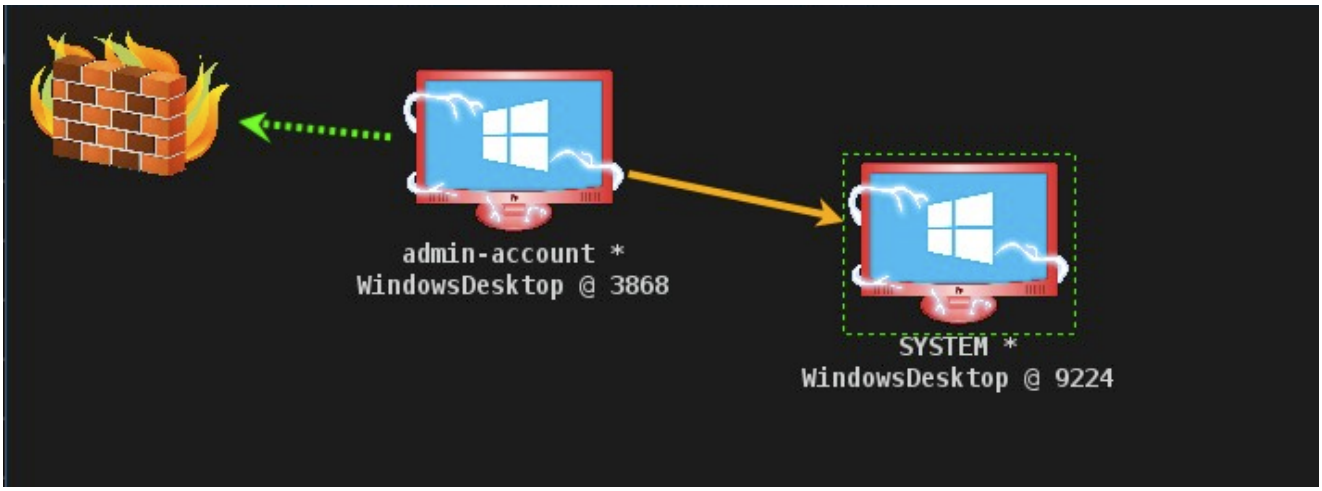
Hunting and detecting Cobalt Strike

sekoia.io/en/hunting-and-detecting-cobalt-strike/

March 24, 2021



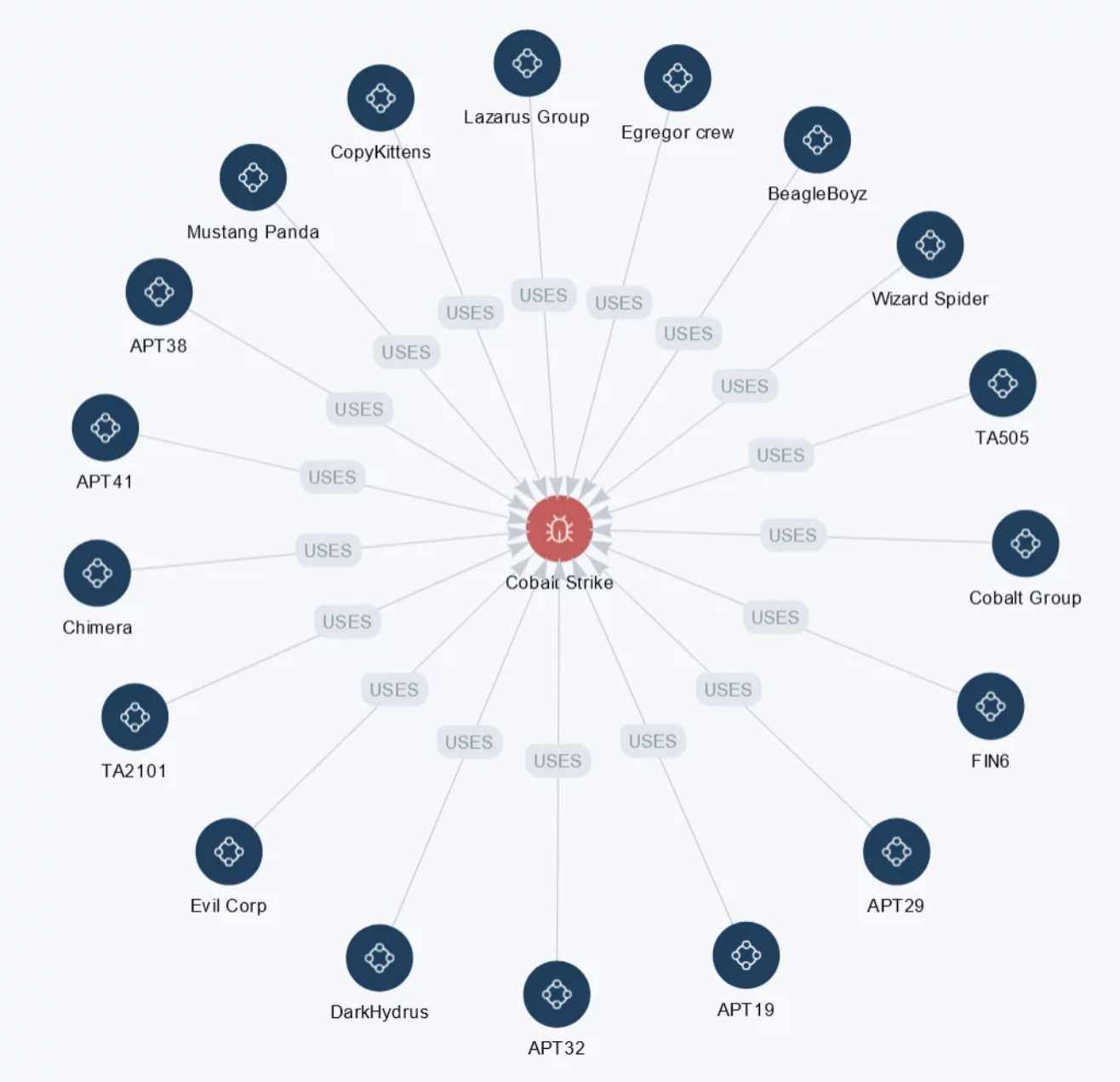
In the last SEKOIA.IO Threat & Detection Lab we dealt with a Man-in-the-middle (MITM) phishing attack leveraging Evilginx2, an offensive tool allowing two-factor authentication bypass. Here, we are tackling a much bigger threat given the frequency it is abused by diverse threat actors. In this blogpost, we describe step by step how to ensure a **proactive and defensive posture** against Cobalt Strike, one of the most powerful pentesting tools hijacked by attackers in their numerous campaigns.

