Cobalt Strike Analysis and Tutorial: Identifying Beacon Team Servers in the Wild

By Durgesh Sangvikar, Chris Navarrete, Matthew Tennis, Yanhui Jia, Yu Fu, Siddhart Shibiraj

Published: 2022-11-03 · Archived: 2026-04-05 18:48:49 UTC

Executive Summary

As Cobalt Strike remains a premier post-exploitation tool for malicious actors trying to evade threat detection, new techniques are needed to identify its Team Servers. To this end, we present new techniques that leverage active probing and network fingerprint technology. This is a fundamental change from previous passive traffic detection approaches.

Over the course of our Unit 42 blog series covering the adversary framework tool Cobalt Strike, we document the encoding and encryption techniques of its HTTP transactions. Specifically, we analyzed the advanced, flexible traffic profiles used by Cobalt Strike's Beacon command-and-control (C2) communication to evade detection by defenders.

Beacon implants communicate to an attacker-controlled application called Team Server. Team Server and the Beacon's C2 traffic allow adversaries to easily and effectively cloak malicious traffic as normal, benign traffic. This is made possible by a modular, extensible domain-specific language called Malleable C2.

Sanctioned adversaries (such as Red/Blue team members, pentesters and ethical hackers) as well as malicious actors can use premade or custom Malleable C2 profiles. These profiles can be exceedingly difficult to continually develop traditional defenses against, such as conventional firewall threat prevention.

In previous approaches, Team Server detections could only be made after a Beacon binary was implanted on a victim's system and attempted to "phone home" to an attacker-controlled server. Our new techniques can proactively detect Team Servers in the wild before an active C2 connection from a victim's system has been initialized.

We will also demonstrate the following:

- How Team Servers behave when they receive specially-crafted HTTP requests
- What kind of network fingerprint can be inferred and with what confidence
- Details of some real-life malicious C2 between Beacon and Team Server in the wild

Palo Alto Networks customers receive protections from and mitigations for Cobalt Strike Beacon and Team Server C2 communication in the following ways:

- Next-Generation Firewalls with a Threat Prevention subscription can identify and block Cobalt Strike HTTP C2 requests as well as responses that are masked with the base64 encoding settings of the default profile (signatures 86445 and 86446).

- Next-Generation Firewalls with an Advanced Threat Prevention subscription can identify and block Cobalt Strike HTTP C2 requests generated by custom profiles.
- WildFire and Cortex XDR can identify and block Cobalt Strike Beacon binaries.
- Cortex XSOAR response pack and playbook can automate the mitigation process.
- Cortex XDR will report related exploitation attempts.
- Malicious URLs and IPs have been added to Advanced URL Filtering.
- If you think you may have been compromised or have an urgent matter, the Unit 42 Incident Response team can provide personalized assistance.

Probing and Fingerprint Identification Technology

The Cobalt Strike Team Server, also known as CS Team Server, is the centralized C2 application for a Beacon and its operator(s). It accepts client connections, orchestrates remote commands to Beacon implants, provides UI management, and various other functions.

During our research and development of Advanced Threat Prevention's inline deep learning detection for Cobalt Strike traffic, we began experimenting with forging C2 requests to suspected malicious Team Servers on the Internet. Through our analysis of attacker-controlled server responses, we developed a variety of techniques to classify previously undetected Cobalt Strike Team Servers before an attack can occur.

In the following sections, we share our findings on the following identification techniques:

- Active Probing Over HTTP
- Active Probing Over DNS

Active Probing Over HTTP HTTP/S OPTIONS Request and Response Fingerprint

The Team Server is a Linux program running an HTTP server configured to respond to a variety of HTTP requests. When the server receives requests with the HTTP OPTIONS method, the server will return HTTP status code 200 and Content-Length: 0.

Figure 1 shows an HTTP request and response to a Team Server. The URI provided in an HTTP OPTIONS request is disregarded as the same response is returned regardless of URI.

