Overcoming fear: reversing with *radare2*

Arnau Gàmez i Montolio | *@arnaugamez*

May 9, 2019
HITBSecConf - Amsterdam

Who am I

- Student - *Maths* & *CS* @ UB
- President - *@HackingLliure*
- Collaborator - *#r2con*

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

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)

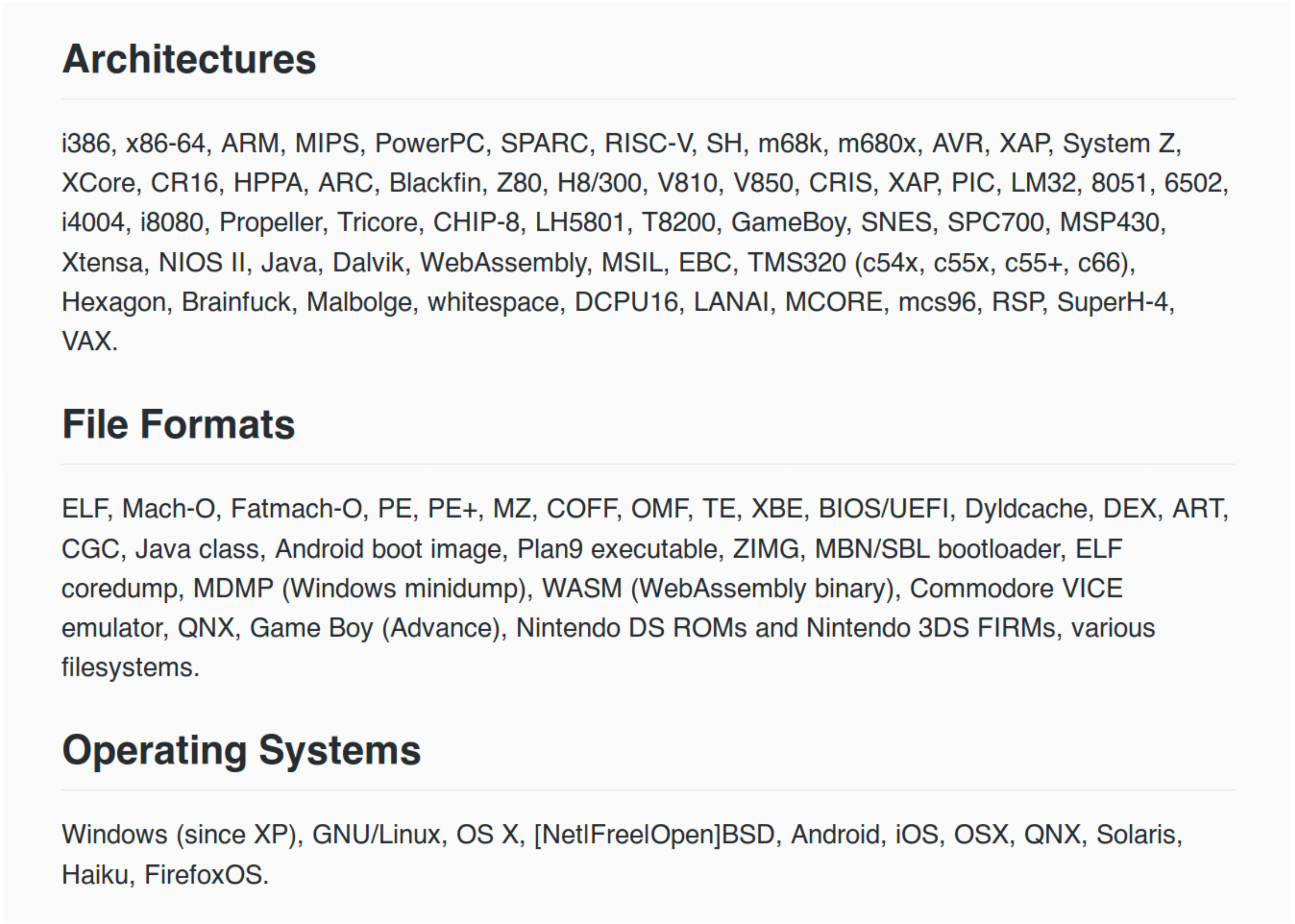
radare2 capabilities

- 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 FIRMs, various filesystems.

