Overcoming fear: reversing with radare2

Arnau Gàmez i Montolio | @arnaugamez
Who am I

- **Student** - *Maths & CS* @ UB
- **President** - [@HackingLliure](https://twitter.com/HackingLliure)
- **Collaborator** - [#r2con](https://www.r2con.org)
Who am I **NOT**

- Professional reverser
- radare2 expert
- radare2 developer
Motivation

• Demystify radare2
• Simple explanations from a non advanced user
• Organize and share knowledge
• CONs should have intro sessions
Who are you

- Students?
- Working in infosec?
  - Low level, RE?
- Know radare2?
- Use radare2?
Outline

1 Overview of radare2
2 Commands & interaction
3 Visual modes & navigation
4 Config. & customization
5 Code emulation with ESIL
6 Extensibility & scripting
7 Common use cases
8 Extras
9 Documentation & resources
10 Conclusions
About radare2

- FOSS Reverse Engineering framework
- (Re)written in C by pancake
- Built from scratch without any 3rd-party dependency
- Portable, scriptable, extensible...
About radare2

• Release every 6 weeks
• Great community
• r2con: annual congres @ Barcelona (early september)
radare2 capabilities

- Disasm bins of several archs & OSs
- Analise code and data
- Low level debugging and exploiting
- Binary manipulation
radare2 capabilities

• Forensics: mount FS, detect partitions, data carving
• Extract metrics for binary classification
• Kernel analysis and debugging
radare2 has support for...
Architectures

i386, x86-64, ARM, MIPS, PowerPC, SPARC, RISC-V, SH, m68k, m680x, AVR, XAP, System Z, XCore, CR16, HPPA, ARC, Blackfin, Z80, H8/300, V810, V850, CRIS, XAP, PIC, LM32, 8051, 6502, i4004, i8080, Propeller, Tricore, CHIP-8, LH5801, T8200, GameBoy, SNES, SPC700, MSP430, Xtensa, NIOS II, Java, Dalvik, WebAssembly, MSIL, EBC, TMS320 (c54x, c55x, c55+, c66), Hexagon, Brainfuck, Malbolge, whitespace, DCPU16, LANAI, MCORE, mcs96, RSP, SuperH-4, VAX.

File Formats

ELF, Mach-O, Fatmach-O, PE, PE+, MZ, COFF, OMF, TE, XBE, BIOS/UEFI, Dyldcache, DEX, ART, CGC, Java class, Android boot image, Plan9 executable, ZIMG, MBN/SBL bootloader, ELF coredump, MDMP (Windows minidump), WASM (WebAssembly binary), Commodore VICE emulator, QNX, Game Boy (Advance), Nintendo DS ROMs and Nintendo 3DS FIRMWAREs, various filesystems.

Operating Systems

Windows (since XP), GNU/Linux, OS X, [Net]Freenet/FreeBSD, Android, iOS, OSX, QNX, Solaris, Haiku, FirefoxOS.
Runs everywhere
Supports everything
Get radare2

Clone repo
$ git clone https://github.com/radare/radare2

Go to radare2 created directory
$ cd radare2

Install / update *(pulls last version from git)*
$ ./sys/install.sh

check https://www.radare.org/r/down.html for other/more installation options
KEEP CALM AND USE R2 FROM GIT
Tools included

rax2 -> base converter
rabin2 -> extract binary info
rasm2 -> (dis)assembler
rahash2 -> crypto/hashing utility
radiff2 -> binary diffing
Tools included

ragg2  ->  compile tiny bins
rarun2 ->  run with different env
rafind2 ->  find byte patterns
r2pm   ->  r2 package manager
radare2 ->  main tool
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
Spawn an r2 shell

$r2$ command is a symlink for $radare2$

<table>
<thead>
<tr>
<th>Open file</th>
<th>Don’t load user settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ r2 /bin/ls</td>
<td>$ r2 -N /bin/ls</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Open file in write mode</th>
<th>Alias for r2 malloc://512</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ r2 -w /bin/ls</td>
<td>$ r2 -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Open file in debug mode</th>
<th>Open r2 w/o opened file</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ r2 -d /bin/ls</td>
<td>$ r2 --</td>
</tr>
</tbody>
</table>
Basic commands
r2 commands are based on mnemonics

- **s** – seek
- **px** – print hexdump
- **pd** – print disasm
- **wx** – write hexpairs
- **wa** – write asm
- **aa** – analyse all
- **ia** – info all
- **q** – quit
Append `?` to any command to get **inline help** and available subcommands
Handy tricks