We show examples of how to track Cobalt Strike command and control servers (C2) and Malleable profiles by focusing on their SSL certificates and HTTP responses.

We also describe ways to detect: (i) Cobalt Strike payloads such as the DNS beacon based on the nature and volume of Cobalt Strike DNS requests, (ii) Cobalt Strike privilege escalation with the Cobalt Strike built-in service svc-exe, (iii) Cobalt Strike lateral movement with the Cobalt Strike built-in service PsExec and (iv) Cobalt Strike beacons communication through named pipes.

Why should defenders focus on Cobalt Strike hunting and detection ?

What do APT29, APT32, APT 41, APT19, UNC2452, FIN6, Wizard Spider and most of the cybercriminals have in common in their toolset?



Well, as shown on the figure above, the answer is Cobalt Strike.

Cobalt Strike is a commercial, post-exploitation agent, designed to allow pentesters to execute attacks and emulate post-exploitation actions of advanced threat actors. It aims at mimicking threat actors' tactics, techniques and procedures to test the defenses of the target. However, over the last years, it's purposes were hijacked by attackers who managed to crack its official versions and leverage them in their attacks thus taking advantage of Cobalt Strike's remote access and defense evasion capabilities.

Cobalt Strike is now widely being used by threat actors regardless of their capabilities, skill sets, the sophistication of their attacks or the objectives of their campaigns. To mention just a few examples, it has been leveraged in the recent advanced and state-sponsored

SolarWinds supply chain attacks [1], as well as in the frequent and offensive campaigns conducted by different cybercriminals groups such as Wizard Spider [2], [3] and the Egregor group [4] ultimately delivering ransomware payloads.

In 2020, it was seen as one the most leveraged pentesting tools by attackers, alongside Mimikatz and PowerShell Empire [5]. **Overall, in Q4 of 2020, 66% of all ransomware attacks involved Cobalt Strike payloads [6]**.

Therefore, all these data highlight our need as a defender to be aware and up to date regarding the threat posed by the use of Cobalt Strike for malicious purposes.

In a few words, how does Cobalt Strike work?

Cobalt Strike works in a client/server mode. The server is known as the Team Server, it runs on a Linux system, controls the beacon payload and receives all information from the infected hosts. The client software (known as the Aggressor) runs on multiple operating systems and enables the user to connect to different Team Servers in order to configure the beacon, deliver the payload and fully use all of Cobalt Strike's features remotely.

Beacon is the Cobalt Strike payload, highly configurable through the so-called "Malleable C2 profiles" allowing it to communicate with its server through HTTP, HTTPS or DNS. It works in asynchronous or interactive mode, and can build stageless or staged payload, offering overall considerable flexibility.