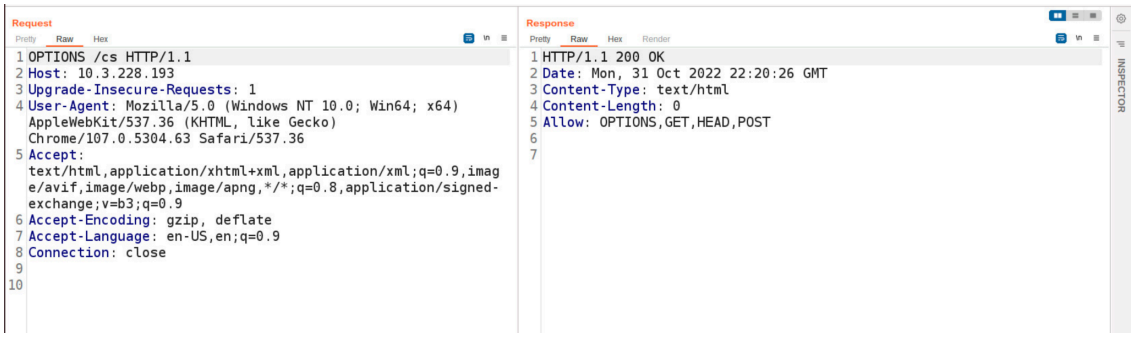

Figure 1. HTTP OPTIONS request with HTTP 200 response.

HTTP/HTTPs GET Request and Response Fingerprint

When a Team Server starts, the HTTP server exposes certain URIs. Figure 2 shows the list of URIs.

The URLs stager and stager64 are masked if the profile has the set host_stage "false"; option set. The HTTP server will return HTTP status code 404 if the URI begins with a forward slash (/).

URI	Host	Port	Type	Description
stager64		80	beacon	beacon stager x64
stager		80	beacon	beacon stager x86
beacon.http-post		80	beacon	beacon post handler
beacon.http-get		80	beacon	beacon handler

Figure 2. List of the URIs from the Team Server.

Request to Stager URI

When a user sends the following HTTP request to a Team Server, the server will return the 32-bit Beacon binary to the client. Figure 3 shows the HTTP request and response. Note the lack of a forward slash (/) at the beginning of the URI path.

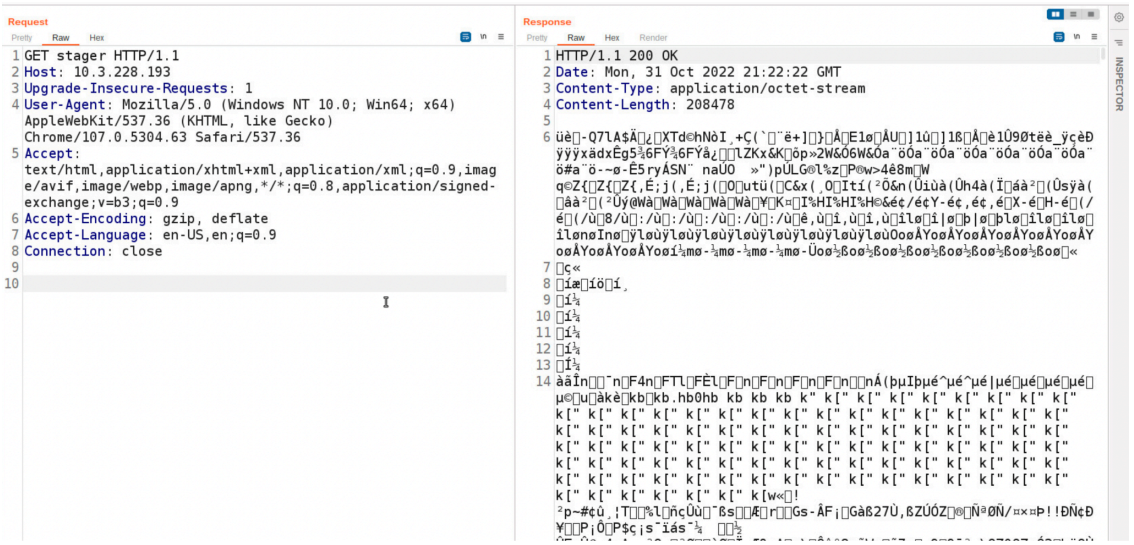

Figure 3. HTTP GET request and response for a 32-bit Beacon payload.

