Detecting Exposed Cobalt Strike DNS Redirectors

labs.f-secure.com/blog/detecting-exposed-cobalt-strike-dns-redirectors

Intro

Cobalt Strike is a well known framework used to perform adversary simulation exercises by offensive security professionals. Its flexibility and broad feature set have made it the de facto framework for red team operations.

Cobalt Strike's implant, known as "beacon", has the ability to communicate back to a Command & Control (C2) server using different protocols:

- **HTTP/HTTPs** - where the data exchanged between the implant and the C2 server is packaged inside an HTTP request.
- **SMB** - used mostly for peer-to-peer communication, this channel uses the SMB protocol to blend in with the regular traffic you might see on an internal Windows-based network.
- **TCP** - with a similar purpose to the SMB channel, but using plain TCP sockets. Useful for lateral movement in situations where SMB is restricted or heavily monitored. Can also be used for privilege escalation purposes.
- **DNS** - using a variety of DNS queries, Cobalt Strike's beacons can communicate back to the C2 server using only DNS. The advantage is that name resolution is almost always allowed and no direct communication takes place between the implant and the C2 server, since the DNS resolution will happen using the default nameservers.
- **Custom** - using the well-documented External C2 protocol it is possible to develop custom communication mechanisms that will allow an operator to rely on Cobalt Strike's post exploitation capabilities but not being bounded by the existing channels. A popular example of this is F-Secure's C3 framework.

This research will focus on some of the active detections that can be used to fingerprint exposed Cobalt Strike servers that are using DNS as a communication channel. Although the research approach will be a bit different, the outcome will be similar to what JARM did for HTTP/HTTPs restricted to the scope of Cobalt Strike.

DNS Redirectors

Before jumping in the actual detection logic, it is important to understand how the standard DNS server within Cobalt Strike is deployed for an operation. The concept of a "redirector" is something that was introduced to further harden the attack infrastructure and decouple what is exposed by the attacked component and the actual C2 servers. This grants the operator greater flexibility in case any of their internet-facing endpoints get "burned" during the attack; in fact, spinning up a new redirector can take hours if not minutes whilst rebuilding a new C2 server might have a greater impact on the operations.

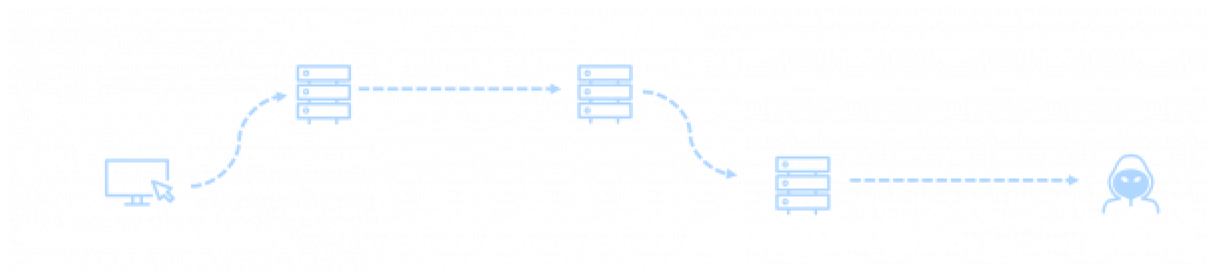
Typically, the standard Cobalt Strike DNS redirector is created using either socat or iptables. The official documentation, in fact, suggests those as the go-to tools for creating DNS based redirectors.

As an example, the following commands can be used to create a simple redirector for DNS:

```
# socat will listen on TCP 5353 and redirect to cobalt strike's DNS server
socat tcp4-listen:5353,reuseaddr,fork UDP:127.0.0.1:53



# port 5353 will be exposed via an SSH tunnel on the external redirector
ssh ubuntu@redir.c2 -R 5353:127.0.0.1:5353

# on the redirector, socat will listen on 53 and forward the data to the SSH tunnel, 
that eventually will reach the C2 server
socat udp4-listen:53,reuseaddr,fork tcp:localhost:53535
```

The creation of a DNS listener within Cobalt Strike will not be covered as it is outside the scope of this research. For more details it is recommended to check out Steve Borosh's

The resulting communication flow will look like the one schematised below:



The setup can be validated by querying the hostname that was initially configured for DNS C2. If the setup is working properly, the DNS response should be the one configured in the dns_idle malleable profile option, and by default it's equal to "0.0.0.0":

```
nslookup malware.c2

Non-authoritative answer:
Name:    malware.c2
Address: 0.0.0.0
```

The "0.0.0.0" reply, as previously said, is the Cobalt Strike's default and as every default value, should be changed.

