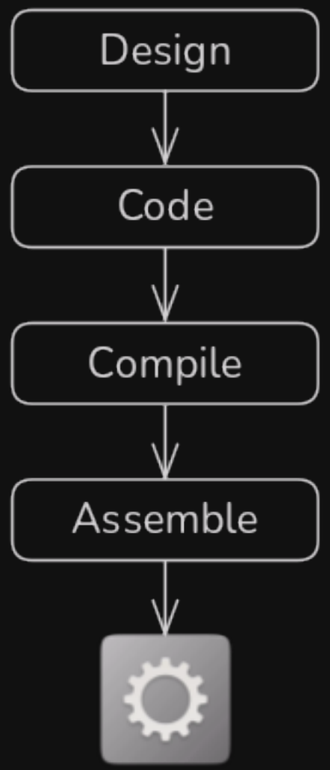
Reverse Engineering

Intro

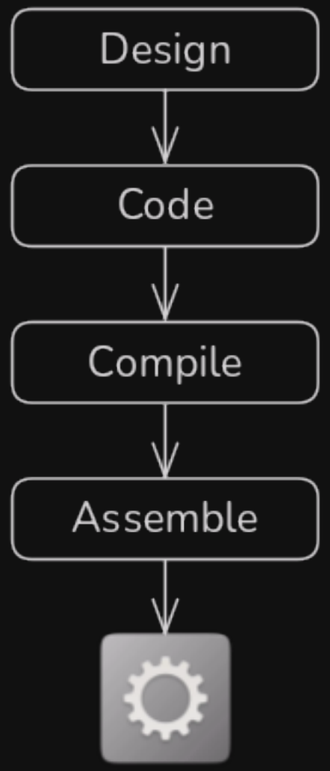
Created by lk0ri4n, 2<3

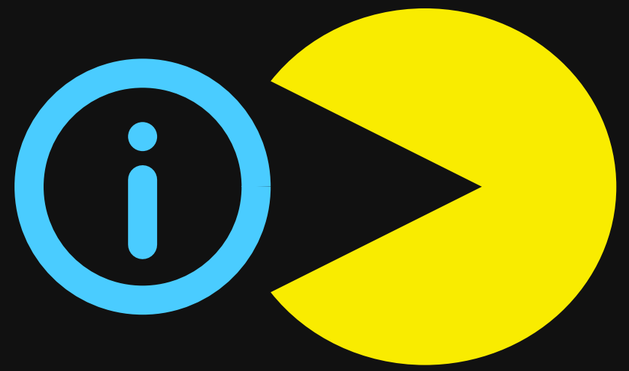
02.05.2024

Forward engineering process

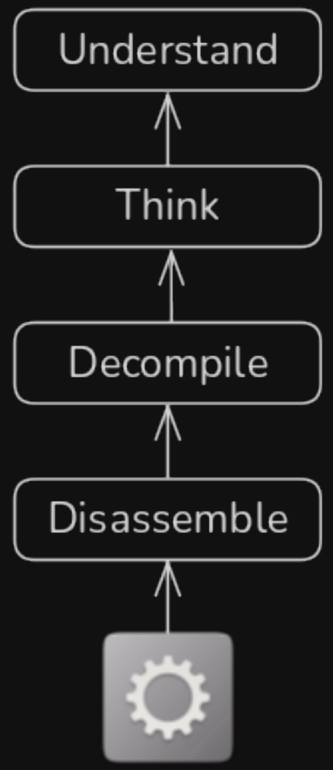


Forward engineering process





Reverse engineering process



Why would I need that?

Why would I need that?

- CTF

Why would I need that?

- CTF
- Security analysis
- Malware analysis
- No docs, source available
- Modding, Cracking

Why would I need that?

