Opening “STEELCORGI”: A Sophisticated APT Swiss Army Knife

yoroi.company/research/opening-steelcorgi-a-sophisticated-apt-swiss-army-knife/

January 12, 2021

01/12/2021

Introduction

2020 was a really intense year in terms of APT activities, in fact it brought us new evidence of sophisticated campaigns targeting Enterprises organization across Europe and also Italy. In particular the threat group we track as TH-239, also mentioned as UNC1945 by FireEye security researchers, has been one of the sneakiest.

We discussed some of the new techniques and modus operandi used by this actor in our previous post, revealing how it leverages modern post exploitation tools even in legacy environments such as old Linux-based machines: with the help of a portable virtual machine, TH-239 is able to move part of its arsenal directly into the victim's internal network.

This time we decided to dissect and share intelligence information about another piece of the TH-239 arsenal: a tiny and mysterious tool dubbed “STEELCORGI” on FireEye research. This tool was heavily protected using a novel technique able to make things really difficult to any DFIR Team tackling with TH-239 intrusion, but it’s contents reveal huge surprises and unattended capabilities.

Technical Analysis

One of the most interesting components of the TH-239arsenal is an ELF binary file classified as “STEELCORGI”. The tool is presented in the form of an ELF named with the following md5: 0845835e18a3ed4057498250d30a11b1.

This binary is protected in a very aggressive way, let’s see how.

A Packed ELF

During the analysis we noticed that this ELF was very far from being readable, we extracted a series of elements confirming us that:

- High file dimension (more than 4MB);
- Obfuscated strings;
- Absence of Dynamic and (*.dynsym*) and Static Symbol Tables (*.symtab*);
- Absence of *section-headers* as Anti-reverse engineering Technique;
- High value of entropy > 7.9
- Runtime linking mechanism with *dlopen* and *dlsym*

As the first step, we focused on the static analysis of the sample in order to reconstruct the high level of sophistication and complexity of the packing. At first impact, strings are obfuscated, the binary is dynamically linked but the dynamic symbols table is empty.

Also, the absence of section-headers is an anti-reverse engineering technique adopted in this packer. Another indicator that the binary is packed is the high value of entropy *7.99*, as it is possible to observe in the following picture, on the right we have the whole portion of the ELF binary with compressed data.

Figure. High entropy section
At this point, we aren’t able to retrieve any other information about the packer, so we have to analyze the malicious routines aimed at unpack the sample. During the code inspection, a very long and complex subroutine emerges and it looks like the following screen:

Figure. Part of the decoding routines
It is a particular decoding routine instructed to decrypt some other protected code and strings. The code is a complex succession of logic instructions, like xor, shift, or etc. In the end of the decoding routine, the sample performs a check on the environment variables,

looking for a custom one installed by the TH-239 operators.

In fact, the environment variable “MCARCH_” contains the decryption key of the protector wrapper. When the malware retrieves the desidered environment variable, it starts the unpacking routine using the key stored in it and then starts the execution of the real payload.

This approach is a great evasion technique because it avoids the execution of the sample in any environments except the ones where TH-239 operators decide to get in.

Figure. Environment variable lookup

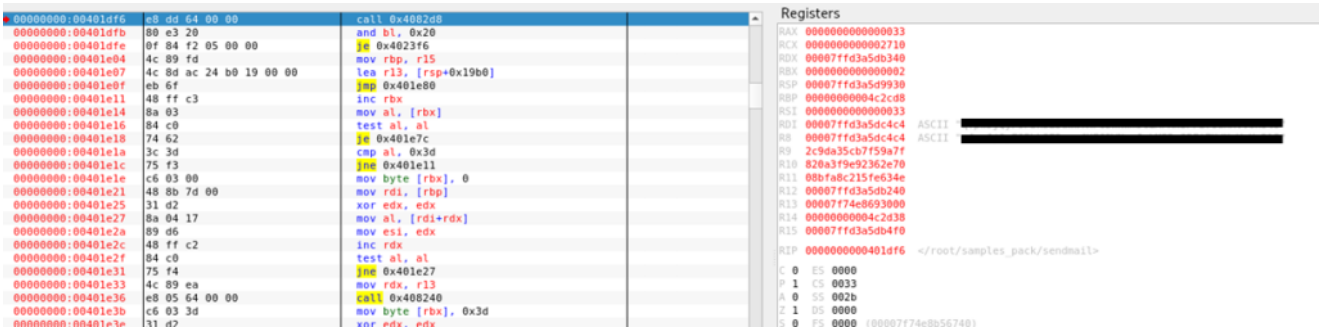

Figure. Environment variable match (redacted)

