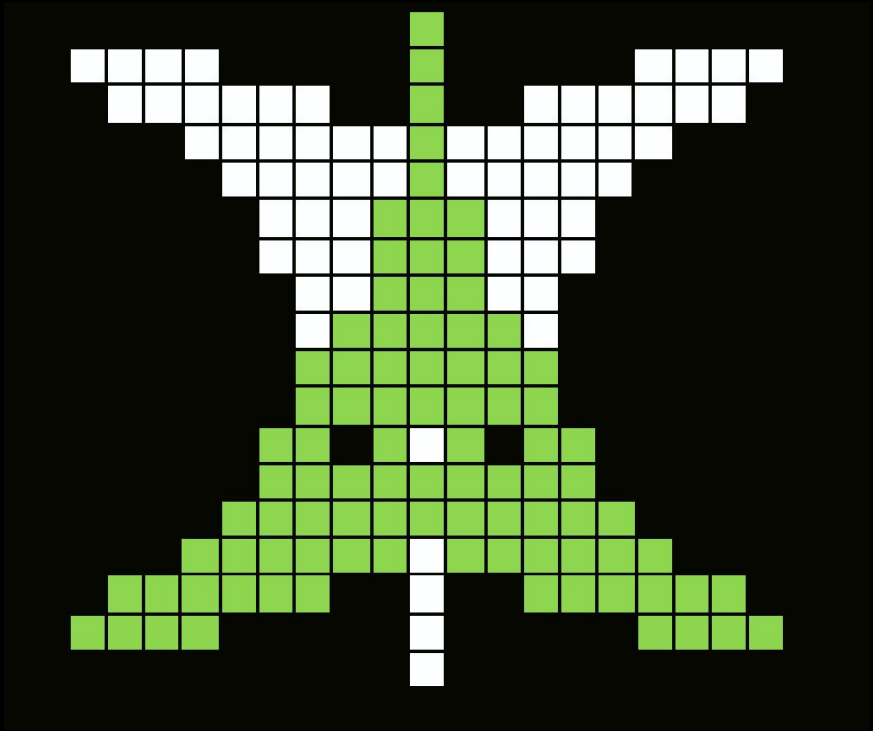
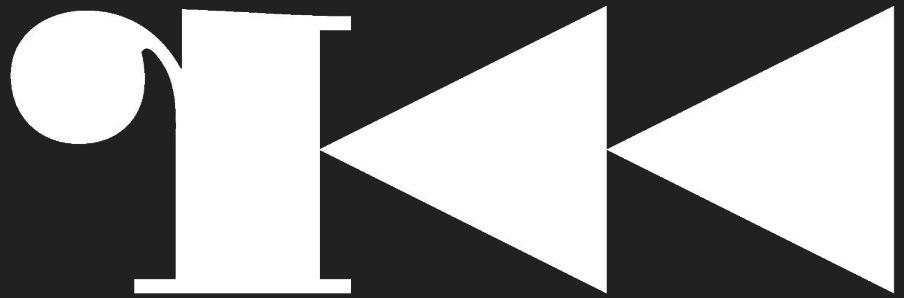


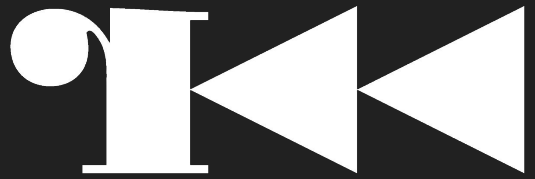
Digging into radare2 for fun and profit

AvTokyo2017 // pancake

What’s Radare2?



What's Radare2?



- 12 yo Free-Open-Source Project
- Reverse-Engineering Framework and Toolset
- Originally written by me (pancake)
- Growing community and contributors over time
- Switch from main/single dev to project leader and maintainer

- Release every 6 weeks
- Bumps +1.0 every year after r2con
- r2con happens in Barcelona (in 2017 we had 230 attendees this year)
- All talks are published in YouTube

Who Am I?