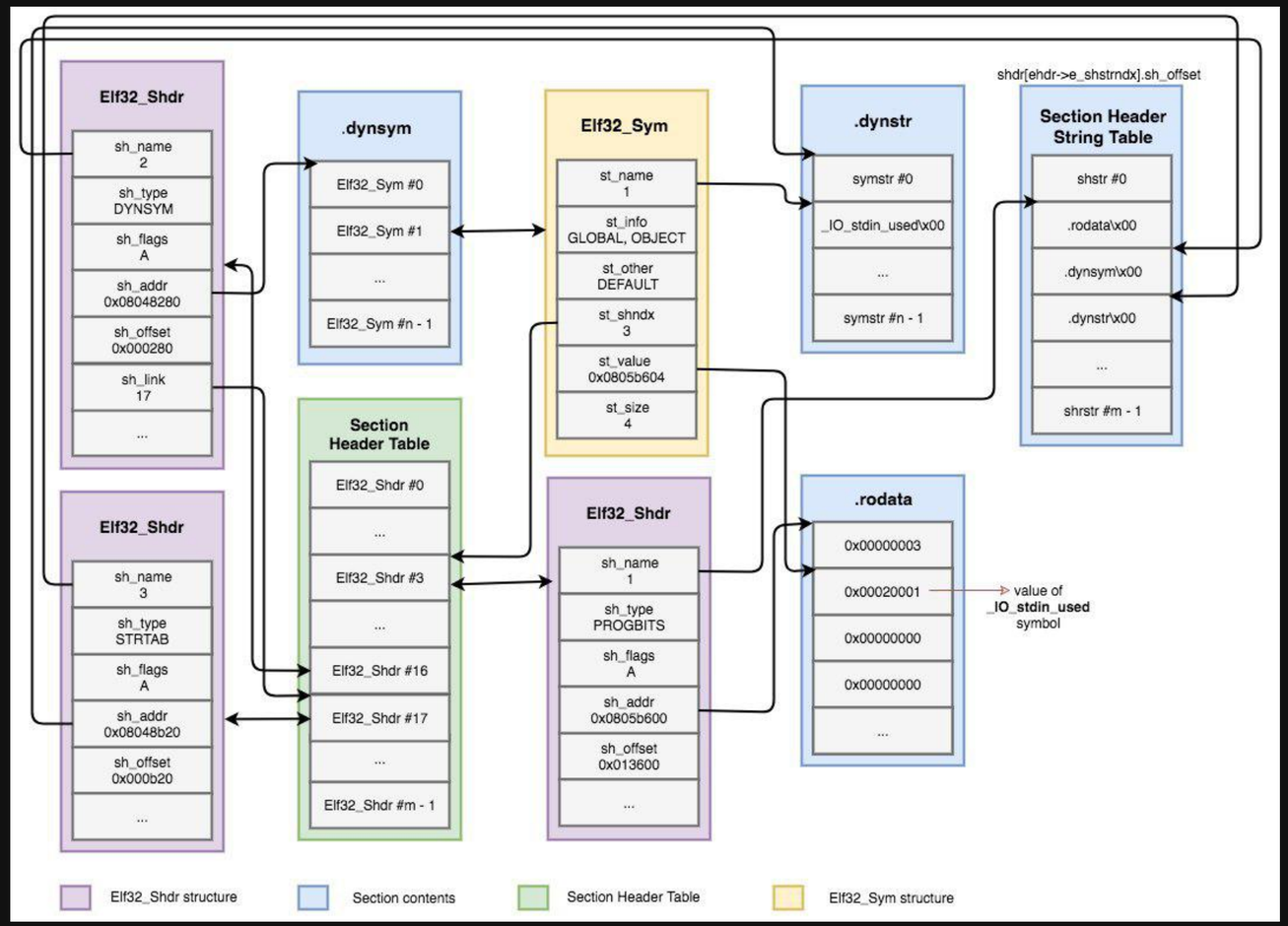
- CTF
- Security analysis
- Malware analysis
- No docs, source available
- Modding, Cracking

...plus it's fun!

What are we dealing with?

```
$ file chal
chal: ELF 64-bit LSB pie executable,
    x86-64,
    version 1 (SYSV),
    dynamically linked,
    interpreter /lib64/ld-linux-x86-64.so.2,
    BuildID[sha1]=e7f3e971abeb24c4d7cc7747b3274f3058e749af,
    for GNU/Linux 3.2.0,
    stripped
```

ELF Structure



Source: intezer, ELF 101. Part 2: Symbols

Important ELF sections

- .text: executable code of the program
- .plt & .got: used to resolve and dispatch library calls
- .data: pre-initialized global writable data
- .rodata: pre-initialized global read-only data
- .bss: uninitialized global writable data

Useful tools

- readelf to parse the ELF header
- objdump to parse the ELF header and disassemble the source code
- nm to view your ELF's symbols
- patchelf to change some ELF properties
- objcopy to swap out ELF sections
- strip to remove otherwise-helpful information (such as symbols)

Static analysis tools

- file type infos based on magic bytes
- binwalk identify & opt. extract embedded files and data
- strings dumps strings found in file
- objdump simple disassembler
- checksec check security features

x86 Opcode & Instruction References:

- coder64 reference, raw byte format
- Felix Cloutier, web adaptation of intel manual

Decompilers

Open source:

- Ghidra reverse engineering tool created by NSA
- angr management academic binary analysis framework
- cutter reverse engineering tool powered by Rizin

Commercial:

- Binary Ninja sleek, affordable IDA competitor (free cloud version)
- IDA pro "gold standard" of disassemblers (expensive)

Demo time

Demo time

Talk: Advanced Ghidra (useful extensions, tricks)

Rev player trust issues

Tool output is not always perfect!

- `file` checks *known* magic bytes (first match)
- Decompilers make (wrong) assumptions all the time!
- Tool output may differ (different strengths)

Rev player trust issues

Tool output is not always perfect!

- `file` checks *known* magic bytes (first match)
- Decompilers make (wrong) assumptions all the time!
- Tool output may differ (different strengths)

Know your tools!

Practice time

Let's do some reversing: intro.kitctf.de!

Dynamic approach

Debugging with gdb

- `gdb -ex 'set disassembly-flavor intel' chal`
- pwndbg: community-powered extension (lots of features)

  Ideally put such settings into .gdbinit

Overview

Function	Meaning
help	Print list of commands and specific help
pwndbg	Print list of pwndbg commands
run args	Run the program
starti args	Run the program and break on first instruction
break expr	Break at the given address or symbol
watch expr	Break when a value is written to the given address
rwatch expr	Break when a value is read from the given address
continue	Continue program execution
si and ni	Step into and step over

Examine Memory

`x/<amount><format><size> <expr>`

Parameter	Meaning
amount	Number of things to read
format	Output format, notably x, a, s for hex, addresses, and strings
size	Size of the data blocks, b, h, w, g for 1, 2, 4, 8 bytes respectively
expr	C-like expression describing data location

telescope [addr] [count] Recursively dereference pointers (e.g., stack overview)

Dynamic analysis tools

- strace trace system calls
- ltrace trace library calls
- gdb GNU debugger
- Emulators

Timeless debugging:

- gdb has built-in record-and-replay functionality
- rr highly-performant record-replay engine
- qira timeless debugger made for reverse engineering

Further reading

Processor ISA Manuals

Gdb and Pwndbg documentation

Ghidra Book

ost2.fyi

Other helpful tools

- angr symbolic execution
- SMT solvers (e.g., z3)
- SageMath (ask our crypto players 😉)

Lots plugins and tools for specific use cases

And... Action!

Start playing at intro.kitctf.de