A Closer Look to the Stub

In addition, this packed ELF is matching some suspicious functions usually found in backdoors using the runtime linking techniques. Following are the functions with their relative offset:

Figure. Packed EFL imports

The presence of the *dlopen* and *dlsym* syscalls inside *libdl.so.2* is a clear indicator that this ELF uses a runtime linking mechanism by which hides all the dynamic symbols. The *dlopen()* function loads a shared object into the calling process’s address space (the same of *LoadLibrary()* in Windows). The symbol resolution is done by the *dlsym()* syscall which returns the address of the first occurrence of the symbol. Setting a breakpoint on *dlopen()* we are able to know which libraries are loaded at runtime:

Figure. Libraries dynamically loaded by the stub

Then, in the same way we dump all the symbol resolved at runtime with the *dlsym()* syscall:

Figure. Syscall invoked during the unpacking

Inspecting the new unpacked memory,we immediately noticed its structure with all the program headers and section headers, then we found all the loaded new segments mapped into Virtual Memory at specific offset:

Figure. Unpacked memory sections

These LOADsegments contain unpacked payload: it has different size than and the number of program-headers and section-headers are also different. The unpacked version have a lot of clear-text LOAD sections that was previously unpacked from memory, the following image summarize the unpacked memory regions (the bar on the right):

Figure. Segment difference
Inspecting all these unpacked regions (in red), we found some dictionaries used by the backdoor for enumeration or brute force. This is very interesting because it shows us the real capabilities and the magnitude of this Kill Chain. More details in the following sections.

Figure. Wordlists and dictionaries inside the ELF binary

The APT Swiss Army Knife

At this point of the analysis, we want to provide an overview of the capabilities of this malware sample. It is a complete toolset for reconnaissance, lateral movement, exploitation and post exploitation activities. When the toolset is launched, it shows the complete menu with all the possible commands.

Figure. Malware tool help
One of the sneakiest commands we noticed is the “bleach” one, able to delete all btmp wtmp and btmp logs. The btmp log keeps track of failed login attempts; wtmp gives historical data of utmp and btmp provides the complete picture of users logins at which terminals, logouts, system events and current status of the system, system boot time (used by uptime) etc. It is also able to clean Syslog logs in /var/log/syslog, /var/log/messages, /var/log/secure and /var/log/auth.log or optionally all of them with the “-A” flag (utmp+wtmp+lastlog+syslog) which is the default.

There is also the possibility to apply the so-called “Clean Filters” to clean logs for specific users or ip or according to date etc.

```
 clean (filters):  [-n <user>]    to filter by user    (can be set multiple times)
|                   [-t <tty>]     to filter by tty     (can be set multiple times)
|                   [-i <ip|host>] to filter by ip/host (can be set multiple times)
|                   [-p <pid>]     to filter by pid     (can be set multiple times)
|                   [-d <date>]    to filter by date    (can be set multiple times)
|                   [-g <str>]     to filter by string  (can be set multiple times
```

Is clear that the usage of the “bleach” parameter during an intrusion results in hard times for the DFIR team.

