Brief introduction to RE using radare2

Arnau Gàmez i Montolio | @arnaugamez

6 April 2018

Agenda

- About
- Preliminars
- Basics
- Introducing radare2
- Get dirty
- More features (quick glimpse)
- Documentation & resources
- Questions

About

Who am I

Arnau Gàmez i Montolio | @arnaugamez

- 20yo. Maths & CS student @ UB
- President of Hacking Lliure
- Worked as software developer in research groups @ UB
- Many CONs (mainly infosec)
- Also interested in music (pianist), rubik's cube(s)...

*Who am I not: RE pro, r2 expert (at all).

Preliminars

Quick poll

How many of you are students? CS? Engineering?

How many of you are working in infosec? RE / lowelevel stuff?

-

How many of you know radare2?

How many of you use (or have used) radare2?

What is Reverse Engineering

Also known as reversing, and usually referred by RE.

Understanding the internal mechanisms in a piece of software or hardware in order to:

- Find (and fix) vulnerabilities/bugs
- Find hidden features / extend functionalities
- Bypass security protections (exploiting, cracking)
- Simply the joy of understanding how it works

Scope and some considerations

- We will focus on:
  - ELF executables from C compiled code under GNU/Linux
  - Basic static analysis (quick visit to dynamic capabilities, though)
- We assume basic notions of computer organization, reading simple assembly and familiarity with command line interface. Anyway, we'll make a quick reminder
- Main ideas and concepts are easily extrapolable to other archs and platforms
- We will go from slides to live demo and back

Basics

Source code & compilation

A source code file is written in a ‘high level’ language. Think about C for the sake of simplicity.

Just keep in mind that a compiler turns C source code into machine code.

If we are on a GNU/Linux distro, for example:

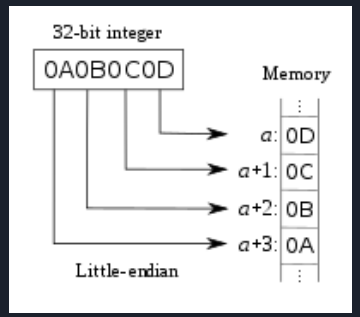
gcc -o helloworld helloworld.c

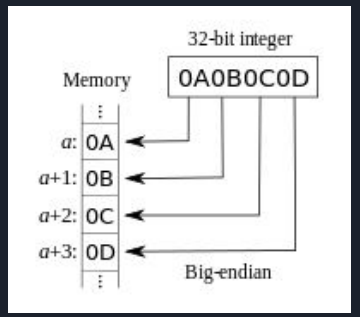
gcc compiler will turn C code in helloworld.c file into ELF machine code file helloworld, which is literally a bunch of 0’s and 1’s usually represented in base-16 (Hexadecimal) for better readability.

*Math trick: bin base (2^1) | hex base (2^4)

Memory

- Memory is addressed by byte (1 byte = 8 bits)
- 256 discret values on 1 byte (0x00-0xFF, 0b00000000-0b11111111)
- On a 32/64 bit arch, 2^32/2^64 (theoretically) addressable memory
- Phyisical memory vs Virtual Memory: every process gets access to full address space
- Little endian vs Big endian: sequential order in which bytes are arranged





Registers (CPU)

- Instruction pointer:
  - rip: points to next instruction
- Stack:
  - rsp: stack pointer (top)
  - rbp: base pointer (bottom)
- Data:
  - rsi: source index
  - rdi: destination index
- General purpose:
  - rax: return values
  - rbx, rcx, rdx

It’s possible to address different parts of a register

rax (64 bits): 0x1122334455667788
eax (32 bits): 0x55667788
ax (16 bits): 0x7788
ah (8 bits): 0x77
al (8 bits): 0x88