Request to Stager64 URI

To receive a 64-bit Beacon payload, a user must send an HTTP GET request to the URI stager64. Figure 4 shows the HTTP request and response for a 64-bit Beacon payload.

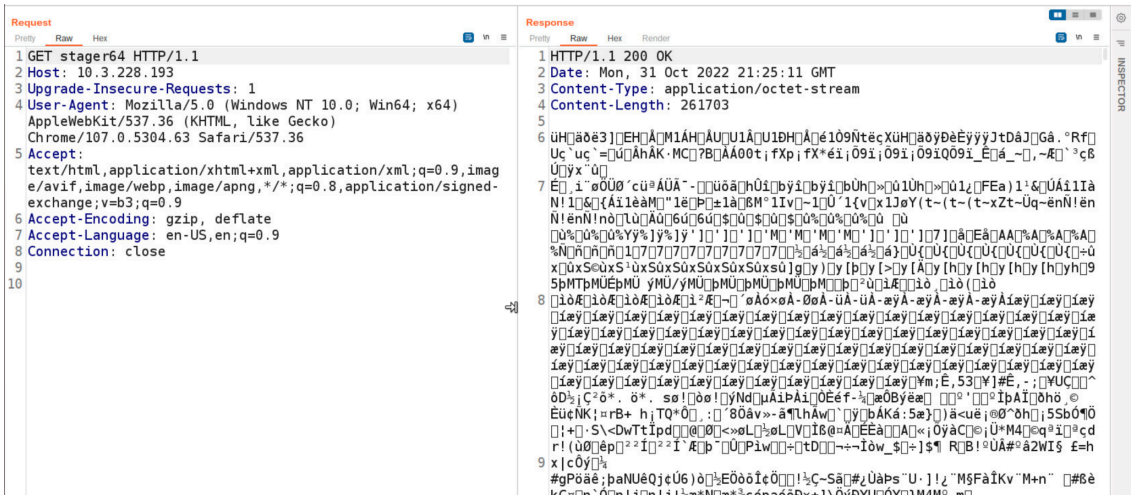

Figure 4. HTTP GET request and response for a 64-bit Beacon payload.

Request to Beacon.http-get URI

Certain preset URI paths can be configured in the Malleable C2 profile to serve static data. If a user sends a GET request to the URI beacon.http-get, the Team Server responds with the data that has been specified in its profile. Specifically, it sends the output section within the server tag of the http-get configuration.

If the output section only contains the command print;, the server responds with HTTP status code 200 and Content-Length: 0. Figure 5 shows the HTTP request and response with the default profile.

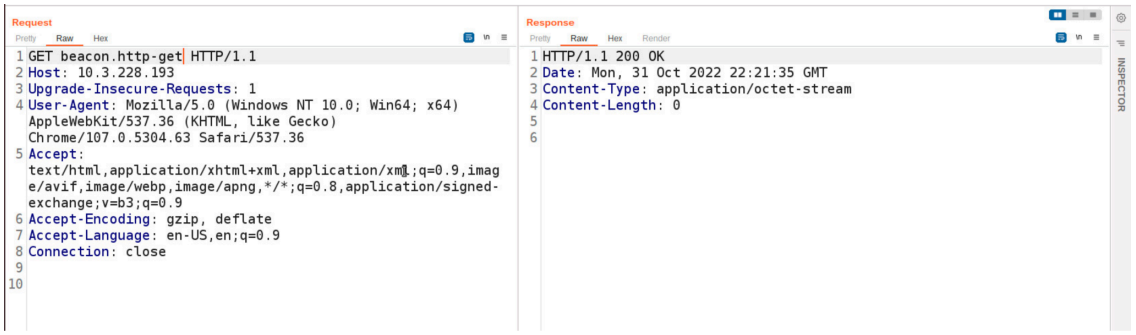

Figure 5. HTTP request and response with default profile.