Operating Systems

Windows (since XP), GNU/Linux, OS X, [Net|Free|Open]BSD, 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

Spawn an r2 shell

r2 *command is a symlink for* ***radare2***

Open file

```
$ r2 /bin/ls
```

Open file in write mode

```
$ r2 -w /bin/ls
```

Open file in debug mode

```
$ r2 -d /bin/ls
```

Don’t load user settings

```
$ r2 -N /bin/ls
```

Alias for r2 malloc://512

```
$ r2 -
```

Open r2 w/o opened file

```
$ r2 --
```

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 ***j***son (intented) output

  Example: izj, izj~{}

- Append ***q*** for ***q***uiet 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

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 *V!* command
  - Really useful when debugging
  - Default panels
  - Customize panel views

Outline

Evaluable configuration variables

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

Evaluable configuration variables

- Look for them: ***e??~whatever***
- List possible values: ***e conf.var = ?***
- Set new value: ***e conf.var = new_value***

Useful configuration variables

Use UTF-8 chars
e scr.utf8 = true

Curved UTF-8 corners
e scr.utf8.curvy = true

Show opcode description
e asm.describe = true

Enable pseudo syntax
e asm.pseudo = true

User uppercase syntax
e asm.ucase = true

Enable cache (r/w)
e io.cache = true

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

What is emulation?

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

What is emulation?

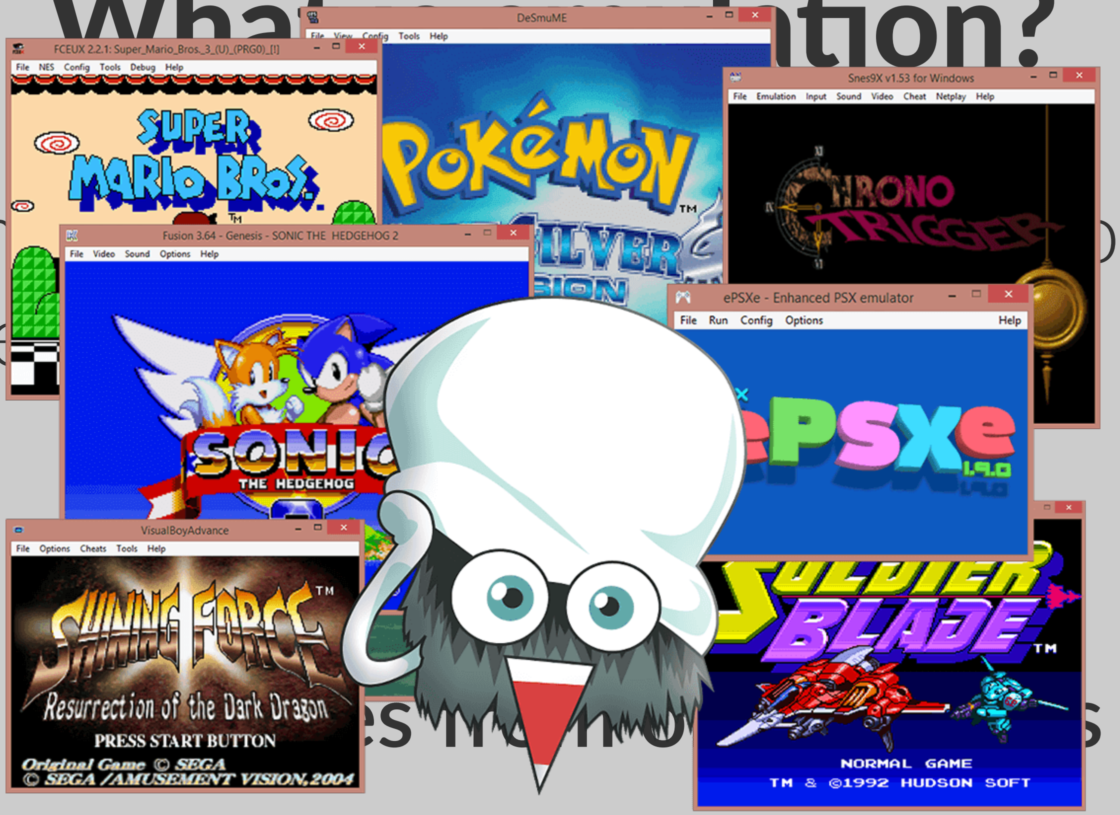
- Simulate the execution of code of the **same or different CPU**

↓

Run **games from old consoles**

What ation?

- Si of
  the
  es s


FCEUX 2.2.1: Super_Mario_Bros._3_(U)_(PRG0)_[!]
File NES Config Tools Debug Help
SUPER MARIO BROS.
DeSmuME
File View Config Tools Help
POKéMON SILVER
Snes9X v1.53 for Windows
File Emulation Input Sound Video Cheat Netplay Help
CHRONO TRIGGER
Fusion 3.64 - Genesis - SONIC THE HEDGEHOG 2
File Video Sound Options Help
SONIC THE HEDGEHOG
ePSXe - Enhanced PSX emulator
File Run Config Options Help
ePSXe 1.9.0
VisualBoyAdvance
File Options Cheats Tools Help
SHINING FORCE
Resurrection of the Dark Dragon
PRESS START BUTTON
Original Game © SEGA
© SEGA /AMUSEMENT VISION,2004
SOLDIER BLADE
NORMAL GAME
TM & ©1992 HUDSON SOFT

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?