Assembly

- Intel vs AT&T
  - Intel: <instruction> <dst>, <src>
  - AT&T: <instruction> <src>, <dst>
- We'll use Intel syntax: more extended and cleaner
- Don't worry if you have not read asm before, I will explain everything needed in demos
- CPU flags:
  - ZF
- Control Flow:
  - call, jump

http://www.jegerlehner.ch/intel/IntelCodeTable.pdf

Introducing radare2

What is radare2

- Free and open source reversing framework (re)written in C
- Many tools (showed just some):
  - rabin2: binary program info extractor
  - rahash2: block based hashing utility
  - radare2 (shell, main tool): hex editor, disassembler, debugger...
  - r2pm: own package manager
- Portable, scriptable, extensible via plugins
- Great community
- Release every 6 weeks
- r2con congress. 3rd edition coming on 5-8 September 2018

What can radare2 do

- Disassemble binaries of several archs and operating systems
- Analyse code, data, references, structures...
- Debugging, tracing, exploiting...
- Binary manipulation, code injection, patching...
- Mount filesystems, detect partitions, data carving...
- Extract information and metrics from binaries for classification
- Find differences between two files
- Compute checksums of the blocks in a file
- Kernel analysis and debugging

Install/Update radare2 (from git, do yourself a favour)

Even if you have installed/updated it just before entering the room, you are probably outdated.

- Clone de repo if not installed before:

  git clone https://github.com/radare/radare2

- Install (or update without git pull needed):

  ./sys/install.sh (system wide) or **./sys/user.sh** (home user)

Get dirty

Open a file

- Open a file (by default writing is disabled):

r2 /bin/ls

- Open a file in write mode:

r2 -w /bin/ls

- Open a file in debugging mode:

r2 -d /bin/ls

Basic commands

Commands follow simple mnemonic rules.

- Each char in the command is a subcommand of the previous one:
  - s -> seek (moves to memory address)
  - px -> print hexdump
  - pd -> print disassembly
  - wx -> write hexpairs
  - wa -> write assembly
  - aa -> analyse all code
  - q -> quit
- Append ‘?’ to the command to get help about it
- Temporary seek with ‘@’

Extracting binary information

- You can use rabin2 tool to extract binary information
  - Entrypoint (-e)
  - Symbols (-s)
  - Imports (-i)
  - Libraries (-l)
  - Strings (-z)
  - Sections (-S)
- You can access directly from r2 shell with i command
  - ie, is, ii, il, iz...
- Check:
  - man rabin2, i?

(quick demo)

Handy tricks

- Append j (j~{}) for json (indented) output:
  - Example: izj, izj~{}
- Append q for quiet output:
  - Example: izq
- Pipe with shell comands:
  - Example: iz | less
- Run shell commands with ! prefix:
  - Example: !echo hello there
- Internal grep with ~
  - Example: iz~string
- Temporal write setting io.cache to true:
  - e io.cache=true
- **Again, append '?' for inline help (most useful feature ever):**
  - **Example: aa?**

Visual mode and Graph view

- Access visual mode with ‘V’ command
  - Rotate print modes with ‘p’ command
  - Press ‘?’ to get visual mode help
  - Use ‘:’ to run radare command
- Access graph mode with ‘VV’ command
  - Pretty useful to see a function’s workflow
  - You have to be on a function or it won’t show anything
  - Move with **hjkl**
  - Zoom in/out with +/-

Useful tips and advices for newbies

- Add ASM description:

e asm.describe = true

- Change color palette for dark or white backgrounds

eco dark | eco white

- You can save your preferred config ‘e’ commands to ~/.radare2rc file so they will be loaded at r2 start (use -N option to prevent radare from parsing it).
- Don’t play with /bin/ls (or any other system’s binary) in write mode (trust me, you don’t wanna deal with it). Make a copy and play with the copy.
- Did I mention that you can append ‘?’ to get inline help?

More features (quick glimpse)

Debugging

- Starts debugging at dynamic loader (not the entrypoint).
- Debugging options under ‘d’ command (play with d?). Basic usage:
  - db -> set breakpoint
  - dc -> continue execution
  - ds -> step
- Low level debugger. Not aiming to replace source debugger.
- Tiled visual mode ‘V!’ is extremely useful here
- Many backends. For instance:
  - gdb (in core)
  - r2frida (via r2pm): memory access and binary instrumentation. JS injection and hooking.