- Append $j$ ( $j~\{\}$ ) for $json$ (intended) output
  
  Example: izj, izj~\{}

- Append $q$ for $quiet$ output
  
  Example: izq

- Internal grep with $~$
  
  Example: iz~string
Handy tricks

- Pipe with shell commands
  Example: iz | less

- Run shell commands with ! prefix
  Example: !echo HITB rocks

- Temporary seek with @
  Example: pd @ main
Demo

Defeat simple crackme

cc @pof @jvoisin
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
Visual mode

• Access visual mode with V command
  • Rotate print mode with p command
  • Press ? to get visual mode help
  • Use : to run r2 command
Graph view

• Access graph view with **VV** command
  • Follow functions' flow
  • Must be seeked on a function
  • Move with arrows or **hjkl**
  • Zoom in/out with +/-
Visual panels

- Access visual panels with \texttt{V!} command
  - Really useful when debugging
  - Default panels
  - Customize panel views
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
Evalueable configuration variables

- Use `e` command (subcommands) to tune radare2
- List configuration variables
  - Show values: `e`
  - Show description: `e??`
Evaluate configuration variables

- Look for them: `e??~whatever`
- List possible values: `e conf.var = ?`
- Set new value: `e conf.var = new_value`
## Useful configuration variables

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use UTF-8 chars</td>
<td><code>e scr.utf8 = true</code></td>
</tr>
<tr>
<td>Curved UTF-8 corners</td>
<td><code>e scr.utf8.curvy = true</code></td>
</tr>
<tr>
<td>Show opcode description</td>
<td><code>e asm.describe = true</code></td>
</tr>
<tr>
<td>Enable pseudo syntax</td>
<td><code>e asm.pseudo = true</code></td>
</tr>
<tr>
<td>User uppercase syntax</td>
<td><code>e asm.ucase = true</code></td>
</tr>
<tr>
<td>Enable cache (r/w)</td>
<td><code>e io.cache = true</code></td>
</tr>
</tbody>
</table>
More handy tricks

• Add `e` configuration commands to `~/.radare2rc` file to load them by default
  • `-N` prevents loading custom configuration

• Visually explore and modify configuration variables with `Ve`
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
What is emulation?

- Simulate the execution of code of the same or different CPU
What is emulation?

- Simulate the execution of code of the **same or different CPU**

Run **games from old consoles**
What is emulation?

- Simulate the execution of code of the same or different CPU.
Why emulation?

• **Understand** specific snippet of code
• **Avoid risks** of native code execution
• **Help debugging** and **code analysis**
• **Explore** non-native executables
Intermediate languages

"Language of an abstract machine designed to aid in the analysis of computer programs" -- wikipedia

Vital for (de)compilation
What is ESIL?

- **Evaluable Strings Intermediate Language**
- Small set of instructions
- Based on reverse polish notation (stack)
- Designed with *emulation and evaluation in mind*, not human-friendly reading
What is ESIL?

- Infinite memory and set of registers
- “Native” register aliases
- Ability to implement custom ops and call external functions
Why ESIL?

- Need for emulation on r2land
- Easy to generate, parse and modify
- Extensibility
- Why not?
ESIL

Stack machine on steroids
Stack machines / PDA's
Stack machines / PDA's

- input symbol
- current state
- stack symbol
- state transition
- manipulate stack (push/pop)
Visual animation

3, 5, +

Stack
Visual animation

Stack

3, 5, +
Visual animation

5, +

Stack

3
Visual animation

Stack

5, +

3
Visual animation

+ Stack

5
3
Visual animation

Stack

+ 5
  3
Visual animation

Stack

+ 5

3
Visual animation
Example

ae 3,5,+
Expanding stack machines

We are here

We want to be here

cc @condr3t
HOW?
HOW?

STEROIDS
(aka cheating)
Steroids x1

- Add random access operations
- Add control flow operations
Steroids x2

- **Register** access
- Add "**extra tape**" with random access (virtual memory, VM stack)
Basic practical usage

ESIL options are under ae (analysis esil) subcommands

- **aei** – *init*
- **aeim** – *init* memory
- **aeip** – *inst. pointer*
- **aes** – *step*
- **aesu** – *step until*
- **aeso** – *step over*
- **aess** – *step skip*
- **aer** – *registers*
ESIL operands

Check \textit{ae??} on a radare2 shell

(description and examples)
ESIL internal vars (flags)

Prefixed with $ | read-only

- $z – zero flag
- $cx – carry flag from bit x
- ...