The Detection

The research that F-Secure conducted is based on the following statement:

> Cobalt Strike's DNS listeners will reply using the value defined in the dns_idle field regardless of the query received, as long as it is not part of a C2 communication

In fact, the dns_idle field is used by the beacon as a heartbeat to check in for new tasks. The "problem" is that the default DNS server will reply using that value to all the other queries for other domains as well.

The hypothesis is that if a DNS server replies to all the queries with always the same IP address, it might be an indicator of the presence of Cobalt Strike. Of course this approach is not free of false-positives, and part of the research was to quantify the fidelity of this mechanism,

It was possible to validate our hypothesis using a live Cobalt Strike server:

```
dig @70.35.206.199 +short amazon.com
0.0.0.0



dig @70.35.206.199 +short google.com
0.0.0.0



dig @70.35.206.199 +short nonexistent.domain
0.0.0.0

dig @70.35.206.199 +short -t TXT google.com
0.0.0.0
```

As it is possible to see, the server replied to all the queries with "0.0.0.0", the default dns_idle value. This approach would have worked even if the default value was changed, since all the queries would still return the same value. The interesting aspect was that even for queries different than "A", Cobalt Strike still returned an "A" record. This broke the parsing logic of most of the common libraries for name resolution. The snippet below shows an example of that:

```
dig TXT test @35.178.76.239



; <<>> DiG 9.10.6 <<>> TXT test @35.178.76.239
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45988
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available



;; QUESTION SECTION:
;test.                IN    TXT



;; ANSWER SECTION:
test.            1    IN    A    8.8.8.8

;; Query time: 97 msec
;; SERVER: 35.178.76.239#53(35.178.76.239)
;; WHEN: Tue Mar 23 10:11:00 CET 2021
;; MSG SIZE  rcvd: 42
```

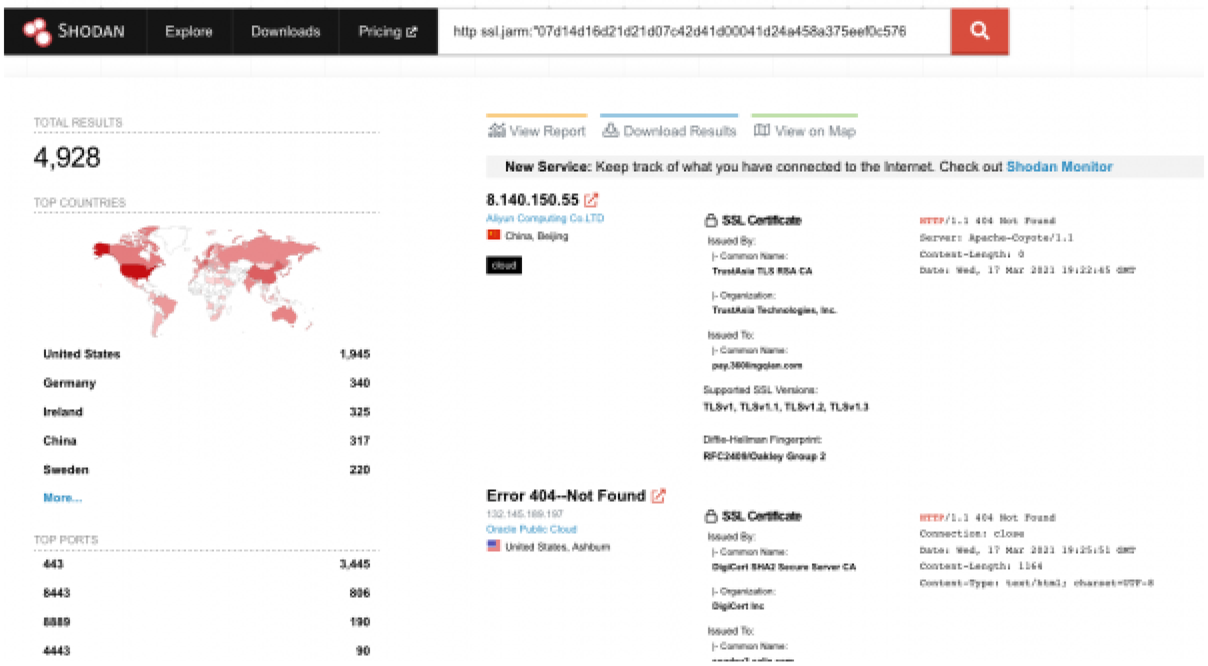
In order to avoid higher level parsing logic, the Scapy library was used to forge raw DNS packets and build an automated scanner.