Figure. Bleach parameter execution
However the functionalities and tools embedded in this ELF binary are really wide and this is exactly why we referenced the tool as an APT swiss army knife. Here we sum up a list of the most interesting ones among the enlisting of all the available commands.

```
sendmail [ sun4me | demo | unixcat | nc110 | netcat | netcat-ssl | telnet |
traceroute | traceroute-tcp | traceroute-tcpfin | traceroute-udp | traceroute-icmp |
traceroute-all | sctpscan | sdporn | onesixtyone | snmpgrab | tftpd | ciscopush |
ciscown | ciscomg | HEAD | GET | ssleak | rmiexec | pogo | pogo2 | elogic | Cmd |
backfire | netbackup | netrider | sniff | bleach | nfsshell | mikrotik-client | sid-
force | ssh-user | sshock | ssh | arpmap | ricochet | mac2vendor | ip2country | ipgen
| ipsort | ipcalc | range2class | crunch | words.pl | passgen | passcheck | getpass |
decrypt-cisco | decrypt-vnc | decrypt-cvs | wmon | pmon | lemon | pty | exec | nsexec
| nsexec2 | setns | dumpkcore | dumpmem | pcregrep | xxd | strings | sstrip | shred |
md5sum | sha1sum | sha256sum | compress | uncompress | encrypt | decrypt | uuencode |
uudecode | base64 | whois | whob | resolv | ahost | adig | axfr | asrv | aspf |
periscope | scanip.sh | aliveips.sh | brutus.pl | enum4linux.pl | snmpcheck.pl | = |
_ | . | -? ] [options] [args]

sendmail [ s4m | demo | ucat | nc110 | nc | ncs | tel | tr | trt | trf | tru | tri |
tra | sctp | sd | sn | sg | tf | ccp | cco | ccg | HEAD | GET | ssleak | rmiexec |
pogo | pogo2 | el | Cmd | bf | nb | nr | sni | clean | nfs | mikro | sid | sshu | ss
| ssh | arp | rick | mac | ip2c | ipg | ips | ipc | r2c | crunch | words | lp |
pcheck | gpass | dec-cisco | dec-vnc | dec-cvs | wmon | pmon | emon | pty | exec |
nsexec | nsexec2 | setns | kcore | dmem | grep | xxd | str | strip | srm | md5 | sha1
| sha256 | comp | uncomp | enc | dec | uue | uud | b64 | whois | whob | res | host |
dig | axfr | asrv | aspf | scope | scanip | aliveips | brutus | e4l | snmpcheck | = |
_ | . | ? ] [options] [args]
```

The amount of available commands is simply impressive: some are known system utilities, some others are offensive scripts, other ones known hacking tools and other ones mysterious, custom commands.To sum up, we noticed at least four categories of tools embedded in this single ELF binary:

- **Network and Enumeration Tools** such asnetcat, unixcat, netcat-ssl, telnet, traceroute, traceroute-tcp, traceroute-tcpfin, traceroute-udp, traceroute-icmp | traceroute-all, tftpd, HEAD, GET, sniff, nfsshell, ssh, ricochet,axfr**,** ,whois, scanip, sctpscan, sdporn, rmiexec, arpmap, whois, who, ahost, resolv, adig, axfr, asrv, aspf, periscope, scanip.sh, aliveips.sh, brutus.pl, enum4linux.pl, mikro, ss, sshu, onesixtyone, snmpgrab, snmpcheck, ciscopush, mikrotik-client.
- **Anti-Forensics** tools such asbleach, clean.
- **System Utilities** such asmd5, sha1**,** mac2vendor, xxd, cmd, netbackup, ip2country, ipgen, ipsort, ipcalc, range2class, crunch, words.pl, passgen, passcheck, getpass, wmon, pmon, pty, exec, nsexec, nsexec2, setns, dumpkcore, dumpmem, pcregrep, strings, sstrip, shred, md5sum, sha1sum, sha256sum, compress, uncompress, encrypt, decrypt, uuencode , uudecode, base64.
- **Escalation and Exploitation** tools like ssleak, decrypt-vpn, pogo, pogo2, sid-force, sshock, decrypt-cisco, decrypt-vnc, decrypt-cvs.

There are tools for enumeration such as arp, dns, active directory, whois, ip enumeration and so on, some network tools and utilities for supporting exploiting and enumerations operations, also some exploitation and decryption tools specifically for CISCO, VNC, CVS and Mikrotik systems.

But some of them require a little deep dive.

SShock

SShock is a tool used to bruteforce SSH logins. In fact it is possible to specify an user list *(-u arg*) and a password list (*-p arg*), as shown in the following figure:

Figure. SShock help file
Another interesting thing of the tool is the possibility (with the *-E* flag) to specify some input file to upload and execute which will then be removed.