If Team Server initializes with the Malleable C2 Gmail profile, the server responds with static data presented as described above. In this profile, GET requests to beacon.http-get result in a response containing a JavaScript payload.

Figure 6 shows the HTTP request and response generated by a Beacon session preset with the Gmail Malleable C2 profile.

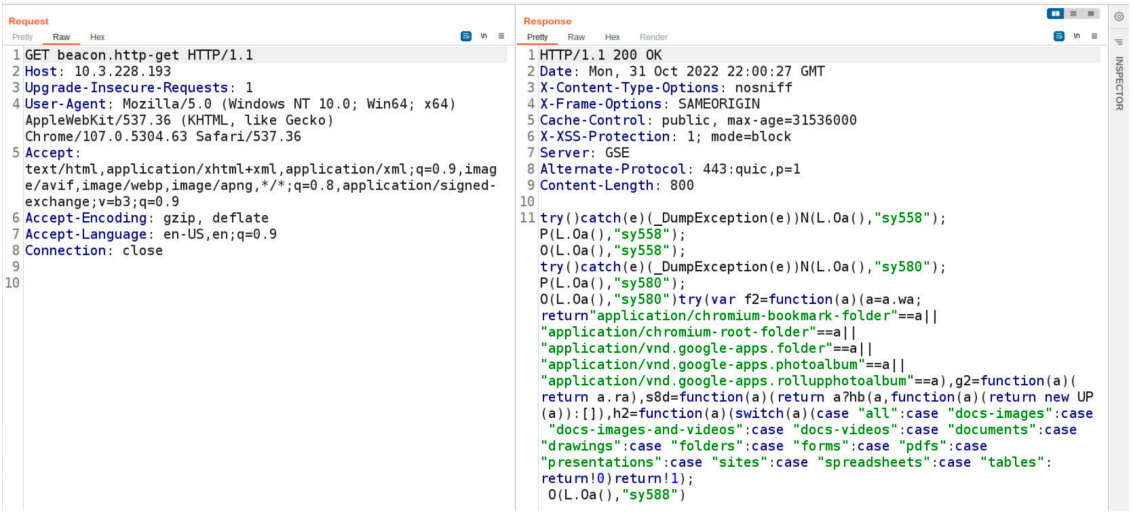

Figure 6. HTTP request and response configured with the Gmail Malleable C2 Gmail profile.

Request to Beacon.http-post URI

Team Server's behavior is the same for a GET request to the URI beacon.http-post as it is for the URI beacon.http-get. Figure 7 shows the HTTP request and response for a Team Server instance that initializes with the default Malleable C2 profile.

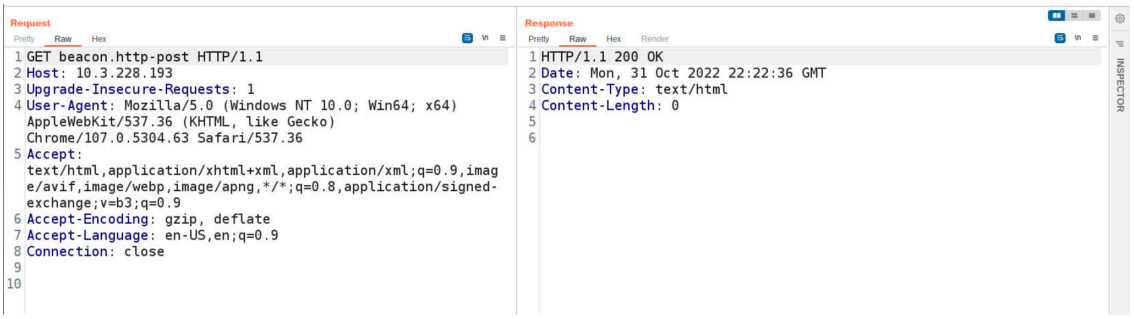

Figure 7. HTTP request and response configured with the default Malleable C2 profile.