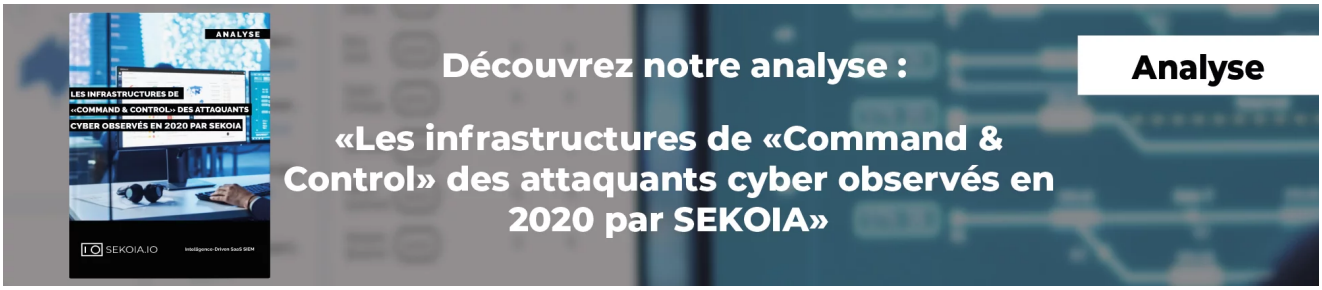
Once connected to its C2 server, the user configures a "listener" (HTTP, DNS …) and a stageless or staged beacon (Windows PE, PowerShell …). The beacon delivery can be directly achieved from the Cobalt Strike server or through another user tool.

This tool is straightforward to use and very well documented [7] which explains its increasing popularity.

To adopt a proactive posture and protect our customers from attacks leveraging Cobalt Strike, we have focused on both tracking Cobalt Strike servers and implementing up-to-date rules capable of detecting each version of Cobalt Strike.

Attacks performed with leaked versions of Cobalt Strike are generally carried out with old versions depending on how easy it is to find these leaks. For this lab session we chose to use the version 4.2 (released the 06/11/2020), which has been leaked on hacker forums and was easy to stumble upon.

The latest 4.3 version was just released (03/03/2021). Aside from the usual new features and bug fixes for each release, we have witnessed some efforts to fix the most specific technical details that help detect Cobalt Strike. We discuss some of them in this article, but it is undoubtedly a never ending game.



Télécharger

This is how we hunt for Cobalt Strike C2 servers

We currently possess more than 50 trackers for Cobalt Strike C2 servers and Malleable profiles, which enabled us to feed, with high confidence, our Intelligence database with more than 10.000 IPs in 2020, that detected Cobalt Strike intrusions. To know more about our hunting results, you can read our analysis following this link.

You will find below an example of three features you can track to spot Cobalt Strike servers. Several trackers are valid for old versions of Cobalt Strike. But as you will notice when considering the number of servers we still detect by dint of these trackers, they are still effective. As said previously, threat actors usually use leaked versions which are not necessarily the most recent ones.

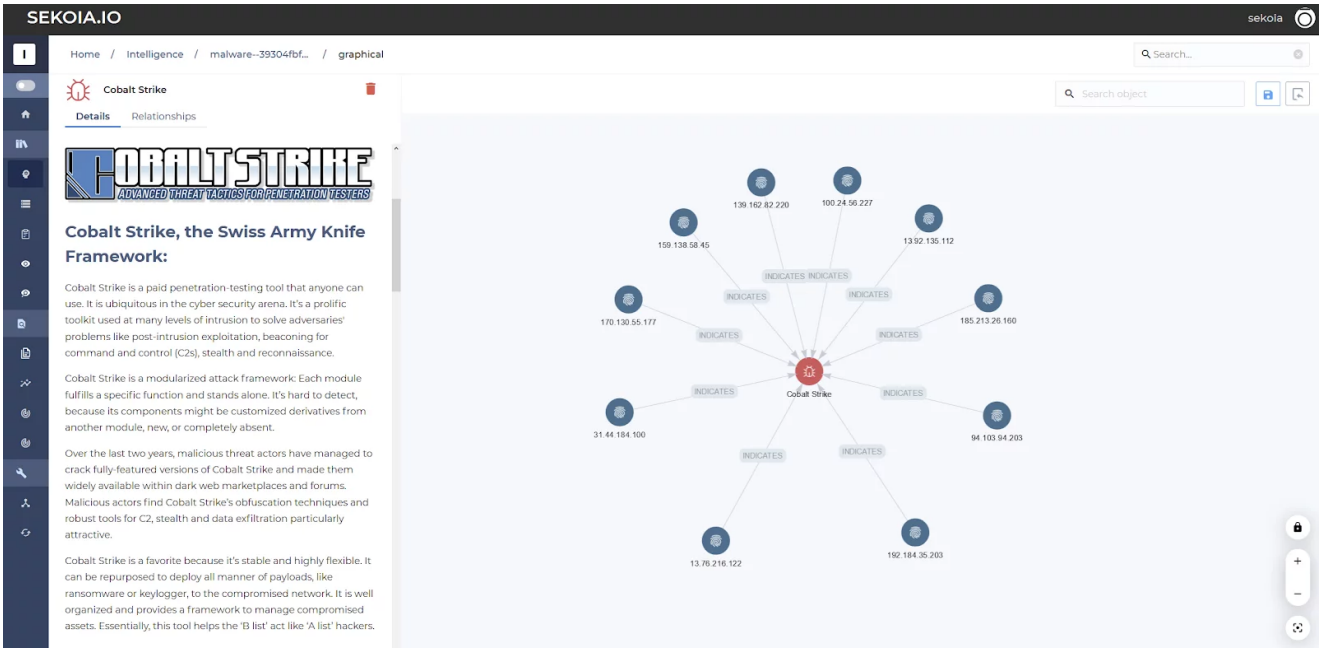
Keep a close eye on default certificates