- **E**valuable **S**trings **I**ntermediate **L**anguage
- 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

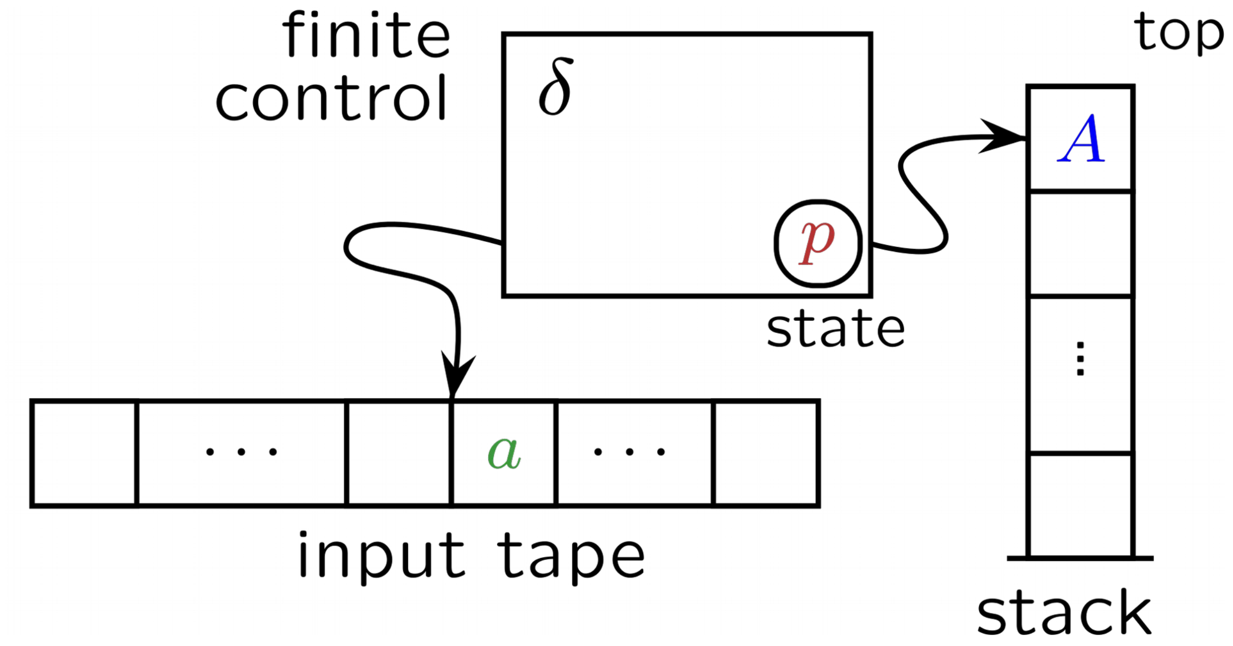
Why ESIL?

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

ESIL

Stack machine on steroids

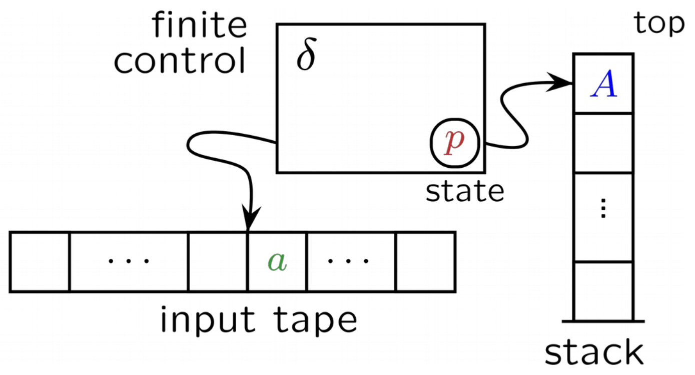
Stack machines / PDA's



Stack machines / PDA's

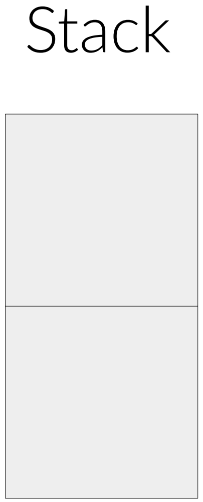
- input symbol
- current state
- stack symbol

- state transition
- manipulate stack (push/pop)