- Free Software enthusiast and Geek
  - Born in Barcelona, Catalonia
  - Wrote and maintain several free software tools
  - Participated at defcon CTF 3 years in a row
  - I enjoy drawing
  - Father
- Links
  - Check github and bitbucket on radare and trufae users
  - You can also find my profile in https://twitter.com/trufae
- Currently working at NowSecure (Mobile Security Analyst, doing R+D)
  - Optimizing codecs in assembly for mips, arm and x86
  - Develop firmware for embedded devices for realtime traffic analysis in the highways.
  - Forensics mainly on Windows platforms
  - Instructor in courses related to programming and hacking

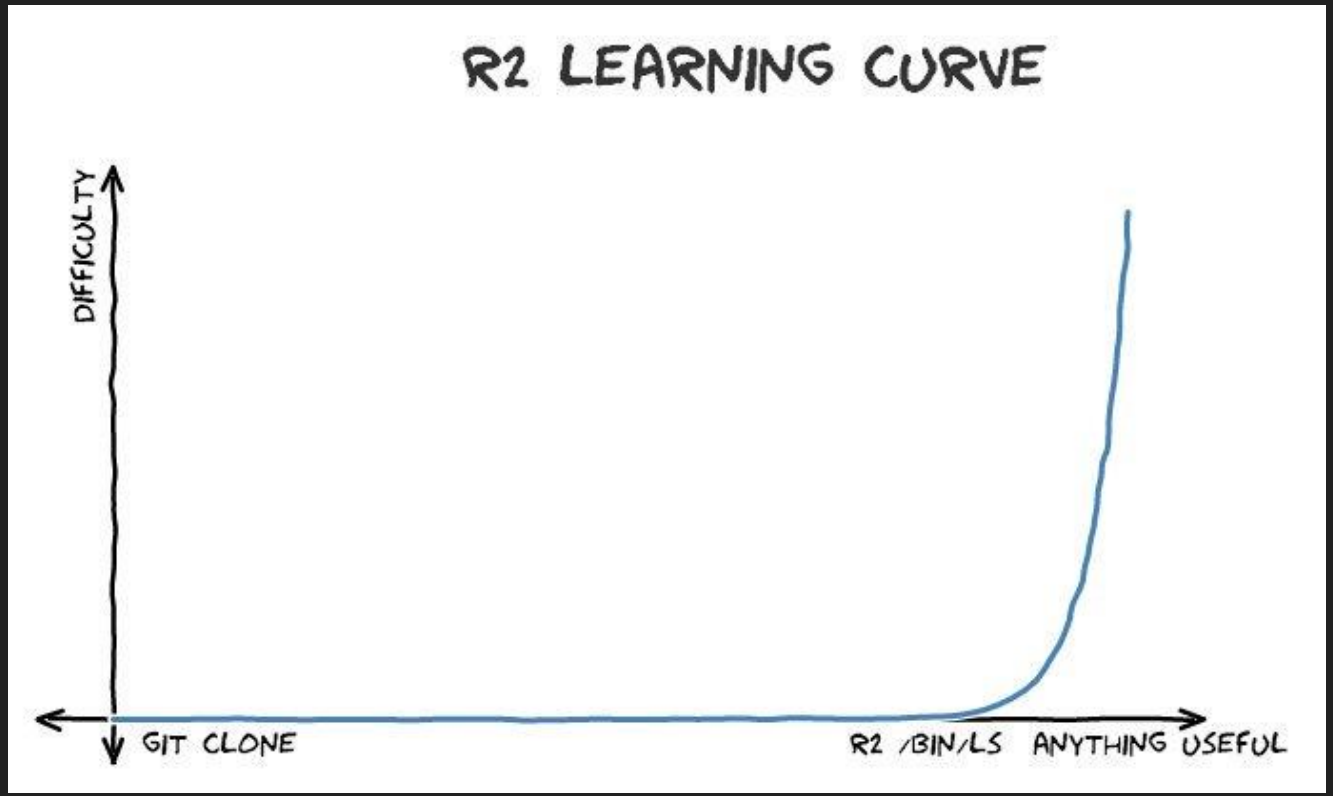
What Can It Do For You?

