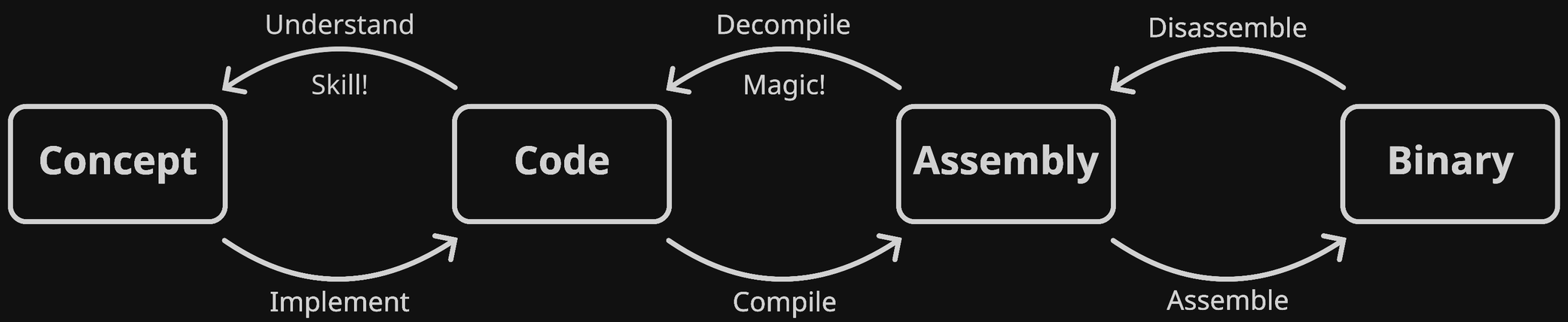
Reverse Engineering

Intro

By Ik0ri4n, with help from MisterPine

21.11.2025

What are we doing here?



Why would I need that?

- CTF
- Vulnerability research
- Malware analysis
- No docs, source available
- Modding, Cracking

...plus it's fun!

Common tools

- file type infos based on magic bytes; Good first step for every challenge
- binwalk identify & opt. extract embedded files and data
- strings dumps strings found in file (sometimes there's a flag in there :P)

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 and cloud version)
- IDA pro "gold standard" of disassemblers (expensive)

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)

Never trust a tool you haven't written yourself!

Especially don't trust one you have!!!

Dynamic approach

Debugging with ~~gdb~~ pwndbg

We don't use plain gdb here!

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)

Automated debugging

- gdb Command Files: run scripts with gdb commands
- pwntools: lots of functionality for scripting (see `pwnlib.gdb`)
- libdebug: simple API to debug programmatically

Dynamic analysis tools

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

Other helpful tools

- angr symbolic execution
- SMT solvers (e.g.,z3)

Lots of plugins and tools for specific use cases

Learning Resources

pwn.college lecture-style modules with challenges:

- Rev intro with detailed information on structure and execution
- Advanced rev with lots of challenges similar to ours
- GDB Refresher and many more modules...

Decompilers:

- Binary ninja's detailed user manual
- Ghidra video series by stacksmashing

And… Action!

Start playing at intro.kitctf.de

Übung macht den Meister!