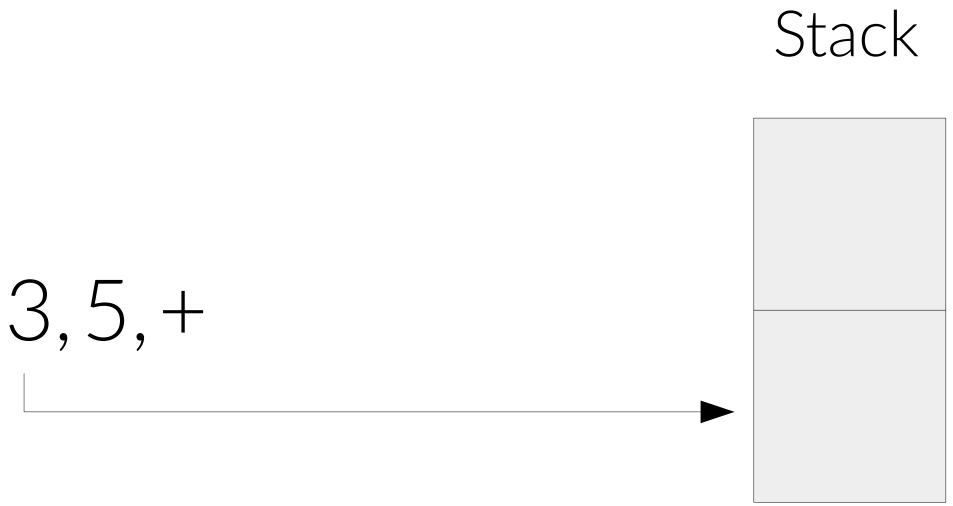


Visual animation

$3, 5, +$

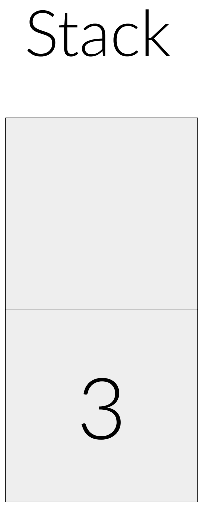


Visual animation

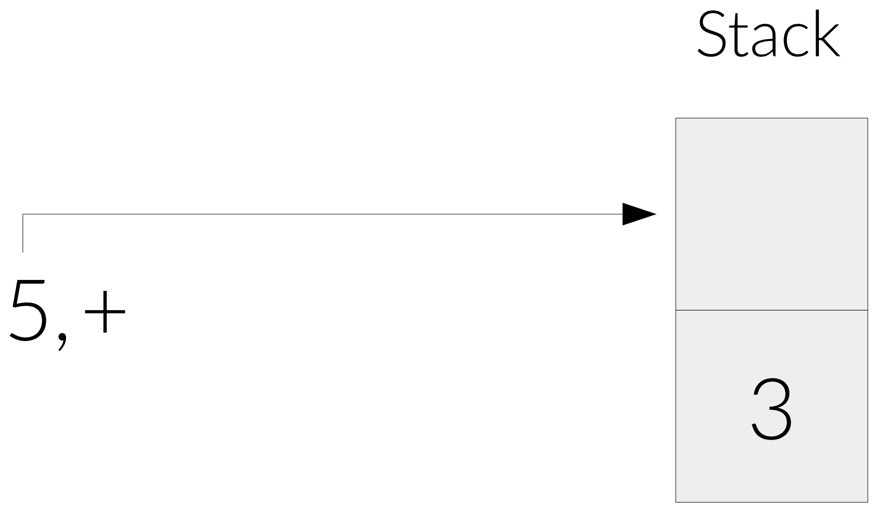


Visual animation

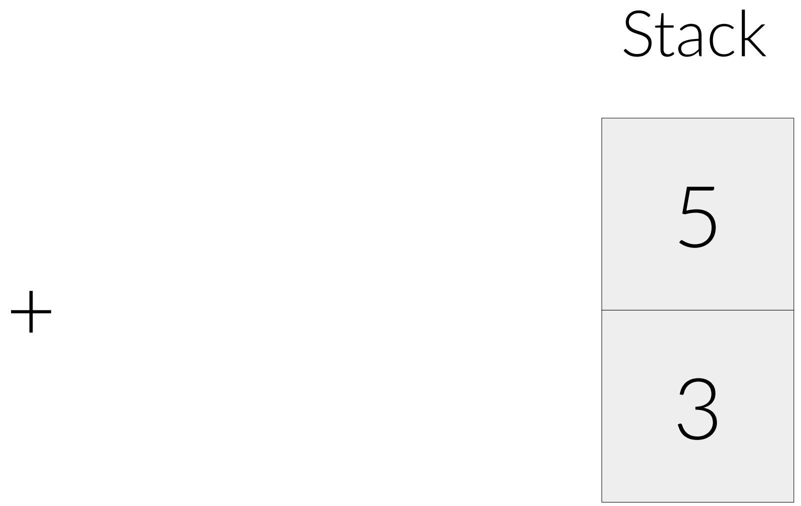
$5, +$



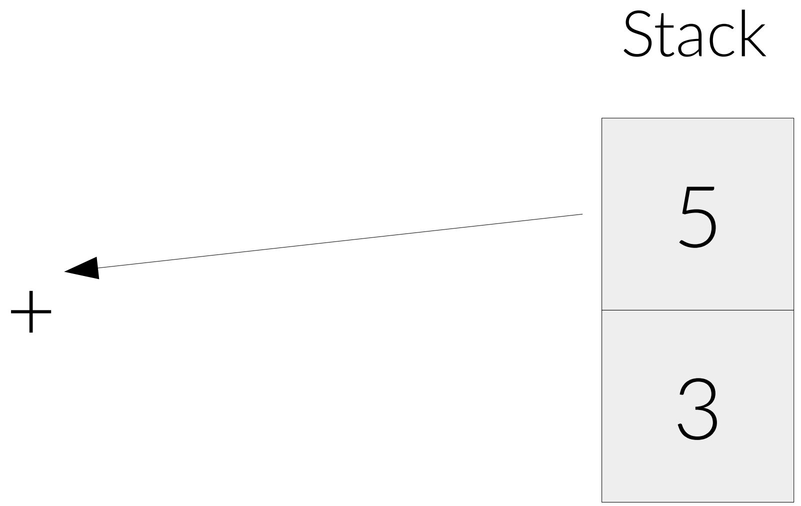
Visual animation



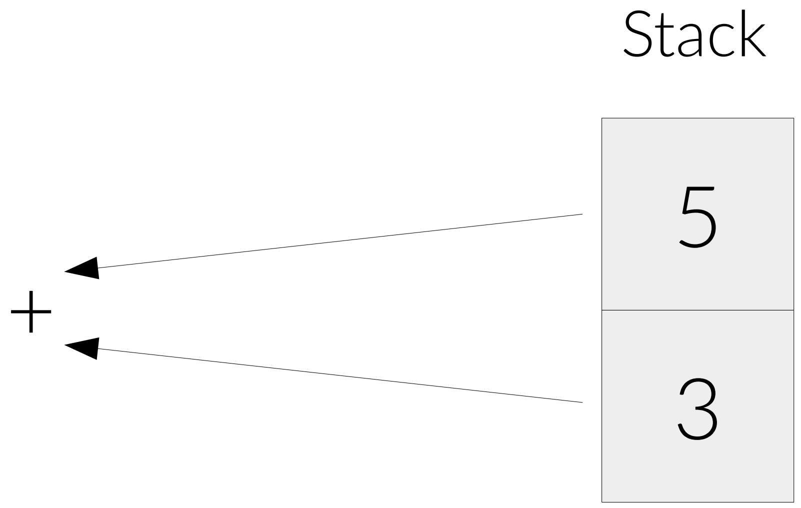
Visual animation



Visual animation



Visual animation



Visual animation

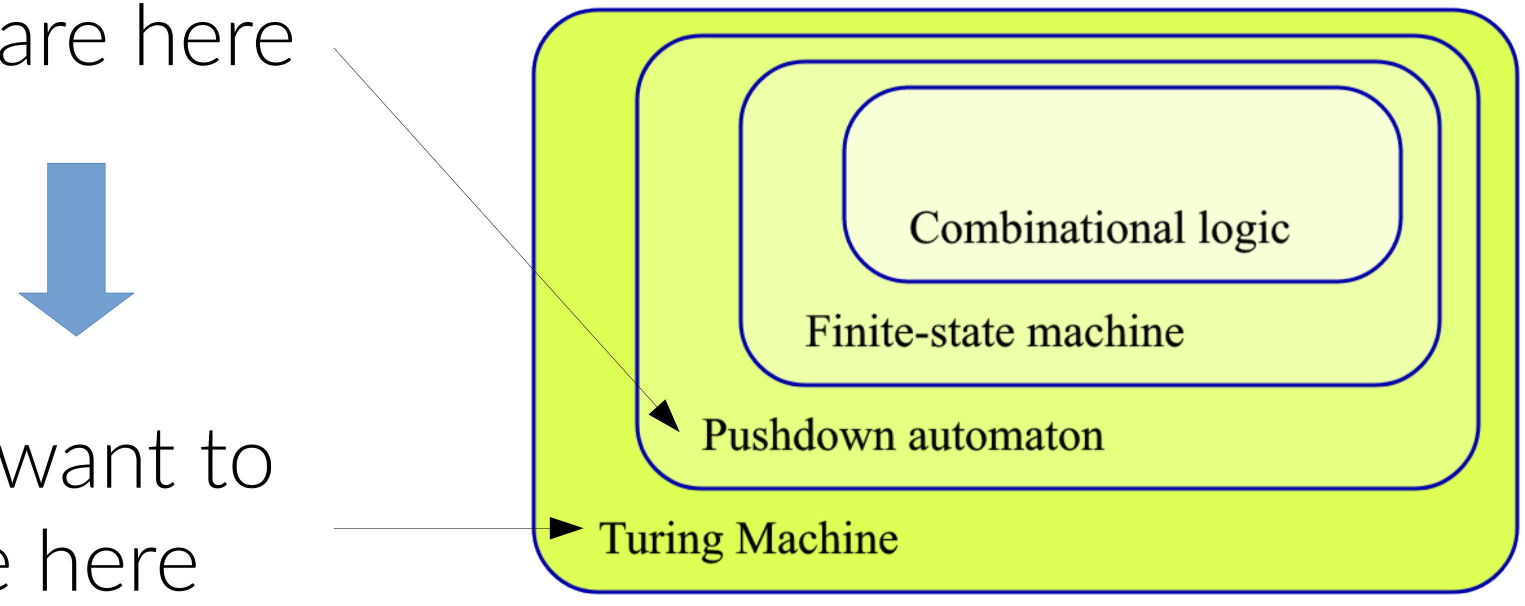
Example

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)

x	y	z	a	b	c	...

Basic practical usage

ESIL options are under **ae** (**a**nalysis **e**sil) subcommands

ae***i***	-	***i***nit
ae***im***	-	***i***nit ***m***emory
ae***ip***	-	***i***nst. ***p***ointer
ae***s***	-	***s***tep
ae***su***	-	***s***tep ***u***ntil
ae***so***	-	***s***tep ***o***ver
ae***ss***	-	***s***tep ***s***kip
ae***r***	-	***r***egisters