- Analyze Programs to understand what they are doing
- Identify strings in several encodings (chinese, korean, cyrilic..)
- Find References to strings using different techniques
- Carve memory dumps or firmwares for known magic
- Mount filesystems and parse partition tables
- Debug programs native or remotely via gdb, r2, frida, windbg,..
- Emulate parts of the program to decrypt blocks
- Use external decompilers or graphing tools
- Check differences between two binaries
- Play games like 2048 or r2wars!

Runs Everywhere!

- Many Operating Systems:
  - Windows, Linux, Mac, QNX, Solaris, NetBSD, FreeBSD, BeOS, Android, iOS, ….
- Many Architectures
  - x86, arm, mips, sparc, ppc, z80, 6502, 8051, avr, wasm, snes, java, dalvik, hppa, ...
- Supports native debugger in most target arch/os pairs
- Also compiles to web-assembly and asm.js
- Can be used in local or remotely

(demo rasm2 -L rabin2 -L r2 -L)


R2 LEARNING CURVE
DIFFICULTY
GIT CLONE
R2 /BIN/LS
ANYTHING USEFUL

Learning Curve

- Steep at first, but pleasant in long term.
- About 10 commands is all you need
- Orthogonality enables commands to be combined and extended
- About 2 weeks of daily use to get in touch

- Learning by doing
- A matter of having interest and dedication
- Different workflow compared to other tools
- It’s open-source! So rwx!

It's Documented®

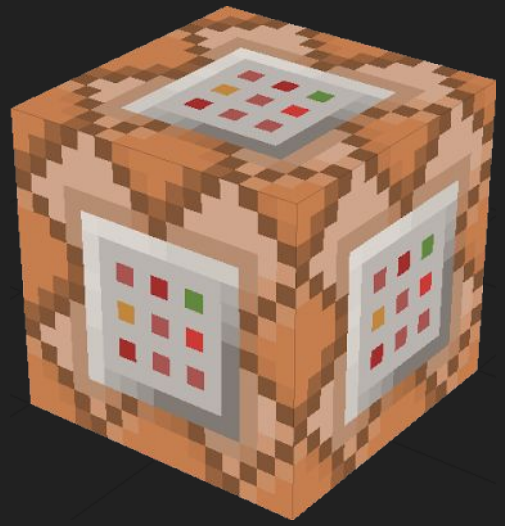
- Fully documented in C
- We have a collaborative book, based on r1 and
- Several blog posts (follow @radareorg on Twitter to catch more)
- Many talks in YouTube and Vimeo
  - All r2con 2016, 2017 videos are published
- Self documented by appending the '?' to the commands
- UNIX Manual pages (man)
- Public IRC and Telegram channels with more than 800 users

https://twitter.com/radareorg

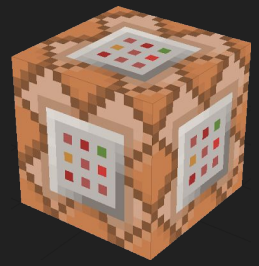
https://t.me/radare

Basic Commands

- Seeking
  - Relative / Absolute / Partial locations (s+0x10, s..10, s 10)
  - History (!, s!)
  - Blocksize (b, @!)
- Printing
  - Hexdump (px, pxr, pxa, prc, pxA, …)
  - Disasm different archs (pd, pD) @a:mips, e asm.arch
  - Decode structures (pf)
  - Checksums, entropy, statistics (p=, ph)
- Writing
  - Assemble new instructions (wa)
  - Strings in different encodings (w)
  - Hexpairs (wx)
  - Contents of files (wf)



Command Modifiers



Prefix

- [1-9]
  - Repeat command n times
- . (backtick)
  - Interpret the output of the command as r2
- “
  - Ignore special characters
- `
  - Insert the output of a command
- !
  - Shell escape
- $
  - Alias command
- \
  - Alias for =!

