

# Reverse Engineering of Malware

Matej Kašťák

Brno University of Technology, Faculty of Information Technology  
Božetěchova 1/2. 612 66 Brno - Královo Pole  
[login@fit.vutbr.cz](mailto:login@fit.vutbr.cz)



April 10, 2021

- Introduce **reverse engineering** and basic concepts
- **Showcase** tools used for reverse engineering
- Use the tools on a real malware sample (Demo)

- Analyze foreign unknown files
- Extract **signatures** that can be used for blocking malware
- Gain knowledge how to better **protect** ourselves

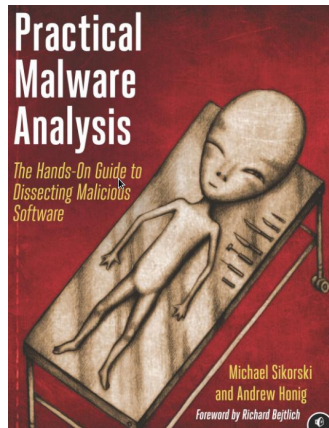


Figure: Practical Malware Analysis



Figure: Common depiction of a hacker :)

- Specialized and complex tools
- Require good low level understanding of systems
- Many of the tools are really really expensive



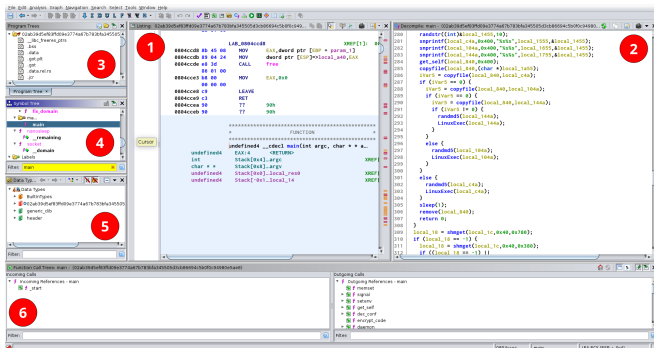


Figure: Ghidra user's interface.

- Open source competitor to IDA pro
- Sometimes lacking functionality

- Debuggers
  - WinDbg / OllyDbg
  - GDB (stace, ltrace)
- Sandboxes
  - Cuckoo
- Symbolic execution
  - Angr

```

GDB dashboard

Output/messages
17 for (i = 0; i < text_length; i++) {
Assembly
0x0000555555551ec 48 8b 45 f8 encrypt+103 mov rax,QWORD PTR [rbp-0x8]
0x0000555555551f0 48 01 db encrypt+107 add rax,rdx
0x0000555555551f3 31 ce encrypt+110 var esi,ecx
0x0000555555551f5 89 f2 encrypt+112 mov edi,esi
0x0000555555551f7 8b 10 encrypt+114 mov ebx,QWORD PTR [rax],0x1
0x0000555555551f9 48 83 45 f8 01 encrypt+116 add QWORD PTR [rbp-0x8],rax
0x0000555555551fa 48 8b 45 f8 encrypt+121 mov rax,QWORD PTR [rbp-0x8]
0x000055555555202 48 3b 45 e8 encrypt+125 cmp rax,QWORD PTR [rbp-0x18]
0x000055555555206 72 b6 encrypt+129 jb 0x555555551c3 <encrypt+62>
0x000055555555208 90 encrypt+131 nop

Breakpoints
[1] break at 0x0000555555552d9 in xor.c:56 for xor.c:56 hit 1 time
[2] break at 0x000055555555199 in xor.c:13 for encrypt hit 1 time
[3] break at 0x000055555555521b in xor.c:27 for dump if i = 5
[4] write watch for output[10] hit 1 time

Expressions
password[1 % password_length] = 101 'e'
text[i] = 32 ' '
output[i] = 69 'E'

History
$$1 = 0x555555550260 "\f(032\va\006)022)004)032)001)037E": 12 'f'
$0 = 0x7fffffffef2c "hunter2": 104 'h'

Memory
password
0x00007fffffffef2c 68 75 6e 74 65 72 32 00 64 6f 65 73 6e 74 20 6c hunter2-doesnt-1
text
0x00007fffffffef34 64 6f 65 73 6e 74 20 6c 6f 6f 6b 20 6c 69 6b 65 doesn't-look-like
0x00007fffffffef44 20 73 74 61 72 73 20 74 6f 20 6d 65 00 48 4f 53 stars-to-me-H0S

Output
0x0000555555550260 0c 1a 0b 07 0b 0b 12 04 1a 01 1f 45 00 00 00 00 .....E....
0x0000555555550270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Registers
rax 0x0000555555550260 rbx 0x0000000000000000 rcx 0x0000000000000065
rdx 0x0000000000000045 rsi 0x0000000000000045 rdi 0x00007fffffffef40
rbp 0x00007fffffffefc0 rsp 0x00007fffffffefc0 r8 0x0000000000000003
r9 0x0000000000000330 r10 0x0000555555550010 r11 0x0000000000000030
r12 0x0000555555550a0b r13 0x00007fffffffefc0 r14 0x0000000000000000
r15 0x0000000000000000 r16 0x0000555555551f9 rflags [ IF ]
cs 0x00000033 ss 0x0000002b ds 0x00000000
es 0x00000000 fs 0x00000000 gs 0x00000000
    
```

Figure: Example GDB output. Taken from <https://github.com/cyrus-and/gdb-dashboard>





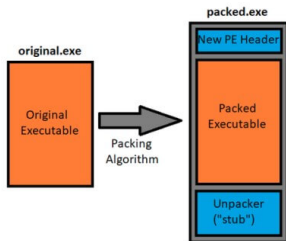


Figure: Packing diagram. Taken from <https://www.arriidae.com/blogs/Packed-Malware.php>

- Goal is to avoid detection
- Original binary is embedded into a new one
- New binary decodes and executes the original code
- Examples
  - UPX
  - Armadillo

Demo

- Introduced basic concepts and tools used for reverse engineering
- Analyzed a malware sample using Ghidra
- Created program that is able to reverse the encryption schema