Lemon

Lemon is a very powerful monitoring utility which is capable of monitoring all system events such as (fork, exec, exit, core etc) of specific processes or users. All monitored events could be filtered with specific switches *(-p, -c, -u*). Below the tool's help menu is show:

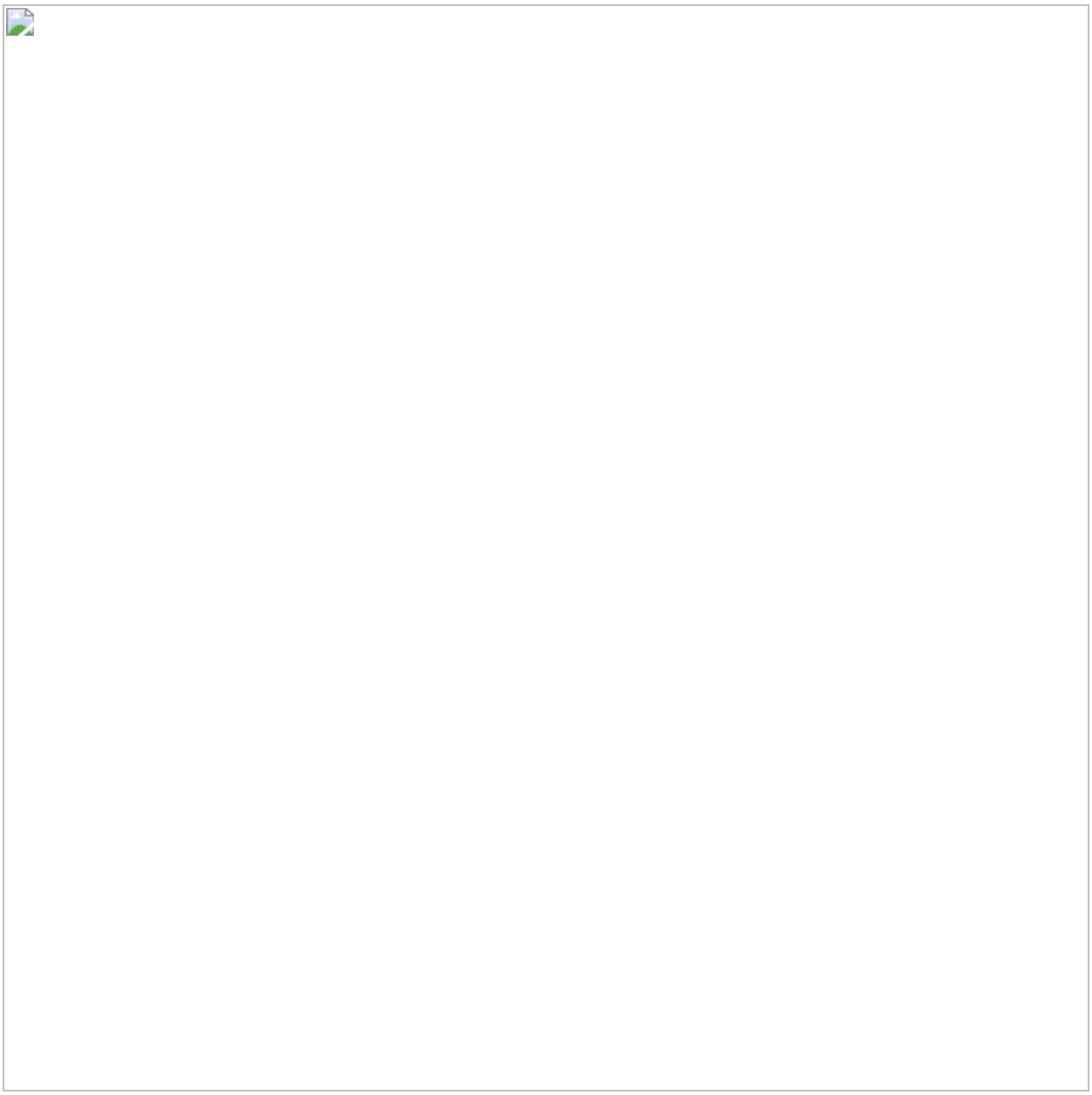

Figure. Lemon help file

For instance, it is possible to monitor all events related to specific user using the following switches lemon -u <username> -e all, in this case we monitor all system events related to kali user:

Figure. Lemon test run
Using this tool it is possible to monitor and track specific user’s activities on specific machines (or multiple machines) in order to spot the presence of specific users in some timeframe.

Ssleak

Ssleak is an utility to sniff SSL traffic. It is possible to specify a target and then dump all packets sent to and from in order to leak some information such as the server's certificate, server's canonical names etc.