(quick demo)

r2pipe

- Simple APIs for many languages: C/C++, Java, Go, NodeJS, Perl... and yes, Python too
  - Install on python with:
    - ***pip install r2pipe***
    - ***pip3 install r2pipe***
- JSON deserialization
- Just 4 basic commands:
  - open()
  - cmd()
  - cmdj()
  - quit()

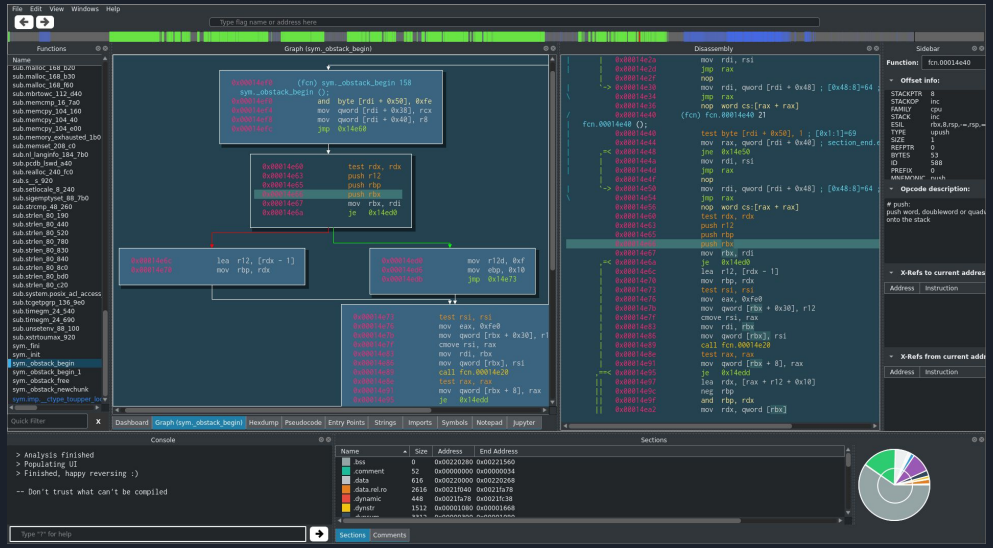
(quick python demo)

ESIL

- Stands for ‘Evaluable Strings Intermediate Language’
- Standard intermediate language in r2
- Each instruction is translated to a single string
  - mov eax, 13 => 33,eax,=
- Used for emulation, assisted debugging
- Search expressions, predict jumps, find references.
- ae subcommands used to manipuilate de VM of ESIL
  - To know more: ae?

GUI | Cutter

- Tons of GUIs have been developed for radare2
- Most advanced and sort-of-official one is Cutter (formerly Iaito).
- Developed with C++ and QT
- Released alongside radare2 releases
- It's pretty, but not as sexy as CLI
- Not aimed for r2 experienced users



Documentation & resources

Documentation & resources

- "*It's already documented in C*" --pancake
- radare2 book (updated by Maijin from r1book):
  - https://www.gitbook.com/book/radare/radare2book/details
- radare2 explorations:
  - https://www.gitbook.com/book/monosource/radare2-explorations/details
- Inline help appending a question mark '?' to any command (yes, again)
- Tons of talks (many recorded and uploaded) and posts.
  - r2con17 talks: https://www.youtube.com/playlist?list=PLjIhlLNy_Y9Oe-nfcPEpaki0_En5dhQ5S
  - 33C3 talk (radare2 demystified): https://www.youtube.com/watch?v=fnpBy3wWabA
  - More: http://rada.re/r/docs.html | http://rada.re/r/talks.html
- Support & help:
  - IRC: #radare at irc.freenode.net | Telegram: https://t.me/radare

Questions

Thank you | Contact


arnau@hackinglliure.org


@arnaugamez | @HackingLliure