Updated on each operation. Used to set flags for particular arch.
Demo

Defeat simple crackme (revisited)

cc @pof @jvoisin
Demo

Deobfuscate encrypted code

cc @superponible
Extensibility

• radare2 design is composed by several C libraries

• Standalone programs (r2land tools) built on top of one or more of them
Structure

• libr/ -> modules with dependencies
  • [lib]/p -> plugins for each module
• binr/ -> binary programs
• shlr/ -> ripped code from 3rd party
Plugins

- (dis)asm -> rasm2 -L
- file formats -> rabin2 -L
- IO and debug -> r2 -L
- ...

HITBSecConf - Amsterdam
May 9, 2019
Plugins

• Install/manage non-core plugins via r2pm
  • Init pkg manager  ->  r2pm init
  • Install plugin    ->  r2pm -i [plugin]

• Check *man r2pm*
Scripting

• Bindings for many languages:
  • Java
  • Go
  • NodeJS
  • Python
  • ...

HITBSecConf - Amsterdam
May 9, 2019
Scripting

• r2pipe API
  • input  -> r2 commands
  • output -> r2 output
  • JSON deserialization into native objects
r2pipe: python example

• Installation
  • `pip(3)` install r2pipe

• Usage
  • `import r2pipe`
  • `open()`, `cmd()`, `cmdj()`, `quit()`
Demo

Deobfuscate encrypted code (revisited)

cc @superponible
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
Debugging

- Debugging options under `d` command
- Starts debugging at dyld, not entrypoint
- Low level debugger, not aiming to replace source code debugging
- Many backends: gdb, r2llvm, r2frida...
Exploiting

- Search strings -> / [string]
- Search ROP gadgets -> /R
- Find function xrefs -> axt [offset]
- Find w/x sections -> iS
Exploiting

- List (libc) imports -> is~imp
- De Bruijn pattern -> ragg2 -P [size] -r
- Find offset of pattern -> wopO [value]
- Craft shellcode -> ragg2 -a [arch] -b [bits] code.c
Exploiting

• More on exploiting
  • https://radare.gitbooks.io/radare2book/content/tools/ragg2/ragg2.html
  • http://radare.today/posts/using-radare2/
  • https://www.megabeets.net/a-journey-into-radare-2-part-2/
Outline

1. Overview of radare2
2. Commands & interaction
3. Visual modes & navigation
4. Config. & customization
5. Code emulation with ESIL
6. Extensibility & scripting
7. Common use cases
8. Extras
9. Documentation & resources
10. Conclusions
Cutter: r2 official GUI

- C++ and QT
- Released alongside r2 releases
- Check https://cutter.re
Decompilation

• r2dec
  • asm to pseudo-C written in JS
  • https://github.com/wargio/r2dec-js

• r2retdec
  • Bridge between r2 and retdec
  • https://github.com/securisec/r2retdec
Decompilation

- **radeco**
  - Aims to be "the r2 decompiler"
  - Written in Rust. Uses ESIL as input
  - Mainly developed during GsoC
  - Work in progress
  - [https://github.com/radareorg/radeco](https://github.com/radareorg/radeco)
r2frida

- Use `frida` as backend for memory access and in-process injection
- Install  
- Open  
- Use  
  - Prefix with `\` (check `\?`)

Install  

Open  

Use  

Prefix with `\` (check `\?`)

HITBSecConf - Amsterdam
May 9, 2019
@arnaugamez
r2frida

• Links
  • https://github.com/nowsecure/r2frida
  • https://github.com/enovella/r2frida-wiki
Demo

r2frida
Written documentation

• "Already documented in C" 😊

• radare2 official book
  • https://radare.gitbooks.io/radare2book
  • Continuously updated
  • Call for GSoD
More resources

• radare2 explorations
  • https://monosource.gitbooks.io/radare2-explorations

• Blogs
  • http://radare.today
  • https://megabeets.net
More resources

- Recorded talks
  - r2con2016
  - r2con2017
  - r2con2018
  - Tons of them: just check on YouTube
Extra tips

• Remember to append `?` for inline help

• Quick trick inside an r2shell
  • Interactive help search `-> ?*~...`

• Quick trick++
  • `alias r2help="r2 -q -c '?*~..' -"`
Support

- IRC
  - #radare at irc.freenode.net
- Telegram
  - https://t.me/radare

IRC & Telegram are bridged
Conclusions

- radare2 is not *that* difficult
  - mnemonic commands
  - UNIX-like shell
  - Less than 10 commands to do most of the tasks
  - Inline help appending ?
Conclusions

- There are many ways to contribute to open source projects like radare2
  - Code
  - Write documentation
  - Report issues
  - Use and share it
Invitation

• r2con2019
  • Community driven
  • From 4th to 7th September @ Barcelona
  • Trainings and conference talks
  • Check https://rada.re/con/2019