Exercises
Malware

2. [2 points] A new malware just broke out, causing a world-wide infection and a huge amount of damages. Unfortunately, all the anti-malware systems are not able to detect this malware. You were able to retrieve a couple of samples.

Consider the code snippets reported below, extracted from the two malware samples you retrieved:

Sample 1

```
 1 pop  ebx
 2 lea ecx, [ebx + 42h]
 3 push ecx
 4 push eax
 5 push eax
 6 sdt [esp - 02h]
 7 pop ebx
 8 add ebx, 1Ch
 9 cli
10 mov ebp, [ebx]
```

Sample 2

```
 1 pop  ebx
 2 lea ecx, [ebx + 42h]
 3 push ecx
 4 push eax
 5 nop
 6 push eax
 7 inc eax
 8 sdt [esp - 02h]
 9 dec eax
10 pop ebx
11 add ebx, 1Ch
12 cli
13 mov ebp, [ebx]
```

It is clear that the malware is showing evasive behavior. What technique is implemented? How this technique works?

It is clear that the malware is showing evasive behavior. What technique is implemented? How this technique works?

Metamorphism. See slides.

2. [2 points] A new malware just broke out, causing a world-wide infection and a huge amount of damages. Unfortunately, all the anti-malware systems are not able to detect this malware. You were able to retrieve a couple of samples.

We made a binary diff of the two samples in order to evaluate the difference in their layout and reported only the differences here:

Sample 1

```
0000000000000675 <decrypt>:
                        [...]
 6a3:   83 f1 42                xor    ecx,0x42
                        [...]
00000000000007b0 <payload>:
 7b0:   28 00                   sub    BYTE PTR
[rax],al
 7b2:   1a bc 86 0a db 10 0a    sbb    bh,BYTE PTR
[rsi+rax*4+0xa10db0a]
 7b9:   fd                      std
 7ba:   6d                      ins    DWORD PTR
es:[rdi],dx
 7bb:   20 2b                   and    BYTE PTR
[rbx],ch
 7bd:   2c 6d                   sub    al,0x6d
 7bf:   6d                      ins    DWORD PTR
es:[rdi],dx
 7c0:   31 2a                   xor    DWORD PTR
[rdx],ebp
 7c2:   15 16 1c 0b cb          adc    eax,0xcb0b1c16
 7c7:   92                      xchg   edx,eax
 7c8:   0b cb                   or     ecx,ebx
 7ca:   90                      nop
 7cb:   4d                      rex.WRB
 7cc:   47                      rex.RXB
```

Sample 2

```
0000000000000675 <decrypt>:
                        [...]
 6a3:   83 f1 42                xor    ecx,0x12
                        [...]
00000000000007b0 <payload>:
 7b0:   78 50                   js     802
<__GNU_EH_FRAME_HDR+0x32>
 7b2:   4a ec                   rex.WX in al,dx
 7b4:   d6                      (bad)
 7b5:   5a                      pop    rdx
 7b6:   8b 40 5a                mov    eax,DWORD PTR
[rax+0x5a]
 7b9:   ad                      lods   eax,DWORD PTR
ds:[rsi]
 7ba:   3d 70 7b 7c 3d          cmp    eax,0x3d7c7b70
 7bf:   3d 61 7a 45 46          cmp    eax,0x46457a61
 7c4:   4c 5b                   rex.WR pop rbx
 7c6:   9b                      fwait
 7c7:   c2 5b 9b                ret    0x9b5b
 7ca:   c0                      .byte 0xc0
 7cb:   1d                      .byte 0x1d
 7cc:   17                      (bad)
```

Additionally, we executed the samples and collected the syscalls in order to evaluate their behaviour (i.e., semantics).

Sample 1 Trace

```
execve("./sample1", ["./sample1"], 0x7ffe4fc39080 /* 60
vars */) = 0
brk(NULL = 0x558154aa3000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No
such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) =
3
fstat(3, {st_mode=S_IFREG|0644, st_size=232207, ...}) = 0
mmap(NULL, 232207, PROT_READ, MAP_PRIVATE, 3, 0) =
0x7f0157a64000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC)
= 3
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\2001\2\0\0
\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2105608, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0157a62000
                        [...]
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f0157a634c0) = 0
mprotect(0x7f015786f000, 16384, PROT_READ) = 0
                      [...]
munmap(0x7f0157a64000, 232207)          = 0
mmap(NULL, 4, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_SHARED|MAP_ANONYMOUS, -1, 0) = 0x7f0157a9c000
execveat(1852400175, "/bin//sh", NULL, NULL, 0) =0
```

Sample 2 Trace

```
execve("./sample2", ["./sample2"], 0x7ffe4fc39080 /* 60
vars */) = 0
brk(NULL) = 0x5575cc15d000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No
such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) =
3
fstat(3, {st_mode=S_IFREG|0644, st_size=232207, ...}) = 0
mmap(NULL, 232207, PROT_READ, MAP_PRIVATE, 3, 0) =
0x7fbb4f1f2000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC)
= 3
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\2001\2\0\0
\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2105608, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbb4f1f0000
                        [...]
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fbb4f1f14c0) = 0
mprotect(0x7fbb4effd000, 16384, PROT_READ) = 0
                      [...]
munmap(0x7fbb4f1f2000, 232207)          = 0
mmap(NULL, 4, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_SHARED|MAP_ANONYMOUS, -1, 0) = 0x7fbb4f22a000
execveat(1852400175, "/bin//sh", NULL, NULL, 0) =0
```

It is clear that the malware is showing evasive behavior. What technique is implemented? How this technique works?

It is clear that the malware is showing evasive behavior. What technique is implemented? How this technique works?

Polymorphism

In order to avoid signature detection, a malware sample saves his own assembly code in text format on the victim machine, and then uses a standard assembler to generate and execute the real malicious object code on the machine.

2. [2 points] How can a signature-based detection method (e.g., antivirus) detect this kind of malware ?

In order to avoid the previous signature detection, a malware sample saves his own assembly code in text format on the victim machine, and then uses a standard assembler to generate and execute the real malicious object code on the machine.

2. [2 points] How can a signature-based detection method (e.g., antivirus) detect this kind of malware ?

Have a signature to detect the malware in its "textual" assembly format. Note that having a signature to detect the assembler is a wrong solution, as it leads to lots of false positives (the system assembler it's a legitimate program, after all!)

3. [1 point] You suspect that your machine have been compromised with a kernel rootkit. You tried to use network traffic tools from your machine but you do not see any malicious traffic. Can you conclude that your machine is safe? If is not there are other way to prove you have been compromised?

3. [1 point] You suspect that your machine have been compromised with a kernel rootkit. You tried to use network traffic tools from your machine but you do not see any malicious traffic. Can you conclude that your machine is safe? If is not there are other way to prove you have been compromised?

No you cannot conclude that the machine have not been compromised. Because the malware can hide its own traffic from tools running on the compromised machine. You could inspect network traffic using an external machine as a MitM between your machine and the router.

4. [1 point] A colleague suggests to replace the hard drive of a machine to be sure to get rid of a very sophisticated rootkit. However, after reinstalling the operating system, it seems like that the machine is infected by the same rootkit. Provide an explanation of what happened. Whatever your answer is, explain why.

4. [1 point] A colleague suggests to replace the hard drive of a machine to be sure to get rid of a very sophisticated rootkit. However, after reinstalling the operating system, it seems like that the machine is infected by the same rootkit. Provide an explanation of what happened. Whatever your answer is, explain why.

If it is a BIOS rootkit then No. If it is a kernel rootkin it is ok to just replace the HD or even just reinstall the OS.