How to Identify Cobalt Strike on Your Network

By Zohar Buber

Published: 2020-11-18 · Archived: 2026-04-06 00:41:24 UTC

5 Min Read

Since its introduction, Cobalt Strike has become one of the most prevalent threat emulation software packages used by infosec red teams. Unfortunately, its combination of multiple exploitation techniques also makes Cobalt Strike a platform of choice by attackers.

In the past several months, we've seen Cobalt Strike used in multiple exploits. In the WastedLocker ransomware attack, an advanced persistent threat (APT) group used Cobalt Strike to move laterally within a network. APT groups also used Cobalt Strike in the military-themed malware campaign to target military and government organizations in South Asia.

Common antivirus (AV) systems, which focus on security data, often miss Cobalt Strike. The platform uses numerous techniques to evade detection. Moreover, Cobalt Strike can be merged with other attack tools like Mimikatz, Metasploit, and PowerShell Empire to move laterally across the network.

But there is good news for security professionals: Cobalt Strike has very distinct network markers. You can use those markets to detect Cobalt Strike on your network.

What's So Difficult About Detecting Cobalt Strike?
Cobalt Strike implements two main techniques to avoid detection by mainstream AV systems. It 1) obfuscates the shellcode and 2) leverages a domain-specific language called Malleable Command and Control (Malleable C2). Let's look at each one.

Technique #1
AV systems today commonly implement sandboxing to detect executables. Sandboxing provides a separate environment to run and inspect suspicious executables. Cobalt Strike, though, hides shellcode over a named pipe. If the sandbox doesn't emulate named pipes it will not find the malicious shellcode. In addition, the attacker can modify and build his own techniques with Cobalt Strike Artifact Kit.

Technique #2
In post-exploitation, Cobalt Strike mimics popular services, such as Gmail, Bing, and Pandora, to evade detection. The platform uses Malleable C2, which provides attackers with the ability to modify Cobalt Strike command-and-control (C2) traffic to their will. The attacker can then identify legitimate applications within the target organization, such as Amazon traffic, and modify the C2 traffic to appear as Amazon traffic using any number of publicly available profiles, like this one for Amazon on GitHub.

In the screenshot below (Figure 1) you can see Cobalt Strike profile that fakes CNN video URI, and HTTP headers like "Host," "Referer," and "X-requested-With" so the HTTP request will look like a request to CNN

video.

Network Indicators for Detecting Cobalt Strike

To identify Cobalt Strike, examine the network traffic. Since Cobalt Strike default profiles evade security solutions by faking HTTPS traffic, you need to use TLS Inspection. Then isolate bot traffic and, once done, identify the suspicious traffic by examining data within HTTPS requests.

Examine Network Communications

To distinguish human-generated traffic from bot-generated traffic, we examine the frequency of communications to a target. Bot-generated traffic tends to be consistent and uniform, as you can see below at the flow frequency graph. Human-generated traffic tends to vary over time, while machine-generated traffic tends to be almost uniformly distributed.

Just because traffic is generated by a bot doesn't make it malicious, however. There are numerous good bots, such as OS updaters. You need to identify bots likely to be suspicious, and you can do that by digging into the traffic flow.

Examine the User Agent

Looking at the origin of the bot traffic, we inspect the user agent generating the TLS traffic. At first, the user agent looks legitimate, allegedly generated by Mozilla/5.0 (Windows NT 6.1), the value for Internet Explorer (IE).

However, user agents can easily be faked. Some machine learning algorithms derive the true user agent of packet flows and, in this case, flagged it as "unidentified." The discrepancy gives us a strong indicator that we're likely looking at malicious traffic.

Examine the Destination

Next, we examine the destination domain -- dukeid[.]com. For many, this point will be less conclusive. According to VirusTotal, we can see that less than 10% of the 83 AV engines (seven to be exact) tagged this domain as malicious. However, vendor reputation models have classified as dukeid.com as malicious giving us another Indicator of Compromise (IoC) or network artifact likely indicating an intrusion.

Examine the Host Header

We move on deeper into the packet and examine the HTTP host header, which in this case was www.amazon.com. However, traffic was directed to the domain, "dukeid[.]com". This gives us another powerful piece of evidence that we're looking at Cobalt Strike as faking host header is part of Cobalt Strike's Amazon Profile.

Examine the URI

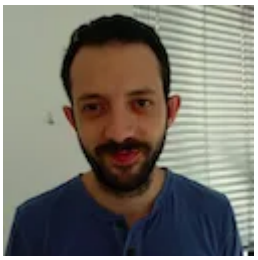
Finally, we examine the target uniform resource identifier (URI) of the flow. We see that URI matches one associated with Cobalt Strike Malleable C2:

/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books

Alone, blocking the URI won't be effective. It's a fake Amazon URI and blocking it would also block traffic to legitimate Amazon URIs. Hence the need to proceed through the steps outlined above.

The Malleable C2 module in Cobalt Strike is an advanced tool that allows attackers to customize beacon traffic and create covert communications. AV systems may not be enough to protect a network. Even after the threat had been identified and the customer notified, their AV systems were still unable to detect and remove the threat. Focusing on the malware's network characteristics, though, allowed the threat to be identified. It's an excellent example of how combining networking and security information can lead to better threat detection.

About the Author



Security Analyst

Zohar Buber is a security analyst in Cato Research Labs at Cato Networks. He focuses on network protocol analysis and malicious traffic detection, specializing in threat identification using network-based methods.
