Radare2

sudo r2 -d 1

What is Radare?

Framework for reverse-engineering and binary analysis

Consists of many tools

Began as scriptable commandline hex editor

Now capable of many things (*)

Parts of the radare Project

rabin2 - Binary information

radiff2 - Binary Diff

rasm2 - Dis/assembler

ragg2 - Compiler

More parts of the radare Project

rax2 - Base converter (and more)

rahash2 - Calculate all the hashes

rarun2 - Run application in special context

r2 - Interactive shell application of everything else

radare2 - Symlink to r2

Why even Radare2?

Free Software!

GDB can’t do fancy shit. (GDB-PEDA is still not as good)

Masochism! (Documentation and syntax)

Fancy console graphics!

Bugs!

Basics in reverse engineering

2 principles of analysis:

Static analysis (staring at assembler code)

Dynamic analysis (debugging)

Basics in reverse engineering

Why re your own code:

- analyze compiler output
- find compiler bugs
- general debugging

(gdb should be better in most of these scenarios)

Basics in reverse engineering

Why re code of others:

- Find hidden things (e.g. hardcoded passwords)
- Modify executables (e.g. remove copy protection)
- Learn application layout (e.g. for building exploits)
- To reimplement application (e.g. nouveau driver)

Assembly 101

The language of the (x86) machine

Assembly knowledge

Default: intel assembler syntax

<op> <dest>,<src>

mov eax, 2;	Copies the value 2 in eax
add eax, 8;	Adds 8 to eax and stores the result in eax
call 0xdead;	Calls function at 0xdead

Assembly commands

How many: TONS. Also: Architecture specific

Calculations: add, sub, mul, div, inc, dec, cmp

Jumps: jmp, jz, jnz, ja, jb, j{something}, call, ret

Copy: mov, lea

Stack operations: push, pop

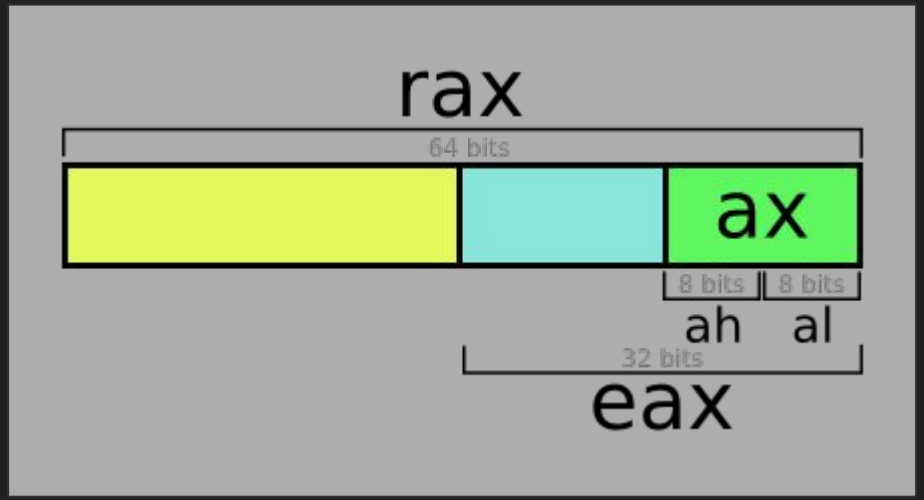
Registers, Memory and Stack

Registers:

- Fastest memory in CPU
- some are “Variables” some are special (ebp, esp, eip, *)
- can be r+w, ro, wo

General purpose registers:

eax, ebx, ecx, edx



Registers, Memory and Stack

Memory:

```
mov eax, dword [0xdead];Copy from 0xdead into eax
mov ebx, dword [eax];          Copy from address into ebx
mov dword [0xdead], ebx;Write from ebx to memory
lea eax, [ebp - 0x10];         Copy pointer to [...] into eax
```

Endianess

How is multi byte information stored

Big endian: highest byte first, lowest last

Little endian: lowest byte first, highest last

x86 is little endian

Little endian

Example:

mov dword [0x1000], 0x12345678;

becomes:

0x1000: 0x78563412

in memory

Back to Radare

GUI?

Build for command line use

Has console based visual mode

There is a (experimental) GUI named “cutter”

Pro: Fast nice overview

Con: Compared to this r2 is bug-free. Also can’t debug.

Radare2 basic usage

Normal mode:

- append “?” for help
- commands are a collection of letters with parameters
- a* analysis commands, d* debugger commands, ...
- run “?” for overview

I recommend radare2 intro:

https://github.com/radare/radare2/blob/master/doc/intro.md

Radare2 visual mode

- Enter with “V”, leave with “q”
- Change display type with “p” and “P”
- Navigate with arrows or hjkl
- Graph-mode with “V”, exit with “q”
- Follow jumps by typing the jump’s annotated label
- Enter normal mode commands with “:”

Again, the intro document is really helpful

And now some workshop time!

(If anyone is interested, otherwise demo time)

git@git.vartijat.de:fpape/Radare2-Demo.git

Haxx0r my password

bin/hardcoded/easypeasy.32

static analysis

I'd just like to interject for a moment

binary editing with /bin/uname

Bufferoverflow

bin/overflow/level03
Example stolen from io.netgarage.org