Cobalt Strike servers come with a default certificate displaying specific values for the serial number, the issuer, the subject and the certificate validity as shown below.

SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 146473198 (0x8bb00ee)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=, ST=, L=, O=, OU=, CN=
        Validity
            Not Before: May 20 18:26:24 2015 GMT
            Not After : May 17 18:26:24 2025 GMT
        Subject: C=, ST=, L=, O=, OU=, CN=
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
```

If they have not been modified by the attackers, these servers can be easily spotted using a shodan request. At the time of writing, 700 servers match this certificate serial number.



What does the HTTP response tell us?

For Cobalt Strike versions prior to version 3.13, the http response displays an extraneous space at the end of the http status.

data

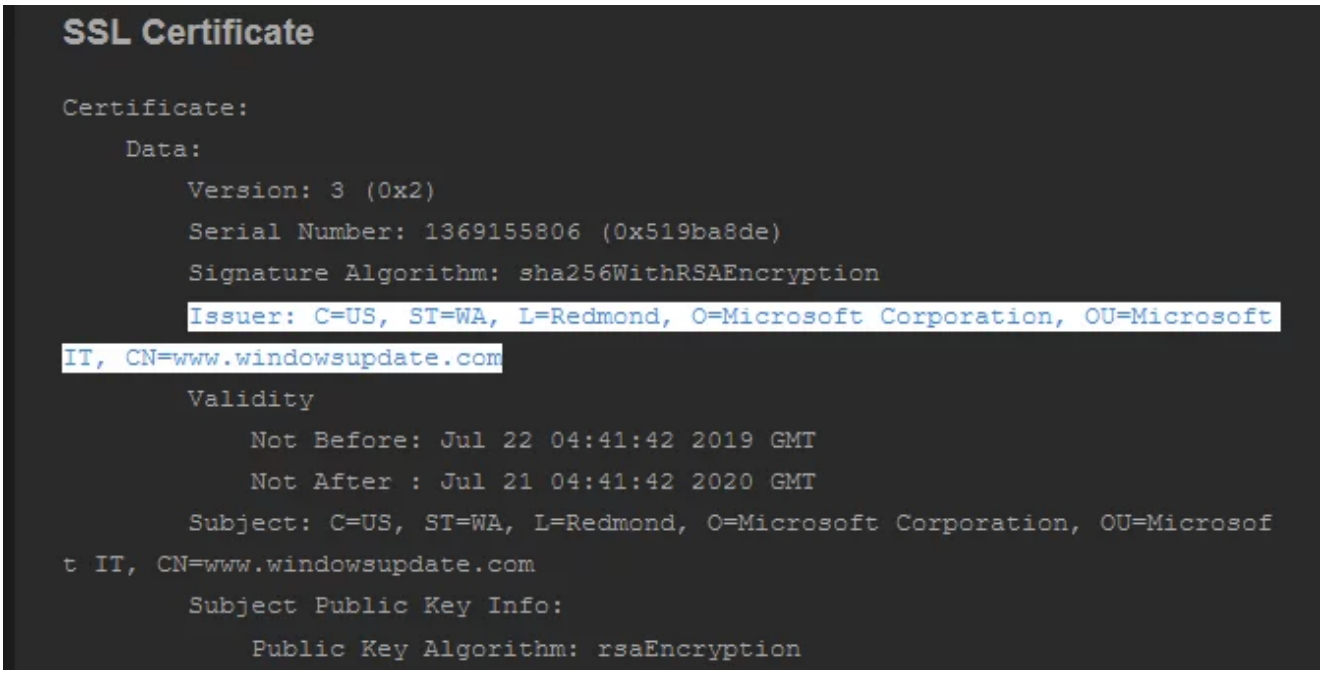
```
HTTP/1.1 404 Not Found
Date: Thu, 4 Mar 2021 03:59:23 GMT
Content-Type: text/plain
Content-Length: 0
```

At the time of writing, we can still catch 30 servers using this tracker. The same detection could be done using Snort rules.

Let's check a malleable C2 profile

As mentioned earlier, Malleable C2 profiles allow to customize Cobalt Strike, which also means that some public configuration could be used to track C2 servers.

Here is an example of a malleable C2 profile, with a self-signed certificate [8] that we can hunt using shodan.



The certificate issuer information (common name, organization, organization unit, location, and country) matched with 116 servers online in 2020.

How can we detect Cobalt Strike with our SIEM?