The snippet below shows the core function where the packers are sent and the responses are compared:

```
def checkDNS(name):
    ip = name
    try:
        ans = sr1(IP(dst=ip)/UDP(sport=RandShort(),
dport=53)/DNS(rd=1,qd=DNSQR(qname="google.com",qtype="TXT")), verbose=False,
timeout=0.5)
        ansTXT = ans[DNSRR][0].rdata
        ans = sr1(IP(dst=ip)/UDP(sport=RandShort(),
dport=53)/DNS(rd=1,qd=DNSQR(qname="amazon.com",qtype="A")), verbose=False,
timeout=0.5)
        ansA1 = ans[DNSRR][0].rdata
        ans = sr1(IP(dst=ip)/UDP(sport=RandShort(),
dport=53)/DNS(rd=1,qd=DNSQR(qname="google.com",qtype="A")), verbose=False,
timeout=0.5)
        ansA2 = ans[DNSRR][0].rdata
        if ansTXT == ansA1 and ansA1 == ansA2 and ansA2 != '' and valid_ip(ansA1):
            print("[+] Cs Detected: " + ip  + " replied with: " + ansTXT)
    except Exception as e:
        pass
```

The code is quite self-explanatory and simply translates the initial hypothesis into actual Python code. The rest of the script's functionalities only added multi threading and better output formats, but the main logic is the one shown above.

The scanner was tested against a small subset of data, specifically using all the HTTP servers exposed on the internet that had a JARM signature equals to the default Cobalt Strike C2 server. The data was gathered using the beta version of Shodan:



http ssl.jarm:"07d14d16d21d21d07c42d41d00041d24e458a375eef0c576

TOTAL RESULTS
4,928

TOP COUNTRIES

United States	1,945
Germany	340
Ireland	325
China	317
Sweden	220

TOP PORTS

443	3,445
8443	806
8889	190
4443	90

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

8.140.150.55
Aliyun Computing Co.LTD
China, Beijing
cloud

SSL Certificate
Issued By:
|- Common Name:
TrustAsia TLS RSA CA
|- Organization:
TrustAsia Technologies, Inc.
Issued To:
|- Common Name:
pay.360lingqian.com
Supported SSL Versions:
TLSv1, TLSv1.1, TLSv1.2, TLSv1.3
Diffie-Hellman Fingerprint:
RFC2409/Oakley Group 2

```
HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Wed, 17 Mar 2021 19:22:45 GMT
```

Error 404--Not Found
132.145.169.197
Oracle Public Cloud
United States, Ashburn

SSL Certificate
Issued By:
|- Common Name:
DigiCert SHA2 Secure Server CA
|- Organization:
DigiCert Inc
Issued To:
|- Common Name:

```
HTTP/1.1 404 Not Found
Connection: close
Date: Wed, 17 Mar 2021 19:25:51 GMT
Content-Length: 1164
Content-Type: text/html; charset=UTF-8
```

All the resulting data was downloaded and subsequently parsed via the Shodan CLI in order to extract the IP addresses:

```
# add shodan query
shodan parse --fields ip_str e5a2e2e7-e029-4b10-a5f8-63687aa7a09c.json.gz >
cobalt.txt
```

The results against the initial dataset were positive, with multiple matches:

```
python scan.py ~/Downloads/cs-dns.txt
[+] Cs Detected: 24.252.133.75 replied with: 10.0.0.1
[+] Cs Detected: 193.202.92.36 replied with: 193.202.92.36
[+] Cs Detected: 194.204.33.130 replied with: 127.53.53.53
[+] Cs Detected: 194.62.33.31 replied with: 192.168.33.31
[+] Cs Detected: 5.8.16.29 replied with: 72.5.65.111
[+] Cs Detected: 37.191.180.47 replied with: 37.191.180.47
[+] Cs Detected: 35.178.76.239 replied with: 8.8.8.8
[+] Cs Detected: 85.27.174.244 replied with: 85.27.174.244
[+] Cs Detected: 70.35.206.199 replied with: 0.0.0.0
[+] Cs Detected: 209.9.227.212 replied with: 209.9.227.212
[+] Cs Detected: 175.176.185.229 replied with: 103.60.101.170
[+] Cs Detected: 179.191.84.68 replied with: 10.0.0.1
```

Internet Wide Survey

Scanning what is already known to be bad is funny, but only to a certain extent. The next step of the research was to expand the scope of our analysis to the whole internet.