Figure 8 shows an HTTP transaction when a GET request for beacon.http-post is sent to a Team Server instance that has initialized with the Gmail Malleable C2 profile.

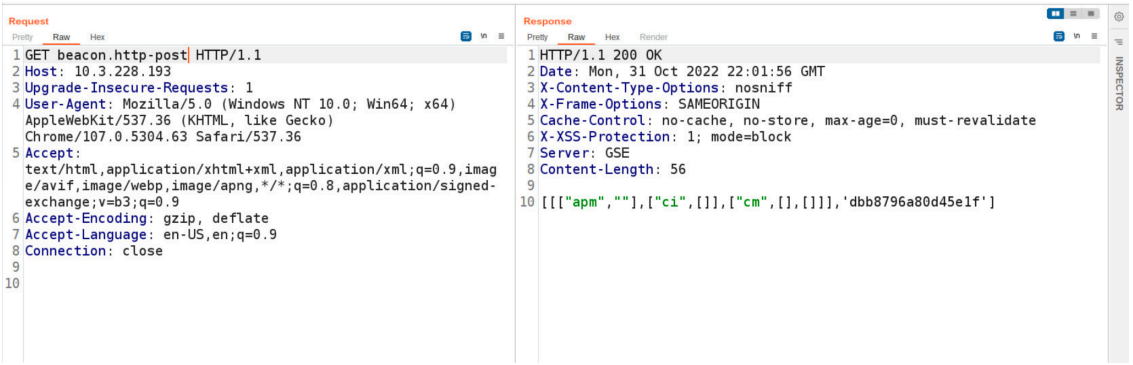

Figure 8. HTTP request and response configured with the Malleable C2 Gmail profile.

URI Checksum

Team Server utilizes a custom one-byte checksum of the request URI as a condition to serve the 32-bit or 64-bit version of the Beacon binary. A simple checksum algorithm implemented in Java named checksum8 is used to calculate the checksum of the request URI.

As shown in Figure 9, for 32-bit payloads, the code compares the URI checksum result to the literal integer 92L (where the L suffix is Java syntax for integer type long). For 64-bit payload requests, the algorithm compares the checksum to 93L.

```
public static long checksum8(String text) {
  if (text.length() < 4) {
    return 0L;
  }
  text = text.replace("/", "");
  long sum = 0L;
  for (int x = 0; x < text.length(); x++) {
    sum += text.charAt(x);
  }

  return sum % 256L;
}


public static boolean isStager(String uri) { return (checksum8(uri) == 92L); }


public static boolean isStagerX64(String uri) { return (checksum8(uri) == 93L && uri.matches("/[A-Za-z0-9]{4}")); }
```

Figure 9. Code to check the URI checksum.

When a user sends a GET request to a Team Server, the URI is passed to checksum8 and is compared to both integer values 92L and 93L. If the checksum satisfies one of the conditions, the server will respond with the raw bytes of the appropriate Beacon binary.

Figure 10 details an example of a URI that computes to a value that satisfies the checksum8 condition, as well as the Team Server's response with the Beacon binary payload. This information was extracted from Beacon configuration scripts, which continue to provide threat intelligence that is useful for preventing Cobalt Strike connections.