We performed some attacks using Cobalt Strike beacons in laboratory conditions, so we could figure out some ways to detect it with our SIEM SEKOIA.IO.

For this blogpost, we chose to focus on an attack that was carried out using a DNS beacon as a first stage listener and the SMB beacon for lateral movement. We then managed to detect each step using either Cobalt Strike leaked source code or the generated logs.

To detect it using the following rules you will need to have access these events logs:

- Microsoft-Windows-Sysmon/Operational (and the relevant symon config, especially for Named Pipe)
- Microsoft-Windows-Windows Defender/Operational (or any other AntiVirus logs)
- DNS and Proxy logs

Here is an explanation of rules that can be implemented into your SIEM to specifically detect this attack.

DNS beaconing is a very useful feature, which allows to bypass any other HTTP filtering or proxy inspection that may exist in the targeted company.

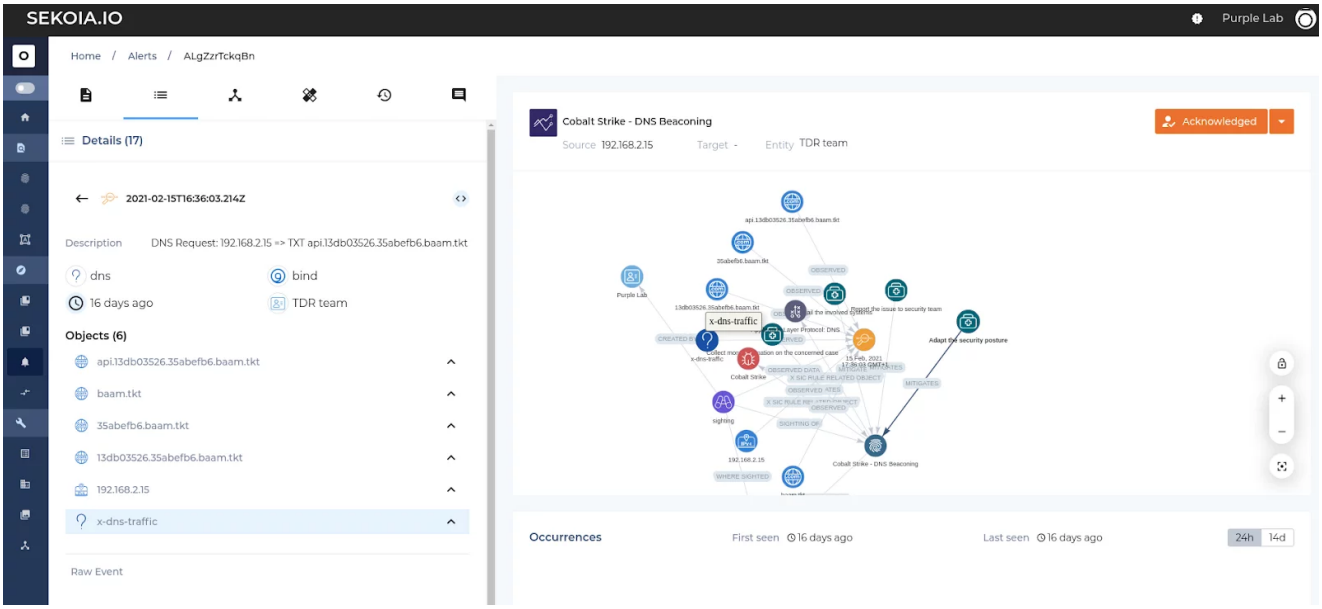
Last year the Cobalt Strike source code (4.0 version) was leaked, and a security researcher quickly spotted some interesting characteristics for DNS beaconing:

> From the CobaltStrike source code, you can now build nice query rules for your local Passive DNS database or NIDS.
>
> https://t.co/EzLrBljp8b pic.twitter.com/wqOTew28u1
>
> — Alexandre Dulaunoy (@adulau) November 12, 2020

The mentioned source code reveals that Cobalt Strike is using three constant DNS labels in pair with DNS question type: “cdn” for A type, “api” for TXT type and “www6” for AAAA type. That means that at some point when the beacon will try to reach its C2 server, aside from two random labels and the one chosen by the user, that constant string will be used: it is very convenient in terms of detection and enables us to build this kind of rule:

```
(dnsquery.value LIKE ‘www6.%’ AND dnsquery.type = ‘AAAA’) OR (dnsquery.value LIKE ‘cdn.%’ AND dnsquery.type = ‘A’) OR (dnsquery.value LIKE ‘api.%’ AND dnsquery.type = ‘TXT’)
```

Using this rule we were able to detect the first stage of our attack that leveraged the DNS beacon as shown below.



The rule needs some exceptions (*e.g.* cdn.onenote.net, cdn.fwupd.org) in order to avoid possible false positives in your DNS traffic, but seems reliable. Latest Cobalt Strike 4.3 version brings new options to override the default values through a Malleable C2 profile [9]. With the previous versions attackers will need to modify the source code or patch their beacon binaries.

Beaconing network traffic

Another behavior that can be detected relies on the interval value between two beacon network requests, which has no “sleep” time by default. This configuration could be modified with a Malleable C2 profile.

Therefore, working with Cobalt Strike in interactive mode will generate a considerable amount of network requests especially with some beacon (*e.g.* DNS) when it comes to downloading/uploading files. That could be leveraged for detection using classical behavior rules.

During our attack, we observed that for both DNS and HTTP beacons, even with only the beacon activity (no exfiltration or command), the total DNS/HTTP requests from the infected host exceeded 200 requests by minutes.

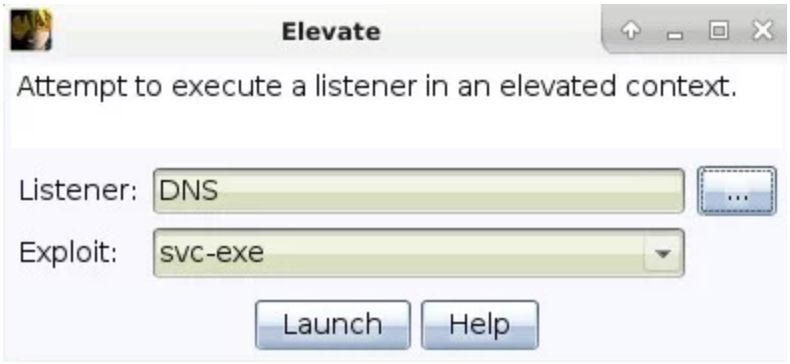
The rule could take this form:

```
selection DNS/HTTP requests | count() by minute src_ip > 200
```

Detects when an attacker elevate its privileges using svc-exe and move laterally using PsExec

We managed to elevate our privileges within the victim system. We chose to achieve this using svc-exe, which is a built-in Cobalt Strike exploit.

It will drop an executable that runs a payload, create a service to run it, assume control of the payload, and cleanup the service and executable. Thus allowing us to get SYSTEM.



Then, we wanted to perform a lateral movement and jump to the new targeted host. We chose to do so leveraging an SMB beacon which is a good candidate frequently leveraged in attacks.

We ran a command on Cobalt Strike that leverages psexec64 as follows:

```
$ jump psexec64 <host-ip> <name-of-our-SMB-beacon-listener>
```

This is one of the features that make Cobalt Strike a strong and efficient tool. It relies on native Windows APIs and not a third-party protocol stack, thus increasing its defense evasion capabilities. Hence, Cobalt Strike has a built-in PsExec, which strength lies in its ability to launch interactive command-prompts on remote systems. It can be run on the victim's system since it uses native Windows components.

action.id	3
action.name	Network connection
action.properties.AccountType	User
action.properties.OpcodeValue	0
action.properties.ProviderGuid	{5770385F-C22A-43E0-BF4C-06F5698FFBD9}
action.properties.Severity	INFO
action.properties.SourceName	Microsoft-Windows-Sysmon
action.properties.Task	3
action.record_id	261073
action.type	Microsoft-Windows-Sysmon/Operational
alert_short_ids	ALXnn2umGnN7
customer.intake_key	88DDYh2CMYYJyhAPtrojRe9GxR9yu9rA
customer.intake_name	WindowsDesktop-Azure
customer.intake_uuid	c2459bbe-4011-4608-9a30-cb481a23e45a
destination.domain	WindowsDesktop.1gz3hij5drhurexzedhrvofwad.parx.internal.cloudapp.net
destination.ip	10.0.3.7
destination.port	135
entity.name	TDR team
entity.uuid	4ba46c44-7b3f-4250-8ae0-4fe85af037c0
event.code	3
event.dialect	windows
event.dialect_uuid	9281438c-f7c3-4001-9bcc-45fd108ba1be
event.id	3b52a165-ed00-457d-a9e6-f6406e9098d3
event.outcome	success
event.provider	Microsoft-Windows-Sysmon
log.hostname	WindowsDesktop
network.transport	tcp
os.family	windows
os.platform	windows
process.executable	c:\windows\system32\rundll32.exe
process.name	rundll32.exe
process.pid	2444
process.thread.id	3236
process.working_directory	c:\windows\system32\

We observed that these operations resulted in a service creation. They both spawned rundll32.exe that initiated a network connection, which is the event ID 3 of the Microsoft-Windows-Sysmon/Operational journal. What makes it different from the usual behavior of rundll32, is that the dll is run without any command line arguments.

This behavior is anchored in Cobalt Strike. Unless attackers wisely decide to change this configuration in the source code, this surely will spot many other attacks.

Pipes to detect them all

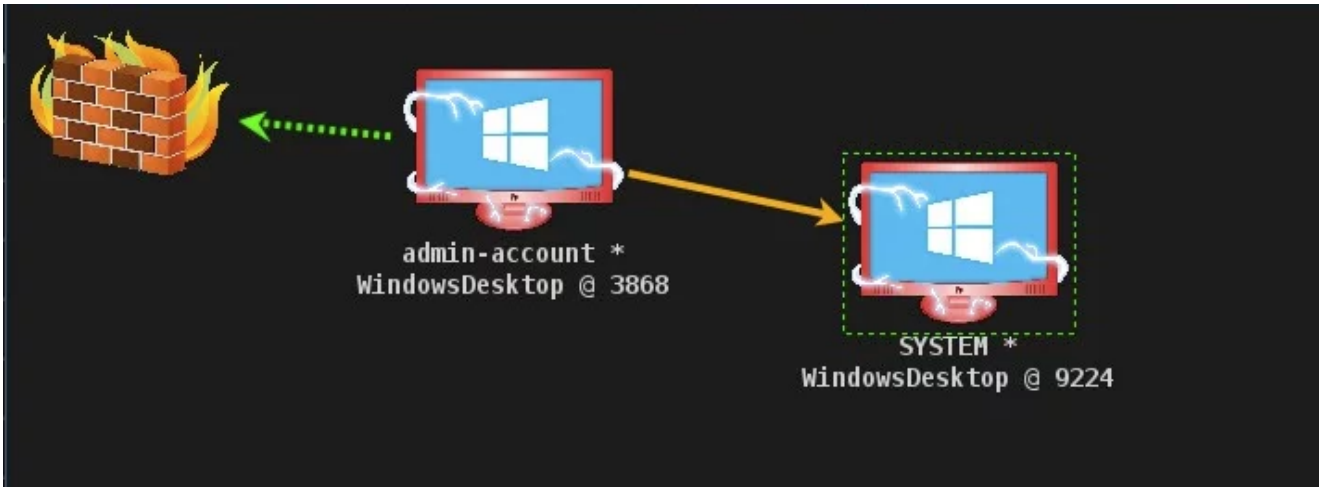
A pipe is a section of shared memory that processes use for communication. The process that creates a pipe is the pipe server. The one that connects to a pipe, is the pipe client. A process writes information to the pipe, while the other process reads the information from the pipe.

There are two types of pipes: named and anonymous pipes.

Named pipes are one-way or duplex pipes that are used for network interprocess communication that can take place between a pipe server and one or more pipe clients. Multiple pipe clients can use the same named pipe simultaneously in the same instance.

Anonymous pipes are unnamed, one-way pipes that are used for interprocess communications between a parent and a child process, only on a local computer.

Cobalt Strike has the ability to pivot over named pipes. It uses pipes to allow a beacon to receive its commands and send its ones to another beacon. In this situation, both beacons will communicate over pipe channels as highlighted by the orange arrow in the pivot graph shown below. Cobalt Strike also uses TCP sockets and SSH sessions to connect a beacon session to another.



Hence, when we connected a listener (*e.g.* DNS beacon) with another beacon (*e.g.* SMB beacon) to perform lateral movement, we observed the creation of the sysmon event ID 17 “Pipe created”, in our logs.