In order to obtain the needed results, it was necessary to perform an internet wide scan of all the hosts that exposed the port 53/UDP. Instead of manually doing the scan, it was decided to use Project Sonar's dataset, which already collected the information needed. This allowed us to reduce the scope of the scan from billions of IPs to something around 7 million records. This was found to be particularly useful considering that the detection used a UDP-based protocol, which in general is slower and not as reliable as TCP.

Running our scanner against the dataset produced approximately 1400 results, a number that we believed to be too high and needed further validation and false positive checks. We started the result sanitisation by filtering out the hosts that presented additional Cobalt Strike specific behaviour, such as:

- Having port 50050 open
- Replying with a 404 status code on HTTP/S and 0 as content length, a distinct sign of the Cobalt Strike's default malleable profile
- Having a matching JARM signature
- Having staging enabled, and it was possible to retrieve the Beacon configuration from an open HTTP port
- Replying with "0.0.0.0" , the default "dns_idle" value

The results of the process were the following:

- 24 IPs also exposed port 50050
- 49 presented the default Cobalt Strike's HTTP response
- 27 Cobalt Strike servers had staging enabled, that allowed F-Secure to retrieve the beacon's config
- 91 servers replied with "0.0.0.0"
- 16 servers had the "07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1" JARM signature, indicative of Cobalt Strike

Amongst the 1400 results, 122 had high probability of being Cobalt Strike servers, due to the presence of one or more of the signatures above. Upon further investigation of the beacon configurations extracted using this extremely useful Nmap script, a recurring pattern using 8.8.8.8 as dns_idle was noticed:

```
{
   "x64":{
      "md5":"26dca1f00735af2e11d856cdbc239a72",
      "sha1":"0f964782e58ac43ab0433b7cbb007295eed1bcd1",
      "time":1616664618025.3,
      "config":{
         "Port":80,
         "Beacon Type":"0 (HTTP)",
         "Polling":50000,
         "Pipe Name":"",
         "User Agent":"Mozilla\\/5.0 (compatible; MSIE 9.0; Windows Phone OS 7.5;
Trident\\/5.0; IEMobile\\/9.0; LG; LG-E906)",
         "Jitter":15,
         "Header 1":"",
         "Method 2":"POST",
         "Spawn To x86":"%windir%\\\\syswow64\\\\rundll32.exe",
         "C2 Server":"[REDACTED],\\/_\\/scs\\/mail-static\\/_\\/js\\/",
         "Method 1":"GET",
         "Spawn To x64":"%windir%\\\\sysnative\\\\rundll32.exe",
         "Max DNS":255,
         "Header 2":"",
         "HTTP Method Path 2":"\\/mail\\/u\\/0\\/",
         "DNS Sleep":0,
         "DNS Idle":"8.8.8.8"
      },
      "sha256":"ce4d2de8d28423bc975b7792b69722e8b8e01c01c723f43e494709062fcdb550"
   },
   "x86":{
      "md5":"d836ddfcb06c1959d002fabd70aff8fd",
      "sha1":"7d8c5d34da8a0ece4fcda322a1c814b285d4b8f8",
      "time":1616664587530.2,
      "config":{
         "Port":80,
         "Beacon Type":"0 (HTTP)",
         "Polling":50000,
         "Pipe Name":"",
         "User Agent":"Mozilla\\/4.0 (compatible; MSIE 8.0; Windows NT 5.1;
Trident\\/4.0; BTRS125526)",
         "Jitter":15,
         "Header 1":"",
         "Method 2":"POST",
         "Spawn To x86":"%windir%\\\\syswow64\\\\rundll32.exe",
         "C2 Server":"[REDACTED],\\/_\\/scs\\/mail-static\\/_\\/js\\/",
         "Method 1":"GET",
         "Spawn To x64":"%windir%\\\\sysnative\\\\rundll32.exe",
         "Max DNS":255,
         "Header 2":"",
         "HTTP Method Path 2":"\\/mail\\/u\\/0\\/",
         "DNS Sleep":0,
         "DNS Idle":"8.8.8.8"
      },
      "sha256":"52029626c23dc2cd60f65e0bfa087021f0ee4d96ed0259a6e03cd7ff2fd471ac"
   }
}
```

It was decided to include that value as another possible indicator, that brought the total number of Cobalt Strike servers to 135. The remaining number of servers were split in the following categories:

