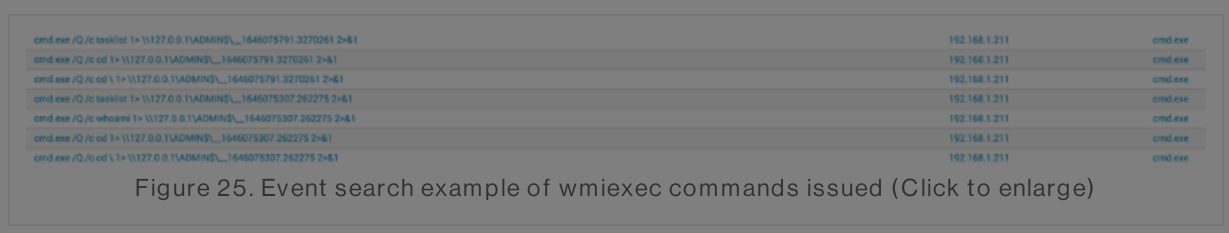
Figure 22. Process command line for wmiexec running tasklist

The prevention will restrict the threat actor's ability to issue remote commands to systems that have an active CrowdStrike Falcon® endpoint protection agent. As shown in Figure 23, the threat actor will receive a response of "SMB SessionError." The failure to execute wmiexec is true for the semi-interactive shell (Figure 23), as well as running single commands (Figure 24).

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. Cookie Notice

identify threat actor activity. Figure 25 shows an example output of an event search in the CrowdStrike Falcon® console to identify all commands issued. By looking at this list, defenders can see the process, `cmd.exe`; the remote host that executed the command, `192.168.1.211` (threat actor Testing System); and the full process command line. This is all captured in real time and enables defenders to investigate and respond quickly. The process command line should look familiar and shows execution of a handful of commands (`cd`, `whoami` and `tasklist`).



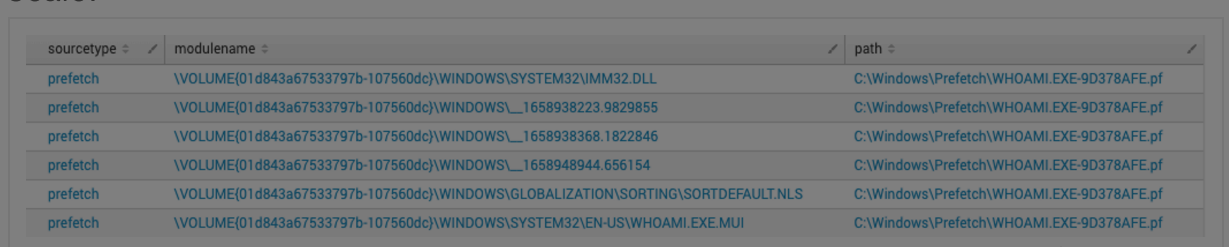
cmd.exe /Q /c tasklist 1> \\127.0.0.1\ADMIN$__1646075791.3270261 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$__1646075791.3270261 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$__1646075791.3270261 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c tasklist 1> \\127.0.0.1\ADMIN$__1646075307.262275 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$__1646075307.262275 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$__1646075307.262275 2>&1	192.168.1.211	cmd.exe
cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$__1646075307.262275 2>&1	192.168.1.211	cmd.exe

Figure 25. Event search example of wmiexec commands issued (Click to enlarge)

Falcon Forensics

When looking historically at the artifacts discussed in this blog, Falcon Forensics can be a great asset to a defender's investigation. Falcon Forensics is a run-once script that is easily deployed through RTR or other deployment tools that will capture a variety of forensic artifacts used for forensic triage of a system. The most beneficial advantage to using Falcon Forensics is the variety of sourcetypes and triage data matched with the ability to perform investigations at scale across an entire environment. Considering the artifacts discussed in this blog, there are specific examples that can be used to find evidence of wmiexec execution at scale, and historically.

Three source types to scratch the surface and highlight when hunting for wmiexec are prefetch, pslist and dirlist. The standard Falcon Forensics executable will not only capture the presence of Prefetch artifacts but also parse the artifact to identify related modules used by the process, similar to parsing Prefetch discussed previously. In Figure 26 below, the `whoami.exe` Prefetch file contains a module name of the temporary wmiexec artifacts. Using Falcon Forensics across your environment can look for these artifacts at scale.



sourcetype	modulename	path
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\IMM32.DLL	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS__1658938223.9829855	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS__1658938368.1822846	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS__1658948944.656154	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS\GLOBALIZATION\SORTING\SORTDEFAULT.NLS	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf
prefetch	\VOLUME{01d843a67533797b-107560dc}\WINDOWS\SYSTEM32\EN-US\WHOAMI.EXE.MUI	C:\Windows\Prefetch\WHOAMI.EXE-9D378AFE.pf

Figure 26. Falcon Forensics event search example of wmiexec artifacts in Prefetch (Click to enlarge)

In addition to the Prefetch sourcetype, pslist can be used to identify the previously discussed command line artifacts associated with wmiexec that may be still running at the time of Falcon Forensics deployment. Shown in Figure 27, since PowerShell and nslookup will both hang on execution, the processes were still running, which allowed for visibility into the process execution.

sourcetype	cmdline	name

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**

dirlist	_1658948979.8670597	C:\Windows\
dirlist	_1658948959.0443158	C:\Windows\

Figure 28. Falcon Forensics event search example of wmiexec artifacts in dirlist (Click to enlarge)

Conclusion

Impacket, and specifically wmiexec, is a tool increasingly leveraged by threat actors. While defenders should remain vigilant on the usage of Impacket, the strategies discussed in this blog can also be used to dissect and understand other threat actor tool sets to identify avenues for detection and prevention.

Additional Resources

- *Learn more about hands-on-keyboard threats and the power of human-led threat hunting at Fal.Con 2022, the cybersecurity industry's most anticipated annual event. Register now and meet us in Las Vegas, Sept. 19-21!*

- *Read about adversaries tracked by CrowdStrike in 2021 in the 2022 CrowdStrike Global Threat Report.*
- *Learn more about the CrowdStrike Falcon® platform by visiting the product webpage.*
- *Test CrowdStrike next-gen AV for yourself. Start your free trial of Falcon Prevent™ today.*

Tweet Share

Related Content

CrowdStrike Named a Leader with "Bold Vision" in 2024 Forrester Wave for Cybersecurity

How to Defend Employees and Data as Social Engineering Evolves

The Anatomy of an ALPHA SPIDER Ransomware Attack

ABOUT COOKIES ON THIS SITE

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. Cookie Notice

Copyright © 2024 CrowdStrike | Privacy | Request Info | Blog | Contact Us | 1.888.512.8906 | Accessibility

ABOUT COOKIES ON THIS SITE

By clicking “Accept All Cookies”, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. **Cookie Notice**