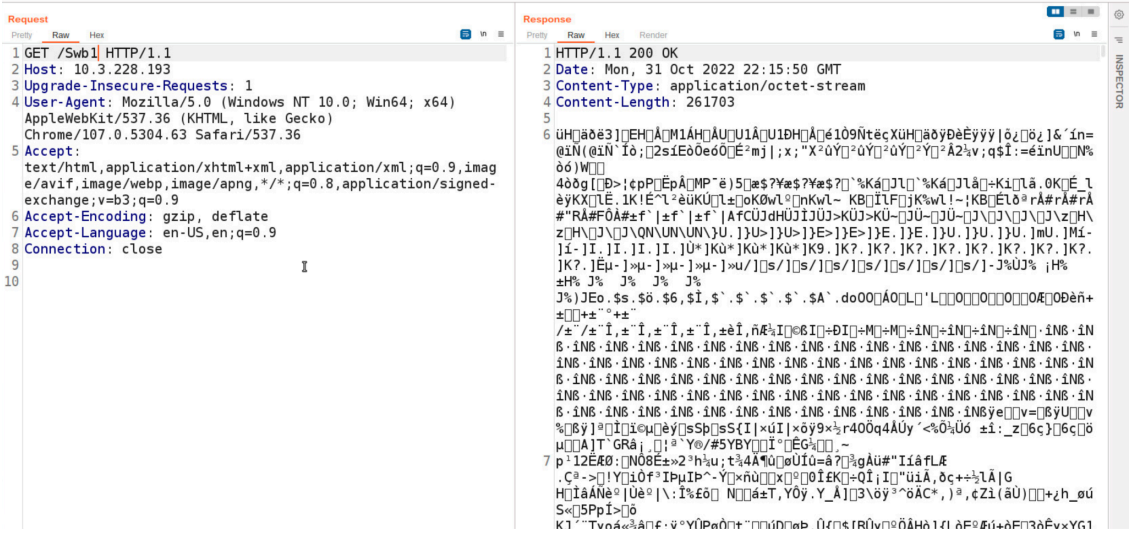

Figure 10. HTTP GET request and response with a URI that satisfies checksum8.

Random URI

If a user sends a randomized URI path, the Team Server will respond with HTTP status code 404 with Content-Length: 0. Figure 11 shows the HTTP response from a Team Server when a user sends a GET request with the URI randomURI.

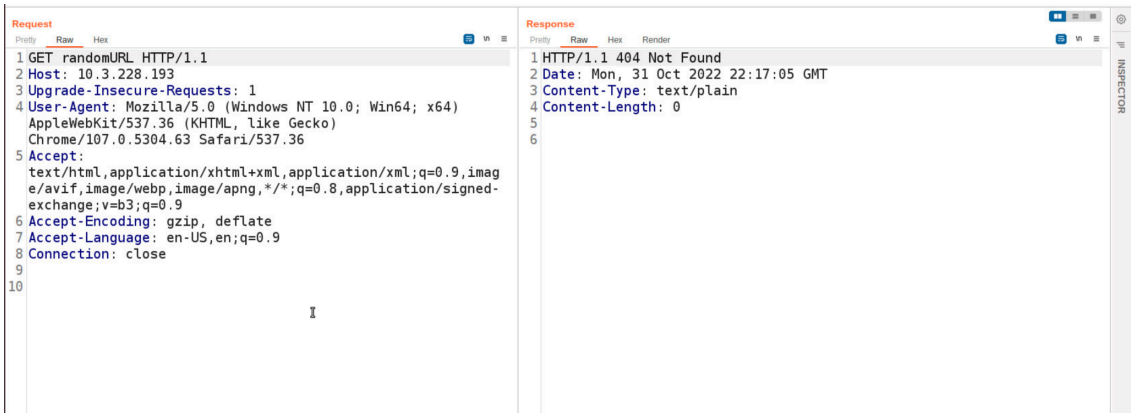

Figure 11. HTTP GET request and response for a randomURI path.

Active Probing Over DNS

Cobalt Strike's DNS listener enables Beacon implants to covertly utilize the DNS protocol to communicate with the Team Server. The DNS-based Beacon uses the DNS TXT, AAAA, and A records for task monitoring and other related functions. The configuration is set by data channel mode in the Malleable C2 profile.

Figure 12 shows a DNS request originating from a Beacon querying the TXT record for the domain aaa.stage[.]xx.

