REVERSE ENGINEERING

@SLEEPUNDERFLOW

TOOLS

- Objdump
- Radare2
- GDB
- Hopper
- Cutter
- Many more

GDB

Tons of functions
Only a few useful
Use GEF to extend capabilities
Disassembly + debugging

gdb ./binary or gdb –p PID

attach PID

break *main
break *main+58
delete breakpoint 1

set disassembly-flavor intel
disassemble main

run
continue
stepi

RADARE2

Disassembly + debugging + patching
Never write and debug at the same time!

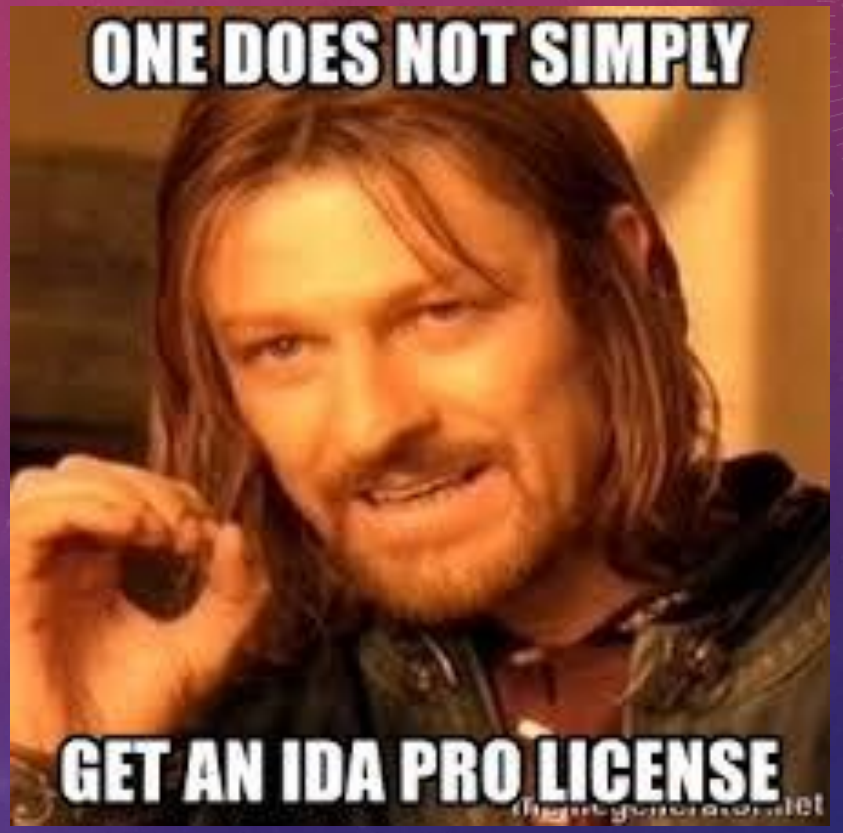
r2 binary
r2 –w binary
r2 –d binary

s sym.main
s +1
s -1

aaa
pdf
pd 20
afl

IDA

- For pros
- Expensive



GHIDRA

- The best one
- Does not support debugging yet

MAIN HINTS

- Do not try to understand the binary in the first go line by line
- Understand what it does on the high level
- Look for patterns:
  - If statements
  - Jump tables (switch)
  - Loops
  - Function calls

VARIABLES

- Local - [RBP-xxx]
- Function arguments - [RBP+xxx]
- Global - [RIP+/-xxx], [xxxx]
- Pointers - LEA RAX, [RAX]; MOV RAX, [RAX]

LOOPS

- 0x000005fe mov dword [rbp - 8], 0
  0x00000605 mov dword [rbp - 4], 0
  0x0000060c jmp 0x616
  0x0000060e add dword [rbp - 8], 1
  0x00000612 add dword [rbp - 4], 1
  0x00000616 cmp dword [rbp - 4], 0x63
  0x0000061a jle 0x60e

IF STATEMENTS

- mov dword [rbp - 4], 0
  cmp dword [rbp - 4], 0xa ; [0xa:4]=0
  jle 0x611
  add dword [rbp - 4], 1
  jmp 0x624
  cmp dword [rbp - 4], 0
  jg 0x61d
  sub dword [rbp - 4], 1
  jmp 0x624
  mov dword [rbp - 4], 5
  mov eax, 0

JUMP TABLES

- mov eax, eax
  lea rdx, qword [rax*4]
  lea rax, qword [0x000020e8]
  mov eax, dword [rdx + rax]
  movsxd rdx, eax
  lea rax, qword [0x000020e8]
  add rax, rdx
  jmp rax

CHALLENGE

- curl https://raw.githubusercontent.com/CUEH-ComSec/Presentations/master/generate.sh | sh