Discovering Linux ELF beacon of Cobalt Strike tool

By Anonymous

Published: 2021-09-19 · Archived: 2026-04-05 14:51:24 UTC

This post was authored by Fareed.

Introduction

Cobalt Strike is a tool used for red teaming and penetration testing to demonstrate the cyber attack. Cobalt Strike is a commercial, full-featured, remote access tool that bills itself as "adversary simulation software designed to execute targeted attacks and emulate the post-exploitation actions of advanced threat actors". Cobalt Strike’s interactive post-exploit capabilities cover the full range of ATT&CK tactics, all executed within a single, integrated system.

Nowadays, real attackers and cyber threat actors have been used this tool a lot in their operations to conduct a cyber-attack against their target. Today, we see another evolution of Cobalt Strike where the threat actor has developed a Linux version of a payload of Cobalt Strike where it gets 0 detection in the VirusTotal and remains stealthy in our client premises for more than 3 months.

Graph flow

Figure 1: Graph flow of the malware

Based on figure 1, the malware has a simple function routine. The core function of this malware is to make the beacon connection to the attacker Cobalt Strike server via DNS using the Cobalt Strike Malleable C2 config embedded in the malware. The other interesting part when reversing this malware is how they decrypt strings and data in the heap and parse those data to set up the DNS request.

Technical analysis findings

Initial assessment

Based on the THOR scanner's result given to the Netbytesec team, the hashes of the malware are as follows:

- 3db3e55b16a7b1b1afb970d5e77c5d98
- c5718ec198d13ef5e3f81cecd0811c98

The YARA rule that matched with the THOR scanner detection is "*CobaltStrike_C2_Encoded_Config_Indicator*" which creates the first bad indicator of this malicious file.

The Netbytesec analyst try to run the malware in our malware analysis lab using Ubuntu 20.04 OS but the malware giving an error related to library dependencies.

Tested on Ubuntu distro

After verifying the infected server's OS which is CentOS 7, the Netbytesec team runs the program in CentOS 7 to mimic the real infected infrastructure of the malware executed. As a result, the malware successfully runs in CentOS 7 without any error unlike in the Ubuntu OS. The Netbytesec team believes that the malware was customized to run only in RedHat Distribution as the Netbytesec experimenting to run the malware in Debian and Ubuntu and give us the same error result.

Tested on CentOS 7

Our first initial assessment is to check for any detection from the Anti-Virus engine in the VirusTotal as our client has uploaded the malware into the VirusTotal. The malware has zero detection in VirusTotal as shown in figure 2 which is quite interesting to reverse this binary.

Figure 2: VirusTotal result

While in the Joe Sandbox, the file was detected as malicious with 48 scores per 100. In the sandbox result, the malware was detected as highly malicious only because of the YARA signatures that detect Cobalt Strike C2 encoded profile config instead of malicious behavior activity.

Figure 3: Joe Sandbox result

Figure 4: YARA signature in Joe Sandbox

To verify the Cobalt Strike C2 profile, the Netbytesec analyst manually analyzed the Linux program without depending only on the YARA signature as it can be false positive. The Netbytesec analyst parses the profile config to extract the information of the Cobalt Strike config. As shown in figure 5 below, the information about the Cobalt Strike C2 config profile can be read include the C2 server DNS and other details.

Figure 5: CS config information

Based on figure 5, the beacon type of this malware will be deliver using DNS. Thus, DNS beaconing will be performed when the malware is executed. Also, we can see the “PublicKey_MD5” value. Once the beacon is executed, the beacon then needs to communicate with the Team Server. Whenever a beacon checks in to the Cobalt Strike Team Server destination, it sends an encrypted metadata blob. The metadata blob is encrypted using this RSA public key, extracted from the stager which is the malware. The other interesting information about the config is how the data communication request headers include GET and POST requests between the infected machine and the Cobalt Strike team server will be communicated after the malware run.

The malware used two C2 server domains which are update.microsoftkernel.com and update.microsofthk.com. Inspecting both domains showing that the domains resolved to IP address 160.202.163.100 and last seen was in 2021-09-06. From the result above, the Netbytesec team conduct OSINT research on both domains and found a tweet about it from the year 2019. Based on the tweet shown in the figure below, this is another strong indicator that the author of the malware uses the Cobalt Strike tool as part of the cyberattack on our client-server infrastructure.

Figure 6: A tweet containing both C2 server domains

Reverse engineering analysis

Focusing on the malware inner code itself, once the malware is executed, it will run in the background silently without giving any output on the screen as the first code of the main function of the malware is calling the daemon

function at *line 8* shown in the decompiler screenshot below. (Most of the function has been renamed to ease our analysis).

Figure 7: The first part of the main function

At the address *4035F3 (line 9)*, the malware will execute a subroutine renamed as "*wrap_sys_getid*" which will be used to get thread ID (TID) or the Process ID (PID) of the running malware. This called function will return the number of the TID and serve the number into function "*j_srand*" to generate an integer number.

On the next line at the address *4035FF* which is renamed as "*wrap_decryption*", the malware calls this subroutine function that will be used to decrypt and parse most of the encrypted strings and data in the malware executable to be used to create network request to the attacker infrastructure. In figure 8 below, the decryption routine of the XOR encoded Cobalt Strike config using XOR key *0x69*.