Suffix

- @
  - temporal seek
- |
  - system pipe
- ~
  - internal grep
- >
  - file redirect
- #
  - Comment
- ?
  - Show help
- j
  - Output in JSON

Demo

Mounting FileSystems and Searching Stuff

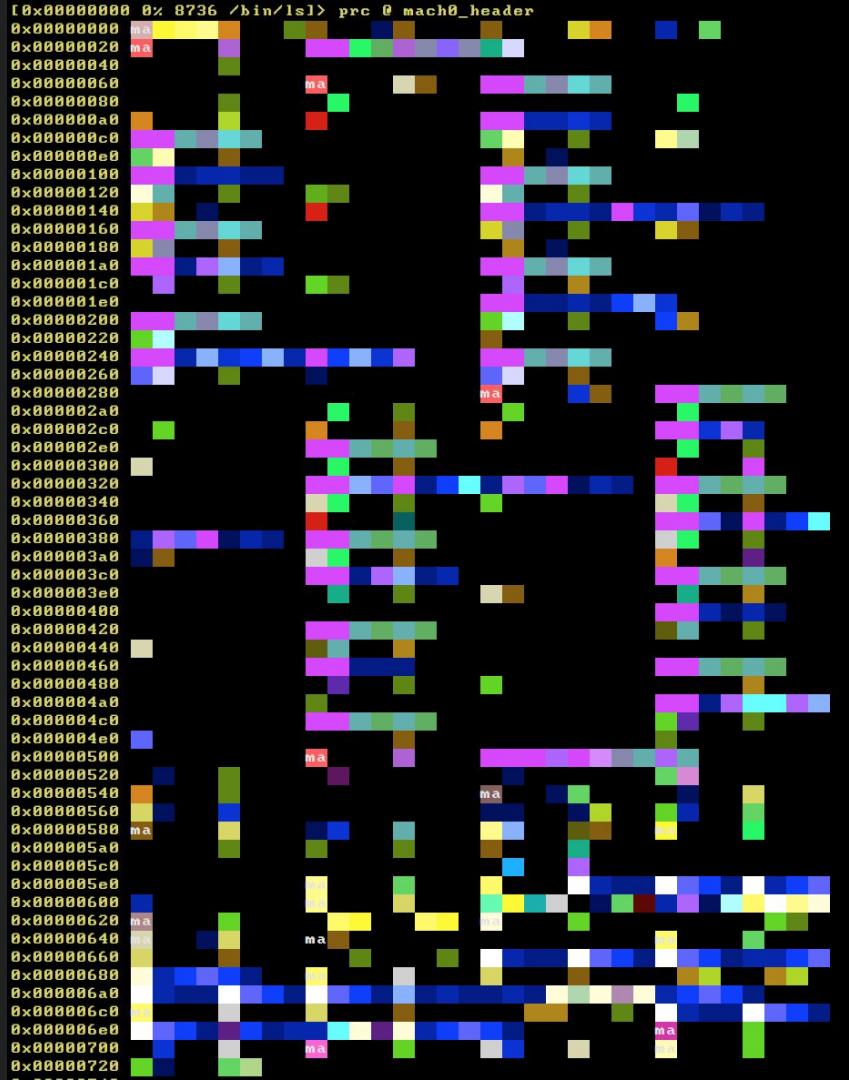
- Initially, radare started as a forensics tool.
- Find offset in disk for a file and vice-versa.
- Search patterns or known headers and dump results
- HFS, FAT, NTFS, EXT2, …
- Squash, jffs2 are wip and not yet working

- The ‘m’ command reads partition tables and mount filesystems.
  - Most plugins based on GRUB code. (GPL warning)
  - Also io and r2 filesystems (wip)

Demo

Parse Binary Headers

- Supports a large list of bin headers
- rabin2 -l
- Can parse malformed and fuzzed binaries
- Loads from disk or memory
- IO layer abstracts access to data
- Emulate a Virtual Address space
- r2 -nn
- rabin2 -H
- Parsing memory headers (oba, .!rabin2 -r)
- Extract resources, entitlements, ...



Demo

Analyze and Disassemble

Most beginners use to go for a generic analysis

- -A, aa, aaa, aaaa, aaaaa, aaaaaaaah!