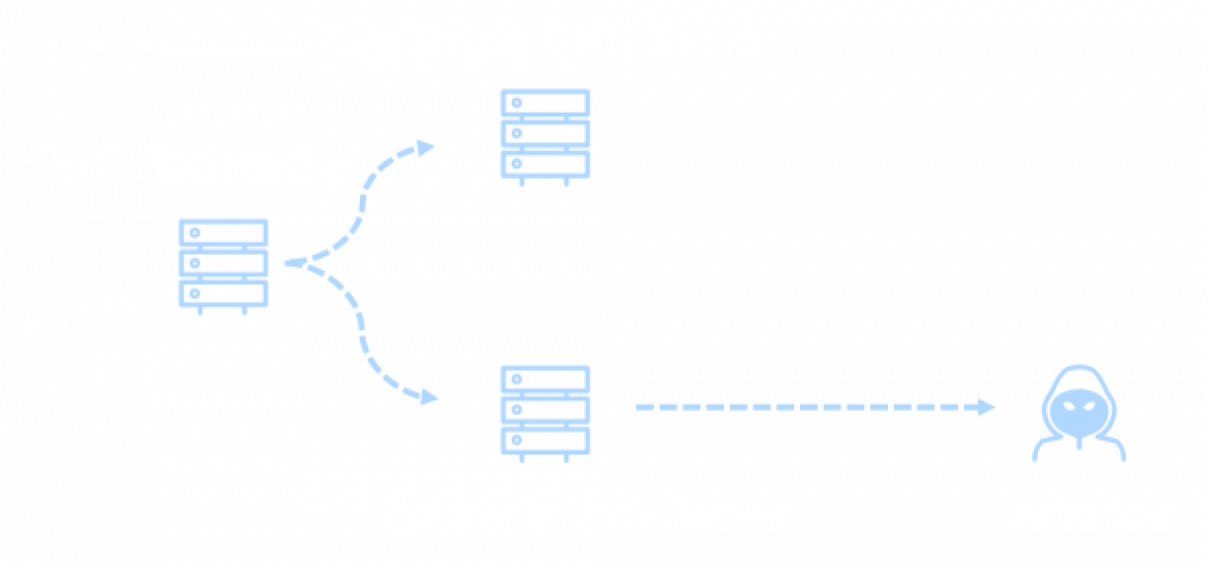
- False positives, we identified that some home router versions acted in a similar way as Cobalt Strike's DNS server. Those could be false positives or actual CS servers.
- Cobalt Strike DNS servers that changed the default "dns_idle" value and did not host an HTTP server on the same IP address.
- DNS sinkholes and honeypots. Despite efforts were made to avoid this, it is possible that some DNS servers acted in a similar way to Cobalt Strike.

The complete scan results will be posted after the release of this blog post, alongside with the source code of the scanner.

Hardening for Red Teamers

The root cause that allowed us to perform this research is that the DNS redirector is "dumb", meaning that it forwards DNS requests to Cobalt Strike without any logic. If, for example, we examine the evolution of HTTP based redirectors we see that using socat or iptables was quickly abandoned in favour of better alternatives such as Apache with mod_rewrite or Nginx.

The problem is that we haven't seen the same approach applied to DNS redirectors. A possible solution would be to create a "smart" redirector capable of proxying DNS requests to specific DNS servers based on the domain name that is being requested:



We found that using open source DNS servers, such as CoreDNS, solved the issue. In fact, using a "Corefile" such as the one shown below, it would be possible to avoid this trivial detection:

```
. {
  forward . 9.9.9.9
}
malware.c2 {
    forward . malware.c2:5353
}
```

CoreDNS can then be deployed on an internet exposed VPS and act as a "smart" redirector. This configuration will proxy to Cobalt Strike only the requests made for the "malware.c2" domain, everything else will be resolved using the "9.9.9.9" public resolver.

Conclusions

The research showed one of the many approaches that can be used to track Cobalt Strike servers exposed on the internet. Whilst not perfect, this can certainly be used to enrich threat intelligence data to achieve better detections.

For red teamers and penetration testers that use either Cobalt Strike or any other C2 framework that supports DNS, we provided an approach that can be used to build better and smarter DNS redirectors using open source tools.

References

Bluescreenofjeff - Red Team Infrastructure Wiki

Steve Borosh - Redirecting Cobalt Strike DNS Beacons

Cobalt Strike - Simple DNS Redirectors for Cobalt Strike

Whiskey-r7 - grab_beacon_config

Salesforce - JARM

CoreDNS