Figure 8: Decryption algorithm of encoded CS config

Moreover, in the function renamed as "*wrap_decryption*", the malware also includes a function to read resolver configuration files which are used to configure DNS name servers in Linux OS.

Figure 9: Read resolve.conf

When debugging a part of "*wrap_decryption"*, there is a part where the program will parse out the C2 domains from the config allocated in memory previously explained. The parsed C2 domains is made at the line shown in Figures 10.

Figure 10: Parsing the first C2 domain from config

The below figure shows some of the strings that have been decrypted and parsed in memory after the function "*wrap_wrap_decryption*" finished execute. These strings did not appear if the program are not running. Thus, dynamic analysis or execution is required to dump the strings from the memory. These strings then will be used to build and create C2 communication in a function renamed as "*wrap_connection_c2*" explain later.

Figure 12: In-memory strings

Besides that, another interesting function to investigate is a function renamed as "*wrap_enumeration_and_createb64*". In the subroutine function, it will enumerate and get the infected host's information.

Figure 13: Subroutine wrap_enumeration_and_createb64 decompiled code

For example, in the following figure, the malware tries to get the process ID and uname.

Figure 14: Function getpid and wrap_uname get the call

Other than that, the information that the malware enumerates is the PID of the process, IP address of the host, UID, hostname, and kernel info. All this info will be appended and save in the heap. This collected host information will be encrypted using the public RSA key then encoded using base64. The construction of the base64 string in the malware is located at function address *0x40C53B*.

Figure 15: Base64 construction

The generation of the base64 is using the base64 BIO filter function from the OpenSSL crypto library. This is a filter BIO that base64 encodes any data written through it and decodes any data read through it. The base64 will be saved in the heap that will be used as part of communication in the DNS beaconing.

The last part of the malware's core functionality is the C2 connection and sleep function shown in the red box as follows:

Figure 16: While loop of the C2 beacon connection

The function "wrap_connection_c2" is a switch case that contains 6 cases. The most important and interesting case is case 2 where all the beacon connections are created and make to communicate with the C2 server.

Figure 17: Part of case 2 code

In the *sub_40D440* subroutine function, the Netbytesec team analyst discovers the IP address of the Cobalt Strike as it appends the IP address to the request header "*Host: "*. As seen in the figure below, IP address *160.202.163.100* is matching with the PCAP communication. This IP address is matching with the previous OSINT research result on both domains.

Figure 18: IP address append to host

OSINT research on this IP found that the IP has a historical record of the Cobalt Strike server using port 80.

Figure 19: https://github.com/fox-it/cobaltstrike-extraneous-space/blob/master/cobaltstrike-servers.csv

Also, the passive DNS of the IP is matched with the DNS found early which are *microsoftkernel.com* and *microsofthk.com*. The IP was also flagged as malicious which has communicated with a malicious file recently.

Figure 20: Virustotal result of the malicious IP

Finally, the connection is officially create using the function *BIO_s_connect()*. This is a wrapper around the platform's TCP/IP socket connection routines. The connection is created and set up using the strings and data like URL and cookies append along with the GET request headers to complete the connection to the attacker infrastructure.

Figure 21: Function BIO_s_connect called by the malware

DNS beaconing

The figure below shows the malware requesting tasks via DNS which response using the TXT channel.

Figure 25: Pcap capture of the TXT query and response

The result of the TXT query is encoded in base64 and encrypted in AES contains the task from the Team Server. For example:

Figure 26: Pcap capture of the TXT query and response

As explained from the CS blog, when Cobalt Strike’s DNS server gets the request, it checks if any tasks are available for that Beacon instance. If no tasks are available, it returns a response that tells Beacon to go to sleep. If a task is available, the Cobalt Strike DNS server returns an IP address. The compromised system then connects to that IP address and makes an HTTP GET request to download its tasks. The tasks are encrypted. That's why we see there is HTTP GET request construction in the code. This is the hybrid communication model. The idea here is that DNS requests on a regular interval are less likely to get noticed than, say, HTTP requests on a regular interval.

To track the compromise events, the NetByteSec Splunk analyst discovers that a few servers of our client's have made the DNS beaconing to the Cobalt Strike since April 2021.

Figure 27: Splunk detection on the IP

Conclusion

The attacker has dropped their malicious software in the servers and run the malware. The malware has the ability to run in the background and create a DNS beacon connection to the Cobalt Strike C2 server hosted on IP 160.202.163.100. Before the malware is set up and creates the connection, the malware will decrypt a lot of strings and data include Cobalt Strike config, and then parse and append it to the function that will make the connection function. Most of the findings of OSINT research of the found IP address and Domain tell that these IOCs are positive belong to an attacker that is using the Cobalt Strike tool to conduct the attack. The Netbytesec team believes that this cyber attack was conducted by an actor who were able to tweak and port the Cobalt Strike payload to Linux's based version and remain stealthy after compromised our client.

Indicator of Compromises

IP address

- 160.202.163.100

Domains

- microsoftkernel.com
- microsofthk.com

Subdomains

- update.microsoftkernel.com

- update.microsofthk.com

Hash

- 3db3e55b16a7b1b1afb970d5e77c5d98
- c5718ec198d13ef5e3f81cecd0811c98