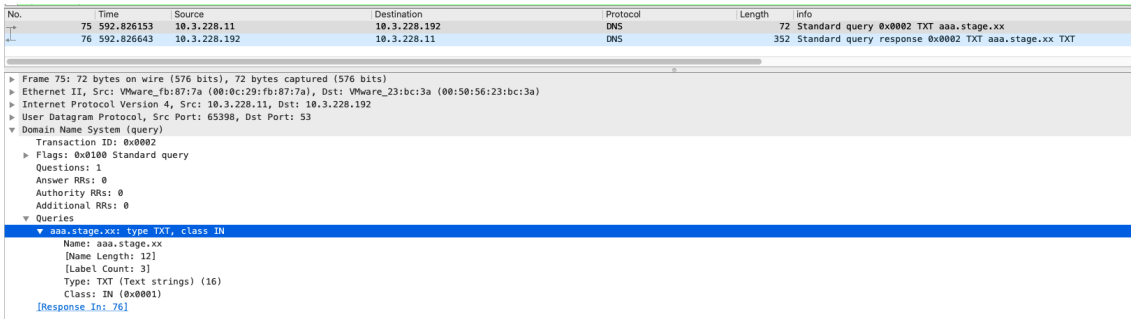

Figure 12. Beacon DNS request for TXT record.

Once the request is received, the Team Server responds with the base64-encoded Beacon binary in the

TXT

record response, as shown in Figure 13 below:

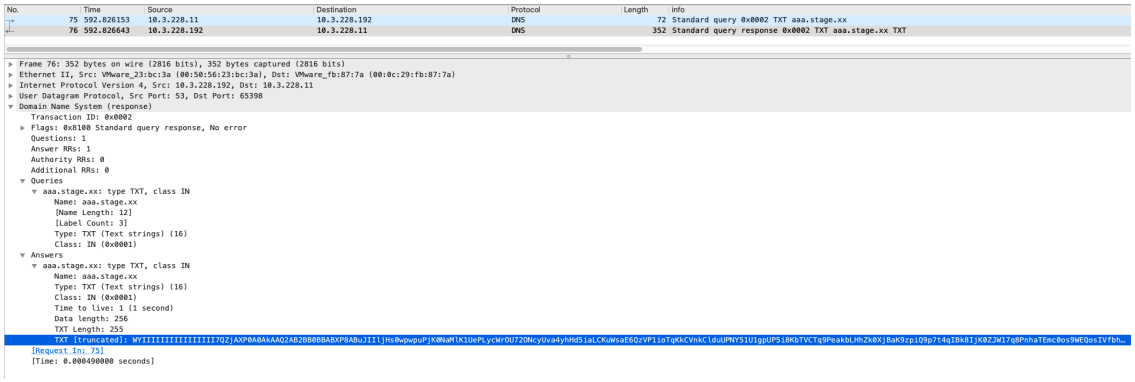

Figure 13. Beacon DNS Listener response (base64-encoded data).

Team Server Found in the wild

Based on the fingerprints and signals discovered, we utilized open source threat intelligence feeds including ZoomEye, Shodan and Censys to scour the internet in search of undetected Cobalt Strike Team Servers in the wild.

The following table details IP/port and URI indicators of compromise (IoCs) related to live Team Server instances we discovered in the wild in September 2022. We utilized Shodan's feed service to collect IP addresses of potential Team Servers before sending 32-bit stager probes to test the daemon for positive indicators. Once a candidate returns the expected Cobalt Strike response, we initialize a TCP connection with netcat to test, verify and extract the served stager bytes as shown in Figure 14.

