

Vermilion Strike: Linux and Windows Re-implementation of Cobalt Strike

 intezer.com/blog/malware-analysis/vermilionstrike-reimplementation-cobaltstrike/

September 13, 2021

Written by [Avigayil Mechtinger](#), [Ryan Robinson](#) and [Joakim Kennedy](#) - 13 September 2021



Get Free Account

[Join Now](#)

Top Blogs

Detecting Phishing Emails with Email Headers, Attachments, and URLs

Emails were created as a method to pass messages between users, and now they are...

[Read more](#)

Automating Alert Triage and Threat Hunting with Intezer + SentinelOne

One of the biggest pain points of cyber security teams is alert fatigue – trying... [Read more](#)

Top Cyber Threats to the Telecom Industry

In our interconnected society, the telecom industry is responsible for keeping the world connected 24/7.... [Read more](#)

Key Findings

- Discovered Linux & Windows re-implementation of Cobalt Strike Beacon written from scratch
- Linux malware is fully undetected by vendors
- Has IoC and technical overlaps with previously discovered Windows DLL files
- Highly targeted with victims including telecommunications, government and finance

Cobalt Strike is a popular red team tool for Windows which is also heavily used by threat actors. At the time of this writing, there is no official Cobalt Strike version for Linux.

In August 2021, we at Intezer discovered a fully undetected ELF implementation of Cobalt Strike's beacon, which we named **Vermilion Strike**. The stealthy sample uses Cobalt Strike's Command and Control (C2) protocol when communicating to the C2 server and has Remote Access capabilities such as uploading files, running shell commands and writing to files. The malware is fully undetected in VirusTotal at the time of this writing and was uploaded from Malaysia.

Based on telemetry with collaboration from our partners at McAfee Enterprise ATR, this Linux threat has been active in the wild since August targeting **telecom companies, government agencies, IT companies, financial institutions** and **advisory companies** around the world. Targeting has been limited in scope, suggesting that this malware is used in specific attacks rather than mass spreading.

After further analysis, we found Windows samples that use the same C2. The samples are re-implementations of Cobalt Strike Beacon. The Windows and ELF samples share the same functionalities.

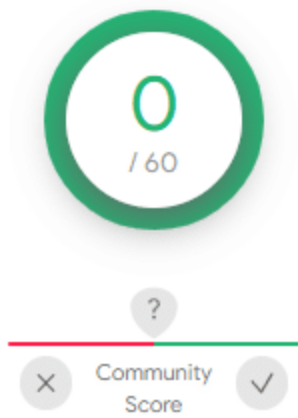
The sophistication of this threat, its intent to conduct espionage, and the fact that the code hasn't been seen before in other attacks, together with the fact that it targets specific entities in the wild, leads us to believe that this threat was developed by a skilled threat actor.

In this post we will provide a technical analysis of the samples and explain how you can detect and respond to this threat.

Technical Analysis

Linux File

The file was uploaded to VirusTotal from Malaysia and has no detections in VirusTotal at the time of this writing.



No security vendors flagged this file as malicious

294b8db1f2702b60fb2e42fdc50c2cee6a5046112da9a5703a548a4fa50477bc

x11-gtk

64bits elf

294b8db1f2702b60fb2e42fdc50c2cee6a5046112da9a5703a548a4fa50477bc in VirusTotal

The screenshot shows the Intezer Analyze interface for a file identified as Malicious (Main Family: VermilionStrike). The file's SHA256 hash is 294b8db1f2702b60fb2e42fdc50c2cee6a5046112da9a5703a548a4fa50477bc. It is classified as Malware (VermilionStrike) with 214 genes. The analysis shows a 94.22% match with VermilionStrike, 3.24% with CobaltStrike, and 3.24% with the Malicious Library. The file is 87.32 KB, ELF executable, and built on a Red Hat Linux distribution. It uses OpenSSL via dynamic linking. The interface also shows file metadata and a link to the VirusTotal report.

Vermilion Strike analysis in Intezer Analyze.