ESIL operands

Check ***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

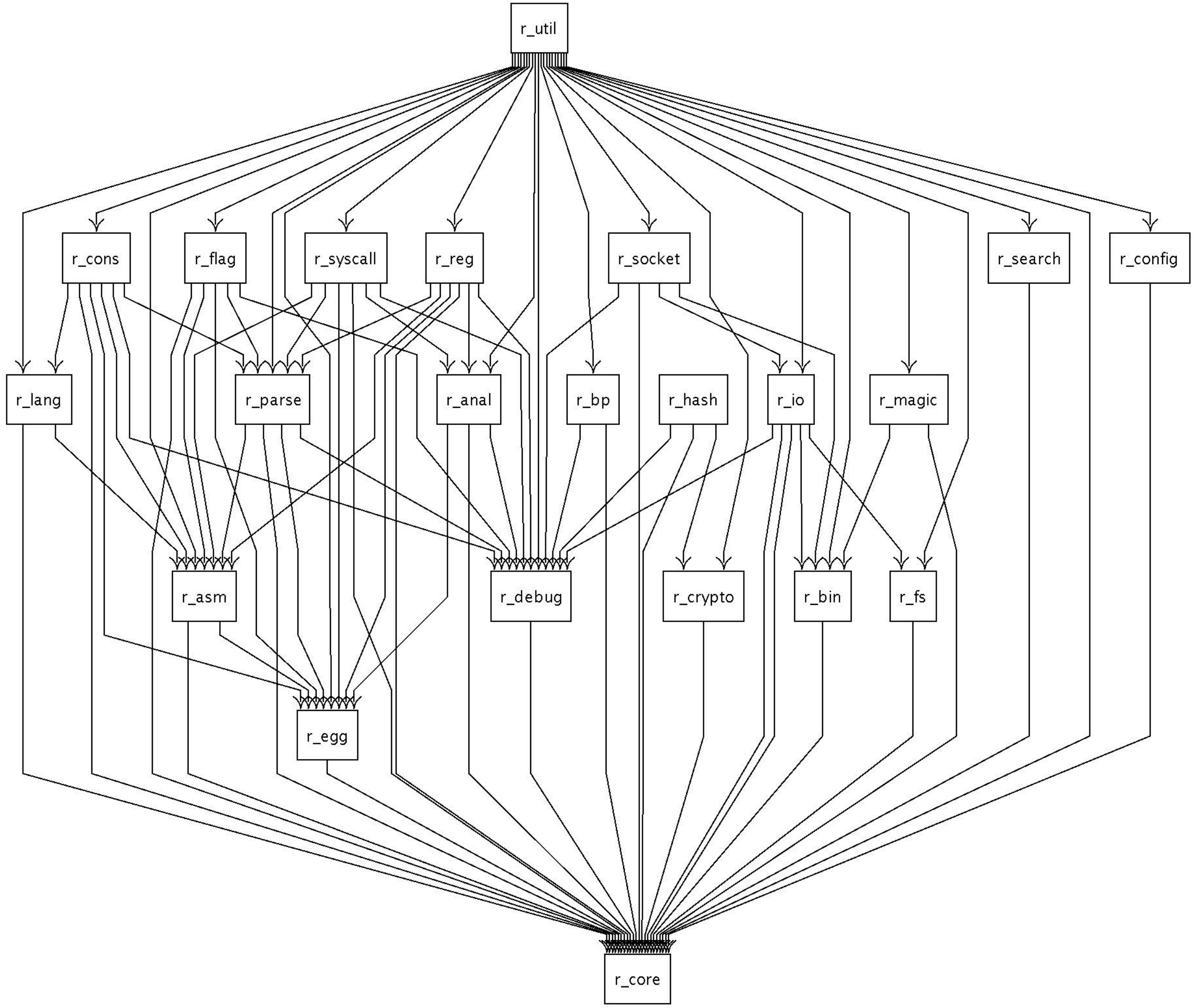
Outline

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


r_util
r_cons
r_flag
r_syscall
r_reg
r_socket
r_search
r_config
r_lang
r_parse
r_anal
r_bp
r_hash
r_io
r_magic
r_asm
r_debug
r_crypto
r_bin
r_fs
r_egg
r_core

Plugins

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

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
  - …

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

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