Figure. SSLeak help file
Moreover it is also possible to exploit Heartbleed Vulnerability (CVE-2014-0160) with custom-forged heartbeat packets with a fake length with *-s* switch and print also the hexdump of such leak with *-x* switch.

Figure. SSLeak test run

Backfire

Backfire is a tool used to establish and manage connect-back (or reverse) shells. A reverse shell permits to establish a connection between the compromised host (pivot) and the target machine when the target machine is not directly accessible for several reasons. For instance to perform maintenance tasks on hosts behind firewalls or NAT.

As, shown in the following screen, *backfire* provides the execution of such commands (*-c commands*) through a connect-back connection that is possible to spawn with -S flag or with -s <commands>

Figure. Backfire help file

Ricochet

Ricochet is a powerful utility for packet spoofing and FW ACL assessment. The tool can act as a client or a server. The client version permits to forge IP-PROTO/ICMP/UDP/TCP packets in order to test fw ACLs while the server is used to listen for replies coming from the firewall. It is possible to use 2 different methods. One is called *spoof (method #1) to spoof packets* and the other is *rick (method#2*) which stands for "ricochet" used also to spoof the address and port of the outgoing requests:

Figure. Ricochet help file

Conclusion

The versatility of the "STEELCORGI" tool used by TH-239 is really impressive: all such capabilities embedded in a single, standalone, ready to deploy binary file, potentially enabling the attacker to establish a hidden communication channel, to recon internal network and to step in remote endpoint abusing various techniques. Also, this sort of "swiss army knife" was also heavily protected in a way that could be activated only during an actual intrusion, because the activation key is inoculated into the compromises system directly by the malicious operators, at run time.

All these facts are reminding us how dangerous and slimy an advanced intruder could sneak into the company network: tackling such kinds of threats requires advanced intelligence and analysis capabilities.

Appendix

Indicator of Compromise

Hash:

0845835e18a3ed4057498250d30a11b1

Yara:

```
rule ELF_packed_STEELCORGI_backdoor_UNC1945{
 meta:
   description = "Yara Rule for packed ELF backdoor of UNC1945"
   author = "Yoroi Malware Zlab"
   last_updated = "2020_12_21"
   tlp = "white"

   category = "informational"

strings:

$s1={4? 88 47 3c c1 6c ?4 34 08 8a 54 ?? ?? 4? 88 57 3d c1 6c}
$s2={0f b6 5? ?? 0f b6 4? ?? 4? c1 e2 18 4? c1 e0 10 4? }
$s3={8a 03 84 c0 74 ?? 3c 3d 75 ?? 3c 3d 75 ?? c6 03 00 4? 8b 7d 00}
$s4={01 c6 89 44 ?? ?? 8b 44 ?? ?? 31 f2 89 74 ?? ?? c1}
$s5={ 4? 89 d8 4? 31 f2 4? c1 e0 13 4? 01 d7 4? }

condition:
   uint32(0) == 0x464c457f and 3 of them
}


rule ELF_unpacked_STEELCORGI_backdoor_UNC1945{
 meta:
   description = "Yara Rule for unpacked ELF backdoor of UNC1945"
   author = "Yoroi Malware Zlab"
   last_updated = "2020_12_21"
   tlp = "white"
   category = "informational"

strings:
$s1="MCARC"
$s2="833fc0088ea41bc3331db60ae2.debug"
$s3="PORA1022"
$s4="server"
$s5="test"
$s6="no ejecutar git-update-server-info"
$s7="dlopen"
$s8="dlsym"
$s9="5d5c6da19e62263f67ca63f8bedeb6.debug"
$s10={72 69 6E 74 20 22 5B 56 5D 20 41 74 74 65 6D 70 74 69 6E 67 20 74 6F 20 67 65
74 20 4F 53 20 69 6E 66 6F 20 77 69 74 68 20 63 6F 6D 6D 61 6E 64 3A 20 24 63 6F 6D
6D 61 6E 64 5C 6E 22 20 69 66 20 24 76 65 72 62 6F 73 65 3B}

condition:
 all of them and #s4>50 and #s5>20
}
```

This blog post was authored by Luigi Martire, Antonio Pirozzi and Luca Mella of Yoroi Malware ZLAB