The file shares strings with previously seen Cobalt Strike samples and triggers a number of YARA rules that detect encoded Cobalt Strike configurations. The ELF file is built on a Red Hat Linux distribution. It uses OpenSSL via dynamic linking. The shared object names for OpenSSL on Red Hat-based distributions are different from other Linux distributions. Because of this, it can only run on machines with Linux distribution based on Red Hat's code base.

Initialization

Decoded configuration of the beacon.

Further decryption is performed in a heap with decoded strings, keys, and values required by the beacon for its operation. The beacon will then generate a SHA256 hash sourced from a random number seeded from the thread ID. This value will be used later in DNS beaconing. Next, a public RSA key will be imported for later use.

```
0x0040736f    nop
; CALL XREF from main @ 0x403610
0x00407370    sub rsp, 8
0x00407374    mov edx, dword [0x006157e8]
0x0040737a    xor edi, edi
0x0040737c    mov esi, 0x6157e0
;-- rip:
0x00407381    call sym.imp.d2i_RSA_PUBKEY
0x00407386    mov rdi, rax
0x00407389    xor eax, eax
0x0040738b    test rdi, rdi
;=> 0x0040738e    je 0x4073a7
| 0x00407390    mov qword [0x00615880], rdi
| 0x00407397    call sym.imp.RSA_size
| 0x0040739c    mov dword [0x00615888], eax
| 0x004073a2    mov eax, 1
^-> 0x004073a7    add rsp, 8
0x004073ab    ret
0x004073ac    nop dword [rax]
0x004073b0    push r15
0x004073b2    mov r15, rdi
0x004073b5    push r14

:~ px @ [esi]
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 0123456789ABCDEF0123
0x00b1cb70 3081 9f30 0d06 092a 8648 86f7 0d01 0101 0500 0381 0..0...*.H.....
0x00b1cb84 8d00 3081 8902 8181 00c4 1073 71ff 57f1 b703 ba92 ..0.....sq.W....
0x00b1cb98 4bb1 8921 2a4f 3cc8 fcb3 1cfc 5541 cb6d 3b4e 31e1 K..!*0<....UA.m;N1.
0x00b1cbac 3573 403d 1b79 6846 d59c 3af0 2c70 9b73 2ec7 6516 5s@=.yhF...;p.s.e.
0x00b1cbc0 9e50 62da c64e c505 b68e 3e86 9660 0eec c0c6 2c5d .Pb..N....>..`.....,]
0x00b1cbd4 24d0 f60a 73d7 8286 af1b d285 949b b5f4 c849 b154 $.s.s.....I.T
0x00b1cbe8 98aa 9517 83df 6003 75b9 5a55 fdd4 933c f862 b4f3 .....`u.ZU...<.b..
0x00b1cbfc d700 0139 5a5b 9853 3001 8508 bc62 ca6e a702 0301 ...9Z[.S0...b.n....
```

Importing of public RSA key to encrypt machine fingerprint.

The beacon will begin fingerprinting the machine. A random number will be generated and the process ID will be fetched. It will grab the kernel version of the machine using **uname**. Next, the beacon will fingerprint network information through the **getifaddrs** function. It will loop through the interfaces looking for IPv4 addresses. It will gather the interface with an address not equal to "127.0.0.1" and stage the IPv4 address.

```
| :| | | 0x00403f02    be10000000    mov esi, 0x10    ; 16
| :| | | 0x00403f07 b    e864efffff    call sym.imp.getnameinfo ;[2]
| :| | | ;-- rip:
| :| | | 0x00403f0c    85c0          test eax, eax
:~ px @ rsp+0x50
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffd809f1d90 3137 322e 3137 2e30 2e32 0000 0000 0000 172.17.0.2.....
0x7ffd809f1da0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffd809f1db0 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

Network interface fingerprinting.

Next, the beacon will fingerprint the entry in the local password database for information about the current effective user ID of the process.