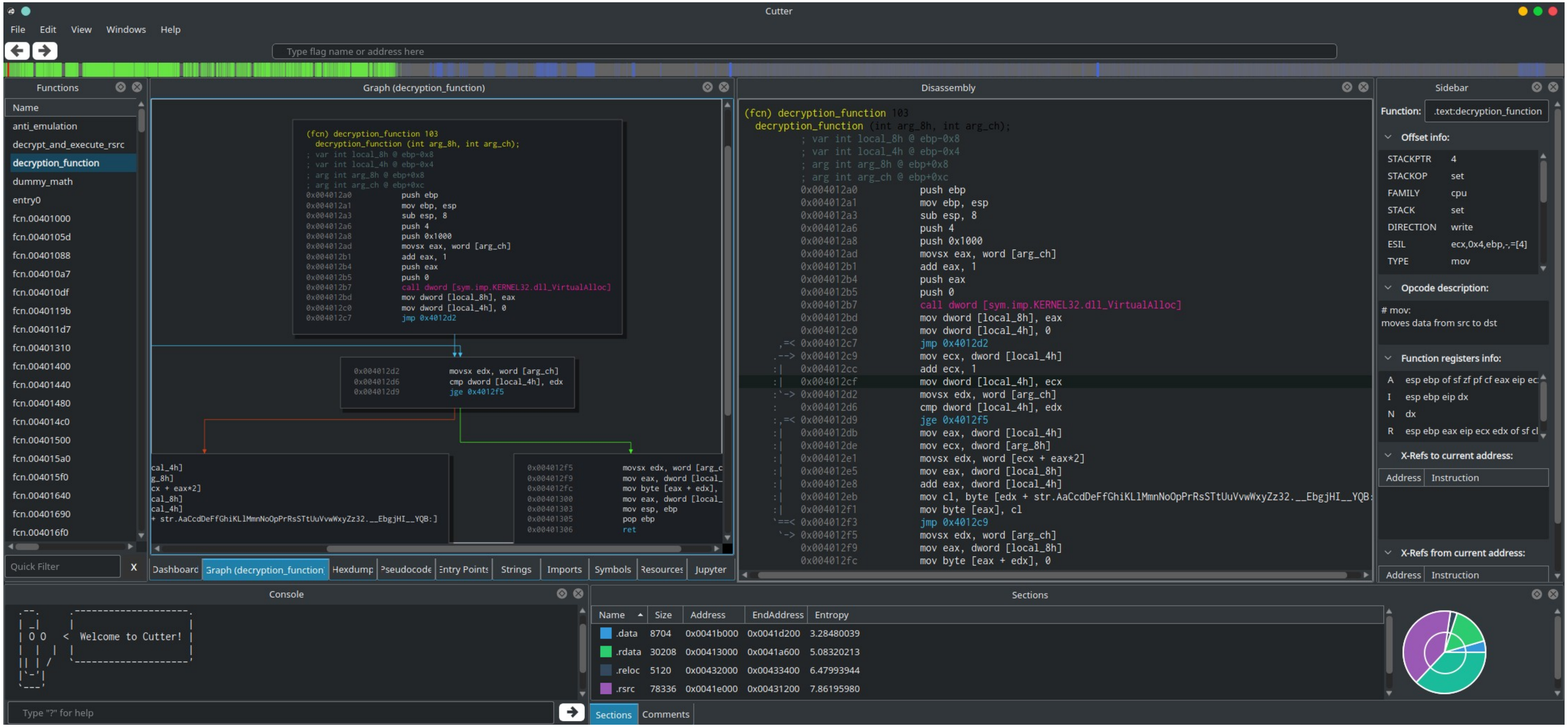
Exploiting

- 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

Cutter: r2 official GUI

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



Functions:

```
Name
anti_emulation
decrypt_and_execute_rsrc
decryption_function
dummy_math
entry0
fcn.00401000
fcn.0040105d
fcn.00401088
fcn.004010a7
fcn.004010df
fcn.0040119b
fcn.004011d7
fcn.00401310
fcn.00401400
fcn.00401440
fcn.00401480
fcn.004014c0
fcn.00401500
fcn.004015a0
fcn.004015f0
fcn.00401640
fcn.00401690
fcn.004016f0
```