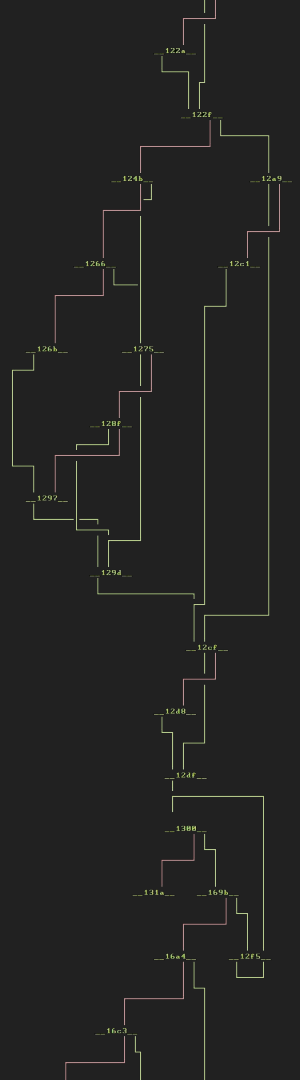
But there are a lot of ways to tweak the analysis

- e??anal.

And several commands to do fine-grained analysis

- aac, aar, aae, aav, aab, ...

Emulation with ESIL is used in some commands.



Analyze and Disassemble

Listing functions

- afl

Listing basic blocks

- afb

Graphing them

- agf

Rename function

- afn

Analyzing a single opcode

- ao

Analyze single function

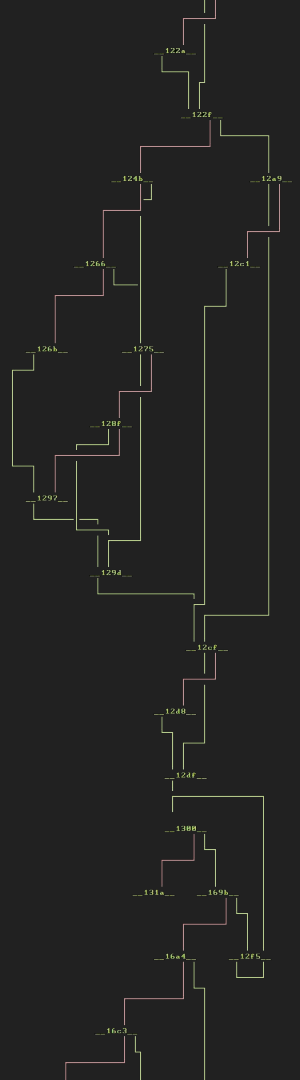
- af (e anal.hasnext)

Alternative analysis loop

- a2f

Autoname function

- afna



Analysis Options

Assume there’s more code

- anal.hasnext

Discover strings when analyzing

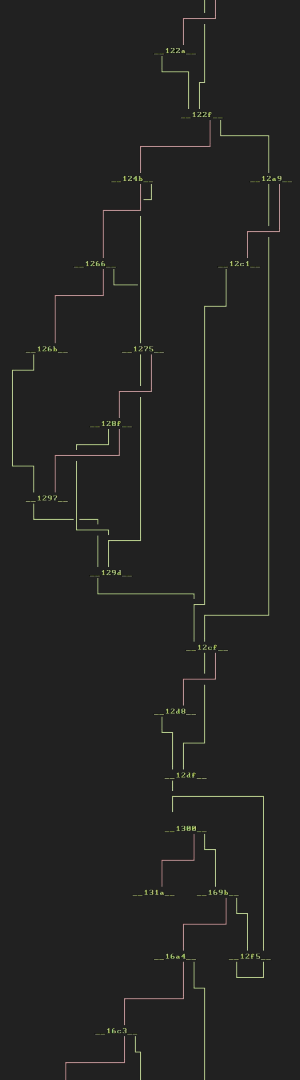
- anal.strings

When messing with non-executable regions as code

- anal.noncode

Analyze Jump-Tables

- anal.jmptbl



Demo

Print Data In Multiple Formats

By default uses the ‘b’ blocksize

Overview with zoom view, entropy, color boxes, instruction blocks, ...

- p?

Format string-like strings by parsing the memory at given address

- pf xxi foo bar cow @ addr

Demo

Debug And Emulate

R2 will bridge all debugger actions to the ESIL vm when open statically. Use -d to open the target file in debugger mode.

- r2 -d

Continue until given address

- dcu addr

Step into / step over

- ds, dso

Debug And Emulate

Support native and remote debugging engines.

- dbg:// winedbg:// gdb:// windbg:// qnx:// ..