```
| 0x00404097 4883ec40 sub rsp, 0x40
| 0x0040409b e8a0edffff call sym.imp.geteuid ;[4] ; uid_t geteuid(void)
| 0x004040a0 89c7 mov edi, eax
| 0x004040a2 89c5 mov ebp, eax
| 0x004040a4 b e877efffff call sym.imp.getpwuid ;[5]
| ;-- rip:
|> px @ [rax]
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x0166f950 726f 6f74 0078 0030 3a30 3a72 6f6f 7400 root.x.0:0:root.
0x0166f960 2f72 6f6f 7400 2f62 696e 2f62 6173 6800 /root./bin/bash.
```

Fingerprinting of local password database.

The beacon will then fingerprint the hostname of the machine. The collected information will be formatted into a string, encrypted with the public RSA key, and base64 encoded, as is standard for communication with a Cobalt Strike server. The stages are shown below.

```
|> px @ rsi
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x00a151a0 0000 beef 0000 005d 9974 8fad fac2 af10 .....}t.....
0x00a151b0 1284 9974 61e9 909a 3533 3935 3809 3531 ..)o...539P8.51
0x00a151c0 094c 696e 7578 2d35 2e31 302a 3235 2d6c .Linux-5.10.25-1
0x00a151d0 696e 7578 6b69 7409 3137 322e 3137 2e30 inuxkit.172.17.0
0x00a151e0 2e32 0964 6636 6164 3338 3661 6136 3109 .2.df6ad386aa61.
0x00a151f0 726f 6f74 202a 0931 0931 0930 0931 2e30 root*.1.1.0.1.0
0x00a15200 2e31 2e4c 5200 0000 b100 0000 0000 0000 .1.LR.....
|> px @ rsi
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x00a19020 70ed 5b00 b566 da53 d958 fef5 adad 55f4 p.[.f.S.X...U.
0x00a19030 e0ae 9b54 d7f9 d591 c759 61c3 2b28 35c6 ...T....Ya.+(S.
0x00a19040 ec52 d6f3 876f 807e 194a af77 1ff5 5556 .R...o...J.w.UV
0x00a19050 6c35 27de eb7b f406 e21c b464 b141 1670 L5'...{...d.A.p
0x00a19060 eb5d ef85 4f0f f6c 9548 .].0...Ic...d.H
0x00a19070 b460 b1ab a640 f0c 9953 ...@...S...*[...S
0x00a19080 b36f b1f2 19a4 a54 7669 .o....G....vi
0x00a19090 c47e a2c2 27a4 951b 6ffe 3703 a53c 97b6 .v...'.o.7...<.
|> px @ rsi
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x00a1e100 634f 3162 414c 566d 326c 9056 5750 3731 c01BALVn21P2WP71
0x00a1e110 7261 3156 394f 4375 6d31 5458 2064 5753 rc1190Xcm1TKw-dR
0x00a1e120 7831 6c68 7779 736f 4663 6273 5574 6270 x11hwyoNchSUtz
0x00a1e130 6832 2b41 6668 6c4b 7233 6366 3956 5657 h2+AfH1Krc3cF9VW
0x00a1e140 6244 556e 3375 7437 3941 6269 484c 526b bDUh3ut79Ab1HLRk
0x00a1e150 7355 4557 634f 7464 3734 5650 442f 6168 sUEWc0td74VPD/dh
0x00a1e160 5354 7a2f 7a4c 706b 6c55 6930 594c 4772 Stz/zLpkU1u0VGr
0x00a1e170 7060 4477 41 a 6c54 p0wAFpypyxZIT