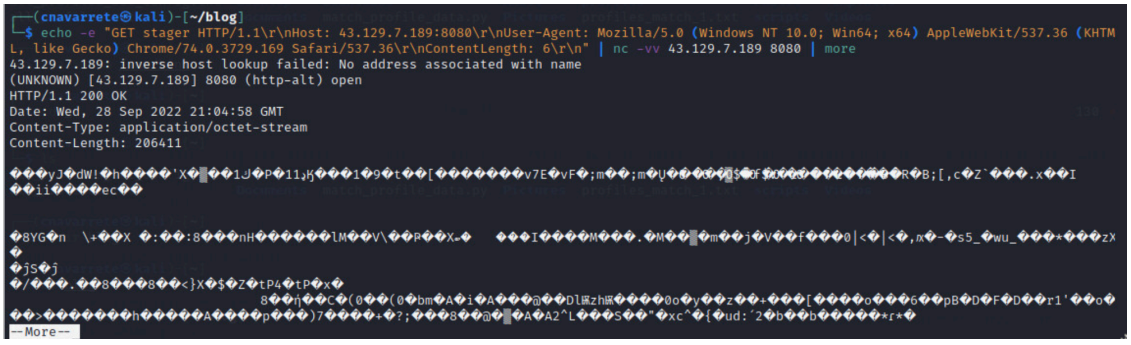

Figure 14. Successful HTTP/S probing for Team Server via stager check.

IP Address:Port	Payload Type	GET-URI	POST-URI
43[.]129[.]7[.]189:8080	windows-beacon_http-reverse_http	/updates	/aircanada/dark.php
117[.]50[.]37[.]182:80	windows-beacon_http-reverse_http	/api/x	/api/y
42[.]192[.]206[.]174:80	windows-beacon_http-reverse_http	/dpixel	/submit.php
194[.]37[.]97[.]160:80	windows-beacon_http-reverse_http	/cx	/submit.php
92[.]222[.]172[.]39:80	windows-beacon_http-reverse_http	/ptj	/submit.php
79[.]141[.]169[.]220:443	windows-beacon_http-reverse_http	/pixel.gif	/submit.php

Table 1. IP/port and URI IoCs related to live Team Server instances.

Conclusion

Cobalt Strike is a potent post-exploitation adversary emulator that continues to evade conventional next-generation solutions, including signature-based network detection. However, Advanced Threat Prevention's inline deep-learning models and heuristic techniques provide defenses against Cobalt Strike Beacon and Team Server C2 communication before they occur.

The probing and fingerprint technology detailed in this publication is very efficient and reliable in identifying Cobalt Strike instances in the wild with a very high degree of certainty. A single modern network security appliance is not sufficient to provide comprehensive coverage against complex malicious tools such as Cobalt Strike. Only a combination of security solutions including firewalls, sandboxes, endpoint agents and cloud-based machine learning can integrate the required data to prevent advanced adversaries from mounting successful cyberattacks from end to end.

Palo Alto Networks customers receive protection from this kind of attack by the following:

- Next-Generation Firewalls (NGFWs) with Threat Prevention signatures 86445 and 86446 can identify HTTP C2 requests with the base64 metadata encoding in default profiles.
- Next-Generation Firewalls with Advanced Threat Prevention subscription can identify and block the Cobalt Strike HTTP C2 request in non default profiles.
- WildFire, an NGFW security subscription, and Cortex XDR identify and block Cobalt Strike Beacon.
- Cortex XSOAR response pack and playbook can automate the mitigation process.
- Cortex XDR will report related exploitation attempts.
- Malicious URLs and IPs have been added to Advanced URL Filtering.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America Toll-Free: 866.486.4842 (866.4.UNIT42)
- EMEA: +31.20.299.3130
- APAC: +65.6983.8730
- Japan: +81.50.1790.0200

Indicators of Compromise

CS Samples

- 50ea11254f184450a7351d407fbb53c54686ce1e62e99c0a41ee7ee3e505d60c

CS Beacon Samples

- /lNj8
    - SHA256 hash:
        - e712d670382ad6f837feeb5a66adb2d0f133481b5db854de0dd4636d7e906a8e

CS Teamserver IP addresses

- 92[.]255[.]85[.]93
- 43[.]129[.]7[.]189
- 117[.]50[.]37[.]182
- 42[.]192[.]206[.]174
- 194[.]37[.]97[.]160
- 92[.]222[.]172[.]39
- 79[.]141[.]169[.]220

Source: https://unit42.paloaltonetworks.com/cobalt-strike-team-server/