Many low level features:

- Backsteps (Thanks Ren Kimura!)
- Memory snapshots
- Software/Hardware breakpoints
- Assisted debugging (emulation + debug)
- Tracing
- Filedescriptor manipulation

Rarun2 Profiles

Execution environment can be configured in:

- Textfile specified via r2 -r or dbg.profile
- Comma separated list of directives via dor or -R commandline flags

Allows to change gid, uid, chroot, chdir, environment, arguments, filedescriptors..

- Any directive value can be a string, a slurped file or output of a program

$ man rarun2

Print Data In Debugger

Show stack contents

- dbt - backtrace
- pxr@r:SP

Show local variables and their values

- afvd

Missing the colorbar?

- p=

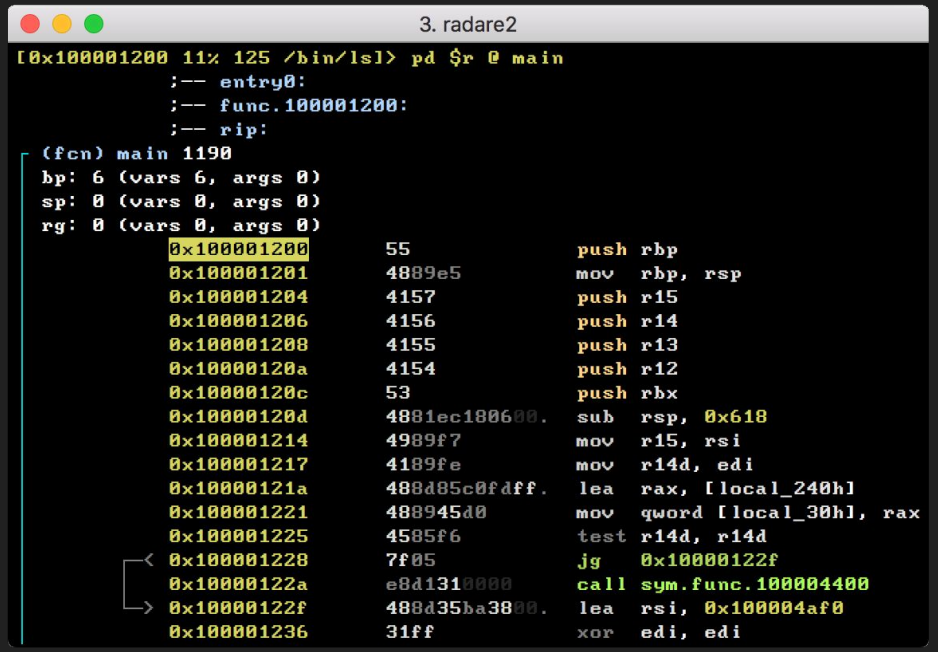
Demo

Console Interface

- Mostly a command-line prompt
- Eventually a Visual mode
- Embedded web server (r2 -c=H)

Visual mode bind actions to keys instead of commands.

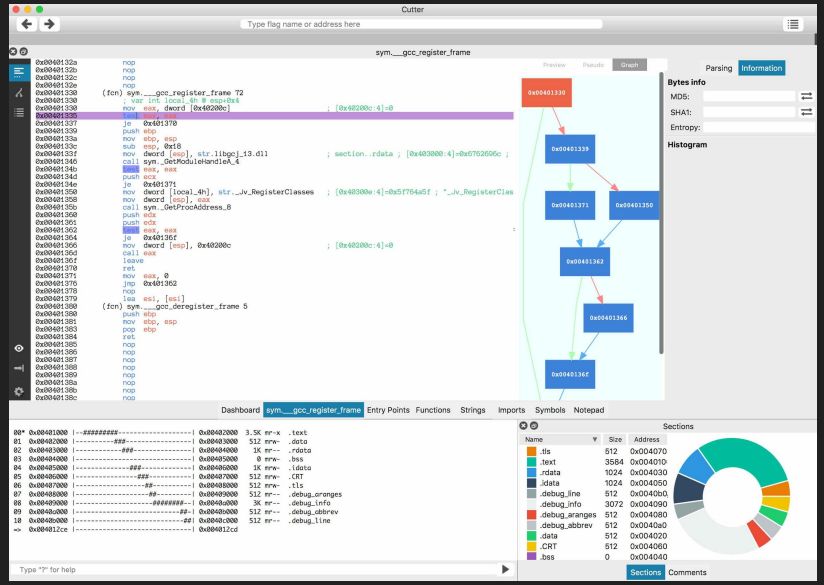
- Change view with pP”|=...
- Step with ‘s’, toggle bp, continue, ..
- Seek history
- Visual assembler
- Interactive Graphs