0x00a1e180 7332 2078 38 a 3863 S2+8mteq1fjpaJ8c
0x00a1e190 386d 4232 61 e 5562 #m3ecR2osImpJlb
0x00a1e1a0 622f 3433 4136 5538 6c37 593d 0000 0000 b/43AGU817Yr...
```

Stages of formatting the machine fingerprint.

Prepended to the fingerprint string is the value “1.0.1.LR”. This appears to be an internal version string. A similar string, “W1.0.1,” was found in a newly discovered Windows sample of Vermilion Strike that shares the same C2 and malware functionality.

The encrypted data is sent to the C2 server in a similar way that the metadata is sent from a Cobalt Strike beacon to the C2 server. The payload that is encrypted starts with the marker **0xbeef**. The same marker is used by the legitimate Cobalt Strike beacon.

Command and Control

Command and Control is primarily performed over DNS but also available over HTTP. This DNS-based approach for communications can help avoid traditional defenses that monitor HTTP traffic. Commands are received via DNS Address (A) and Text (TXT) records. The beacon first makes DNS requests out to hardcoded subdomains and gets an IP address returned. Normally, DNS requests on hostnames are intended to be translated into an IP address for which to visit. In this case, the IP address returned is not used as an IP address but for triggers to change the beacon behavior.

Once the beacon gets the signal to download a task, it will perform a DNS TXT query to the domain's nameservers, as shown below.

11	11.623868235	10.0.2.15	168.63.129.16	DNS	-- Standard query 0x77a4 A 86907.update.microsoftkernel.com
12	11.933851334	168.63.129.16	10.0.2.15	DNS	-- Standard query response 0x77a4 A 86907.update.microsoftkernel.com A 255.255.255.242
13	11.934029265	10.0.2.15	168.63.129.16	DNS	-- Standard query 0xfbfe A apple.dPK8sNHbf.86907.update.microsoftkernel.com
14	12.370098937	168.63.129.16	10.0.2.15	DNS	-- Standard query response 0xfbfe A apple.dPK8sNHbf.86907.update.microsoftkernel.com A 0.0.0.64
15	12.370263557	10.0.2.15	168.63.129.16	DNS	-- Standard query 0xd56b TXT facebook.aNJQhxc3l.86907.update.microsoftkernel.com
16	12.683275494	168.63.129.16	10.0.2.15	DNS	-- Standard query response 0xd56b TXT facebook.aNJQhxc3l.86907.update.microsoftkernel.com TXT

Packet capture of C2 communication.

The result of the TXT query is a base64 encoded and AES encrypted struct containing task information. An example of a returned task is shown below.