```
{
    "EventTime": "2021-03-03 09:35:37",
    "Hostname": "WindowsDesktop",
    "Keywords": -9223372036854776000,
    "EventType": "INFO",
    "SeverityValue": 2,
    "Severity": "INFO",
    "EventID": 17,
    "SourceName": "Microsoft-Windows-Sysmon",
    "ProviderGuid": "{5770385F-C22A-43E0-BF4C-06F5698FFBD9}",
    "Version": 1,
    "Task": 17,
    "OpcodeValue": 0,
    "RecordNumber": 243359,
    "ProcessID": 2444,
    "ThreadID": 3668,
    "Channel": "Microsoft-Windows-Sysmon/Operational",
    "Domain": "NT AUTHORITY",
    "AccountName": "SYSTEM",
    "UserID": "S-1-5-18",
    "AccountType": "User",
    "Message": "Pipe Created:\r\nRuleName: -\r\nEventType: CreatePipe\r\nUtcTime: 2021-03-03
09:35:37.827\r\nProcessGuid: {2d950e13-5869-603f-9d01-000000000d00}\r\nProcessId: 7792\r\nPipeName:
\\MSSE-6013-server\r\nImage: \\\\127.0.0.1\\ADMIN$\\9203afd.exe",
    "Category": "Pipe Created (rule: PipeEvent)",
    "Opcode": "Info",
    "RuleName": "-",
    "UtcTime": "2021-03-03 09:35:37.827",
    "ProcessGuid": "{2d950e13-5869-603f-9d01-000000000d00}",
    "PipeName": "\\MSSE-6013-server",
    "Image": "\\\\127.0.0.1\\ADMIN$\\9203afd.exe",
    "EventReceivedTime": "2021-03-03 09:35:38",
    "SourceModuleName": "eventlog4",
    "SourceModuleType": "im_msvistalog"
}
```


During our various tests, we observed that the created pipe displayed the same pattern that can be detected by this regex:

```
MSSE-[0-9]{4}-server
```

Cobalt Strike users cannot change the default value of these pipes without accessing and modifying the source code configuration of Cobalt Strike.

It is important to distinguish the pipes that are created to allow beacons to communicate, from the named pipes that are generated specifically for the SMB beacon, and which default value is in the form of: *msagent_39* as shown below. Unlike the MSSE pipes, the default value of the pipe name of the SMB beacon can be easily modified on the attack interface.

Default payload

For a lot of user interactions, Cobalt Strike displays a default value, and especially for the payload naming. Of course it is a very weak detection indicator, but a mistake is always possible and defenders only need one.

During our lab tests with different use cases, we had theses default binaries names:

```
beacon.{bin|exe|dll|ps1}, artifact.{dll|exe}, payload.{java|ps1|py|rb|vba}
```

Anti-virus logs at that point could also be an easy win: most AVs have specific signatures that could be used to trigger an alert in your SIEM, and detect an attacker who forgot to use its custom payload.

More globally, binaries' other characteristics (on disk or in memory), could also be used as detection indicators. That is why Cobalt Strike's editor advises to customize it with a Malleable C2 profile or the Artifact Kit [10].

Many public yara rules exist in order to precisely do that, and try to follow existing payload available in the wild [11]. This is recommended if you have an EDR capability.

How to mitigate Cobalt Strike?

As said before, most of Cobalt Strike beacons characteristics can be customized. Either directly on the user interface for some of them, through a malleable profile or by directly reversing the source code. Hence, it is essential to be as exhaustive as possible regarding the detection capacity.

Furthermore, it seems that Cobalt Strike designer made it one of its priority to always ensure that its tool can not be detected, releasing a new version each time the last version was well documented by defenders. Highlighting for us the need to be up to date regarding the new versions characteristics. Since, attackers mostly used leaked versions, we are still a step ahead in detecting the latest threats.

Given the volumes of attacks performed with Cobalt Strike, combining both C2 server hunting and beacon detection as shown in this article, is definitely a good way to ensure the best protection and tighten the net around that threat.

To be protected against it, we highly recommend you to rely on a real-time detection solution fuelled by cyber threat intelligence.

Besides having an up-to-date SIEM, there are also these evident course of actions that defenders could leverage:

- Cobalt Strike can be dropped in victims systems following phishing campaigns leveraging VBS scripts. It is recommended to disable document macro in MS office. Training users to notice malicious emails should also be performed on a regular basis.
- Cobalt Strike payload can be delivered as a powershell script. It is recommended to restrict powershell script execution to allow signed scripts only.
- Some Cobalt Strike payload signatures can be identified by antivirus. It is recommended to have a good antivirus product.
- Cobalt strike beacons generate abnormal behaviors that can be hunted using Sysmon, Security, PowerShell and WMI logs.
- It is recommended to hunt for parent processes spawning unexpected child processes.
- Monitor suspicious modifications to registry keys, startup folders, task scheduler and service execution.

If you have been infected by Cobalt Strike, it is recommended to carry out memory forensics. The tool CobaltStrikeScan available on github scan for files and process memory for Cobalt Strike beacons and parse their configuration [12]. It scans Windows process memory for evidence of DLL injection.