Graphical Interface

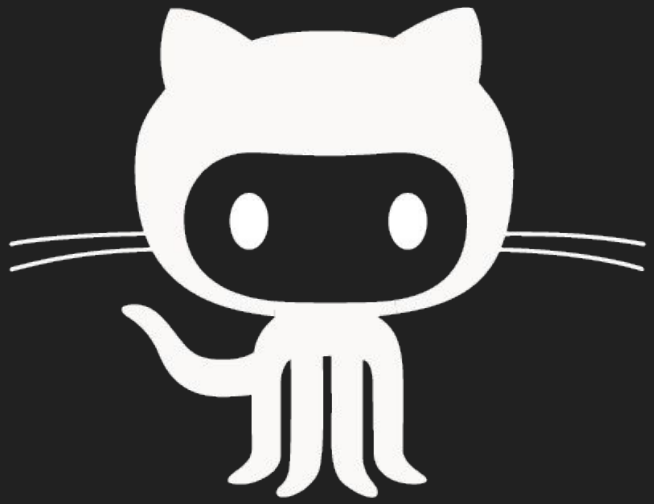
We can install most of them via r2pm (sorted by time)

- Gradare2 (simple gtk2/3 + vte ui)
- Ragui (unreleased)
- Bokken (unmaintained)
- Blessr2 (nodejs-blessed based UI)
- WebUIs (material, enyo, tiled, …)
- Radare2gui (.net for windows)
- Cutter (previously known as laito)
- …



Demo

Easy To Modify and Improve



- Libraries
  - Default installation method done with symlinks
  - cd libr/* ; vim ; make; run
- Plugins
  - ./configure-plugins
  - r2pm
- Bindings
  - C API have bindings for Python, Perl, Ruby, Scheme, Haskell, …
  - Thanks to Valabind
- Scripting
  - Script with RLang using Python, C, or even Vala
  - Bindings automatically loaded if needed

r2pipe

Easiest way to automate r2

- Single api function: run a command, returns the output
- Supports lot of programming languages

Multiple communication channels

- Pipe
- Socket
- HTTP
- Native
- Spawn

Demo

Third Party Stuff

The project covers a huge

- r2 can be extended with scripts, plugins, patches…
- Most of them available via r2pm, our package manager
  - Install everything by default in your home (unless -g is used)
- Decompilers, SMT Solvers, More disassemblers, tools, …
- Use r2pm -s to list them all

(DEMO)

r2frida

- Frida is a hooking engine, supports injecting javascript to interact with a running process in local or remotely.
  - It comes with a REPL, a tracer, process list, etc..
- Radare2 can be used as a frontend for Frida
  - Uses the power of the IO plugins
  - Access functionality via io->system
  - Using the \ or =! Command
- There’s also r2preload in rarun2 to inject into a process using self://

WineDBG

- Wine is not a Windows Emulator
- Comes with winedbg, a very rustic commandline low level debugger
- The io.winedbg plugin allows to interface with it
- Similar to the bochs:// one
- Allows to debug window programs with r2 on Linux and Mac platforms.
- In early stage of development
  - Lot of potential here

Other 3rd Party Debugger Backends

- GDB / LLDB
  - Debug kernels via the gdbserver embedded in qemu, vmware, vbox, ..
  - Apple’s debugserver, GNU’s gdbserver
  - AVR emulator and jtag
- WINDBG
  - Connect to a windbg server
- WINEDBG
  - Debug Windows programs on wine (Linux, Mac, ..)
- QNX
  - The debugserver used in automobile
- Bochs
  - X86 CPU debugger

Demo

しつもんがありますか？

(Questions?)

Thanks For Watching!