```
> ps @ [rdi]
wQ7mYmyLh2gijCitUJCe5hI2hjrY5HmdSwc0d0ARn+TQKYLPUmioT59HiMW//f8aaRop4UAD0tum/j2pBhxqGg==
```

A DNS TXT query result for a task.

A decrypted task is shown below.

```
> px @ rsi
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x00635f80 6123 8bfb 0000 0010 0000 0004 0000 0008 a#.....
0x00635f90 006d dd00 0000 0000 4141 4141 4141 4141 .m.....AAAAAAAA
0x00635fa0 1010 1010 1010 1010 1010 1010 1010 1010 .....
```

Decrypted command.

Tasks that the beacon can perform are:

- Change working directory
- Get current working directory
- Append/write to file
- Upload file to C2
- Execute command via popen
- Get disk partitions
- List files

The malware uses a separate thread to execute the tasks. The tasks are scheduled as jobs via a semaphore to ensure not too many jobs are executed at once. Vermilion Strike has a third way of communicating with the C2 server via ICMP ping messages. The malware adds the current pid to the offset **0x4** in the header and the encrypted payload is sent as data in the ICMP packet. The data size for an ICMP packet is limited to 65,507 bytes but the malware uses a size limit of 64,000 bytes for the payload. The code for sending and

processing ICMP messages exists in the malware but the code for enabling it via the configuration is not present. This means it has the capability but can't be configured to use it. This suggests it may be a new feature that hasn't been fully developed yet.

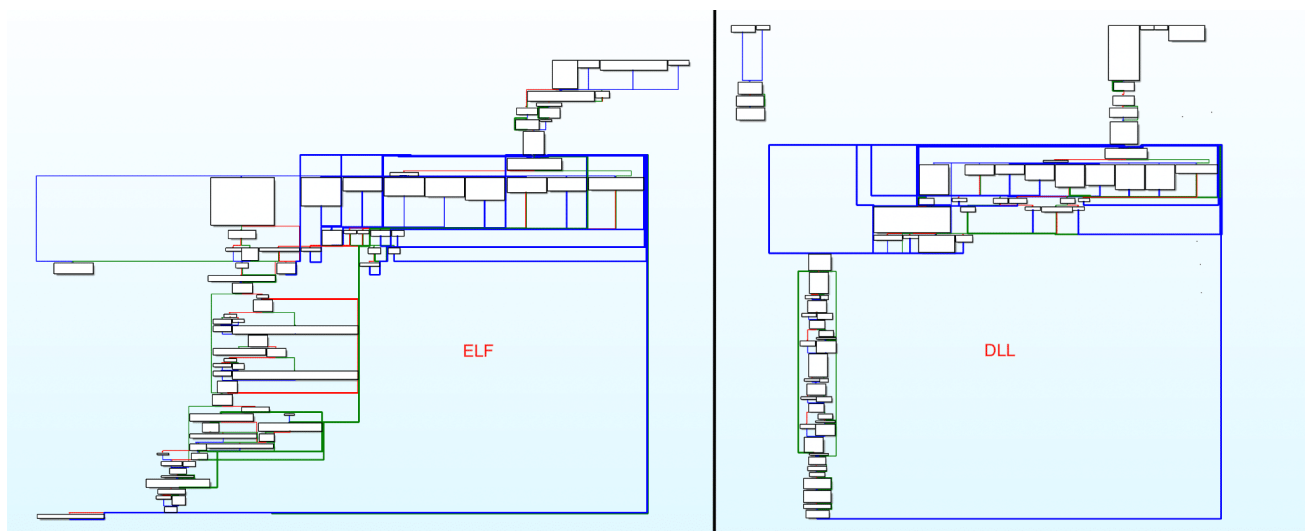
Links to Windows Files

When investigating this Linux file, we discovered related Windows samples. The first sample we noticed was:

3ad119d4f2f1d8ce3851181120a292f41189e4417ad20a6c86b6f45f6a9fbcfc. This is a 32-bit EXE sample that shares a C2 IP address (160.202.163[.]100). This is a stager that will fetch a DLL from the C2 over HTTP and execute it in-memory.

An example of the next stage DLL is

7129434afc1fec276525acfeee5bb08923ccd9b32269638a54c7b452f5493492. This sample, first noticed in 2019 by [Silas Cutler](#), is the Windows DLL equivalent of the ELF file. The functionality is almost exactly the same, except for the Windows environment. A side-by-side comparison of the configuration decoding function for the ELF and DLL beacons is shown below.



Configuration decryption function comparison.

The DLL has the same domains as the ELF for C2, as well as an additional configured domain “amazon.hksupd[.]com”.

Using the stager we managed to get a new payload from the server ([e40370f463b4a4feb2d515a3fb64af1573523f03917b2fd9e7a9d0a741ef89a5](#)). It has a lot of shared code with the sample from 2019. This sample and another Windows version of Vermilion Strike (**c49631db0b2e41125ccade68a0fe7fb70939315f1c580510e40e5b30ead868f5**) includes a similar version string as the ELF version. The version string in these samples is “W1.0.1”.


```

0x100025ec      8d442444      lea eax, [var_44h_18]
0x100025f0      e8ebf1ffff    call fcn.100017e0 ;[1]
0x100025f5      6a06          push 6 ; 6
0x100025f7      68b4f50210    push 0x1002f5b4 ; hit0_2 ; "W1.0.1"
0x100025fc      8d442444      lea eax, [var_44h_19]
0x10002600      e8dbf1ffff    call fcn.100017e0 ;[1]

```

Internal version string in recent Windows versions.

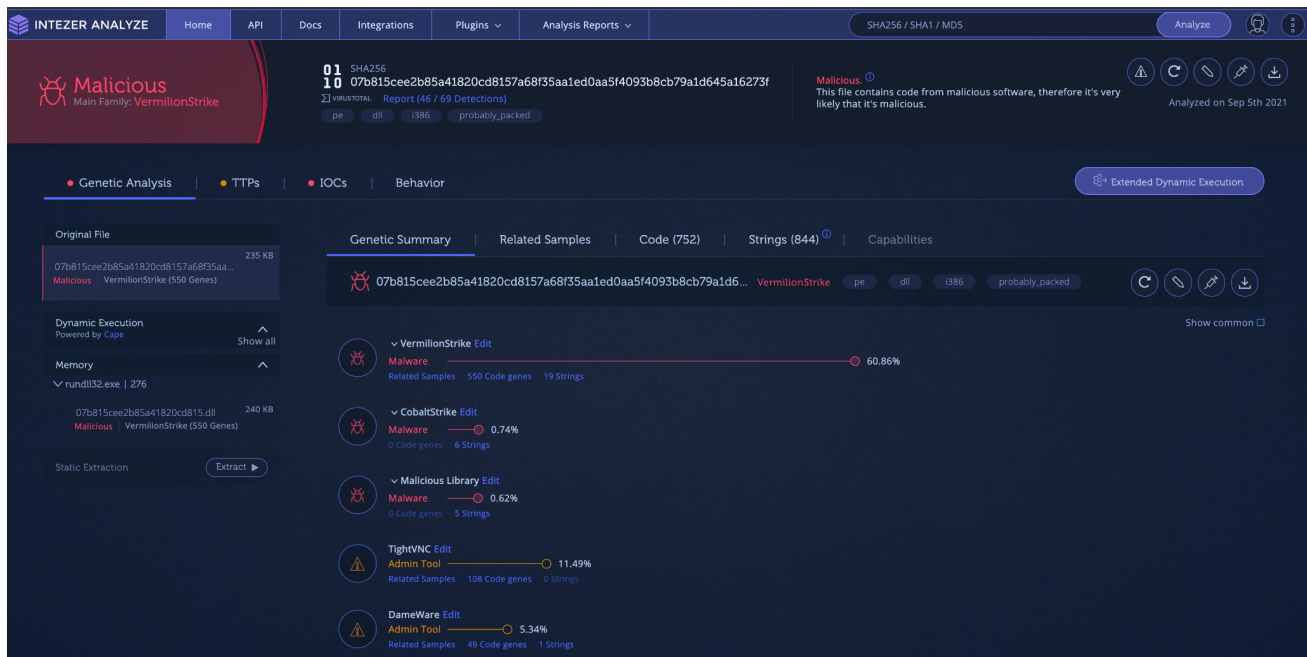
Conclusion

Vermilion Strike and other Linux threats remain a constant threat. The predominance of Linux servers in the cloud and its continued rise invites APTs to modify their toolsets in order to navigate the existing environment. Linux threats often have low detection rates compared to their Windows counterparts due to reasons discussed in [Why we Should be Paying More Attention to Linux Threats](#).

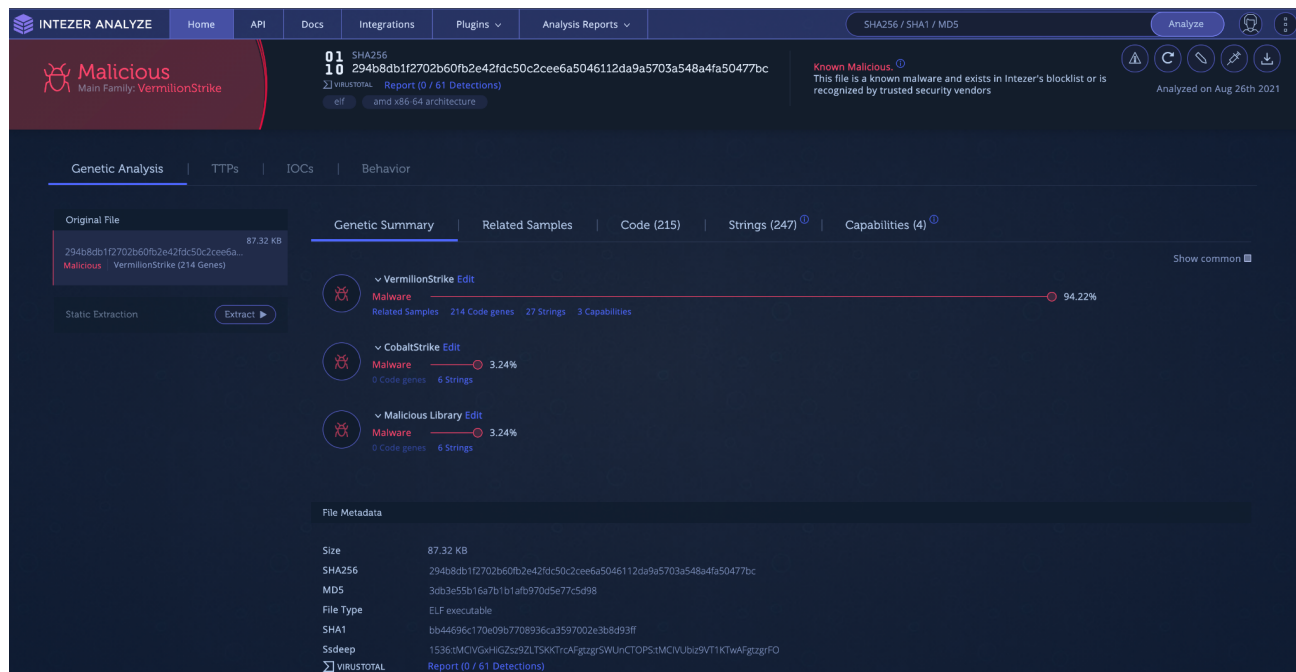
Vermilion Strike is not the only Linux port of Cobalt Strike's Beacon. Another example is the open-source project [gbeacon](#), a Go-based implementation. Vermilion Strike may not be the last Linux implementation of Beacon.

Detection and Response

Intezer Analyze can detect both Linux and Windows variants of Vermilion Strike, based on code reuse, TTPs, and strings. Shown below are the verdicts for both versions.



Intezer Analyze [verdict](#) of Windows version of Vermilion Strike.



Intezer Analyze [verdict](#) of Linux version of Vermilion Strike.

Detect if a Machine in Your Network Has Been Compromised

Get full runtime visibility over your code

For Linux-based systems, use [Intezer Protect](#) to get alerted on any malicious or unauthorized code executed in runtime. [Protect 10 hosts, nodes or machines for free](#)

For Windows-based systems, use the Intezer Analyze [Endpoint Scanner](#) to scan the entire memory of your machines to find any traces of malicious code running on them.

We also recommend using the IoCs section below to ensure that the Vermilion Strike process does not exist anywhere on your system.

Response

If you are a victim of this operation, take the following steps:

1. Kill the process and delete all files related to the malware.
2. Make sure that your machine is clean and running only trusted code using a runtime security platform like [Intezer Protect](#), or use Intezer Analyze [Endpoint Scanner](#) for Windows systems.
3. Make sure that your software is up-to-date with the latest versions and security patches and configured to security best practices.

IoCs

ELF

294b8db1f2702b60fb2e42fdc50c2cee6a5046112da9a5703a548a4fa50477bc

PE

Stager

3ad119d4f2f1d8ce3851181120a292f41189e4417ad20a6c86b6f45f6a9fbcfc

Beacon

7129434afc1fec276525acfeeee5bb08923ccd9b32269638a54c7b452f5493492
c49631db0b2e41125ccade68a0fe7fb70939315f1c580510e40e5b30ead868f5
07b815cee2b85a41820cd8157a68f35aa1ed0aa5f4093b8cb79a1d645a16273f
e40370f463b4a4feb2d515a3fb64af1573523f03917b2fd9e7a9d0a741ef89a5

C2

160.202.163.100

update.microsoft[hk[.]com

update.microsoft[kernel[.]com

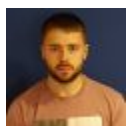
amazon.hksupd[.]com

Intezer would like to thank McAfee ATR for their help during the research process.



Avigayil Mechtinger

Avigayil is a product manager at Intezer, leading Intezer Analyze product lifecycle. Prior to this role, Avigayil was part of Intezer's research team and specialized in malware analysis and threat hunting. During her time at Intezer, she has uncovered and documented different malware targeting both Linux and Windows platforms.



Ryan Robinson

Ryan is a security researcher analyzing malware and scripts. Formerly, he was a researcher on Anomali's Threat Research Team.

**Joakim Kennedy**

Dr. Joakim Kennedy is a Security Researcher analyzing malware and tracking threat actors on a daily basis. For the last few years, Joakim has been researching malware written in Go. To make the analysis easier he has written the Go Reverse Engineering Toolkit (github.com/goretk), an open-source toolkit for analysis of Go binaries.