Graph (decryption_function):

```
(fcn) decryption_function 103
  decryption_function (int arg_8h, int arg_ch);
; var int local_8h @ ebp-0x8
; var int local_4h @ ebp-0x4
; arg int arg_8h @ ebp+0x8
; arg int arg_ch @ ebp+0xc
0x004012a0          push ebp
0x004012a1          mov ebp, esp
0x004012a3          sub esp, 8
0x004012a6          push 4
0x004012a8          push 0x1000
0x004012ad          movsx eax, word [arg_ch]
0x004012b1          add eax, 1
0x004012b4          push eax
0x004012b5          push 0
0x004012b7          call dword [sym.imp.KERNEL32.dll_VirtualAlloc]
0x004012bd          mov dword [local_8h], eax
0x004012c0          mov dword [local_4h], 0
0x004012c7          jmp 0x4012d2

0x004012d2          movsx edx, word [arg_ch]
0x004012d6          cmp dword [local_4h], edx
0x004012d9          jge 0x4012f5

cal_4h]
g_8h]
cx + eax*2]
cal_8h]
cal_4h]
+ str.AaCcdDeFfGhiKLlMmnNoOpPrRsSTtUuVvwWxyZz32.__EbgjHI__YQB:]

0x004012f5          movsx edx, word [arg_c
0x004012f9          mov eax, dword [local_
0x004012fc          mov byte [eax + edx],
0x00401300          mov eax, dword [local_
0x00401303          mov esp, ebp
0x00401305          pop ebp
0x00401306          ret
```

Disassembly:

```
(fcn) decryption_function 103
  decryption_function (int arg_8h, int arg_ch);
          ; var int local_8h @ ebp-0x8
          ; var int local_4h @ ebp-0x4
          ; arg int arg_8h @ ebp+0x8
          ; arg int arg_ch @ ebp+0xc
          0x004012a0          push ebp
          0x004012a1          mov ebp, esp
          0x004012a3          sub esp, 8
          0x004012a6          push 4
          0x004012a8          push 0x1000
          0x004012ad          movsx eax, word [arg_ch]
          0x004012b1          add eax, 1
          0x004012b4          push eax
          0x004012b5          push 0
          0x004012b7          call dword [sym.imp.KERNEL32.dll_VirtualAlloc]
          0x004012bd          mov dword [local_8h], eax
          0x004012c0          mov dword [local_4h], 0
      ,=< 0x004012c7          jmp 0x4012d2
     .--> 0x004012c9          mov ecx, dword [local_4h]
     :|   0x004012cc          add ecx, 1
     :|   0x004012cf          mov dword [local_4h], ecx
     :`-> 0x004012d2          movsx edx, word [arg_ch]
     :    0x004012d6          cmp dword [local_4h], edx
     :,=< 0x004012d9          jge 0x4012f5
     :|   0x004012db          mov eax, dword [local_4h]
     :|   0x004012de          mov ecx, dword [arg_8h]
     :|   0x004012e1          movsx edx, word [ecx + eax*2]
     :|   0x004012e5          mov eax, dword [local_8h]
     :|   0x004012e8          add eax, dword [local_4h]
     :|   0x004012eb          mov cl, byte [edx + str.AaCcdDeFfGhiKLlMmnNoOpPrRsSTtUuVvwWxyZz32.__EbgjHI__YQB:
     :|   0x004012f1          mov byte [eax], cl
     `==< 0x004012f3          jmp 0x4012c9
      `-> 0x004012f5          movsx edx, word [arg_ch]
          0x004012f9          mov eax, dword [local_8h]
          0x004012fc          mov byte [eax + edx], 0
```

Sidebar:

Function: .text:decryption_function

Offset info:

STACKPTR	4
STACKOP	set
FAMILY	cpu
STACK	set
DIRECTION	write
ESIL	ecx,0x4,ebp,-,=[4]
TYPE	mov

Opcode description:

mov:
moves data from src to dst

Function registers info:

A	esp ebp of sf zf pf cf eax eip ec
I	esp ebp eip dx
N	dx
R	esp ebp eax eip ecx edx of sf cl

X-Refs to current address:

Address	Instruction

X-Refs from current address:

Address	Instruction

Console:

```
 .--.     .---------------------.
 | _|     |                     |
 | O O   <  Welcome to Cutter!  |
 |  |  |  |                     |
 || | /   `---------------------'
 |`-'|
 `---'
```

Sections:

Name	Size	Address	EndAddress	Entropy
🟦 .data	8704	0x0041b000	0x0041d200	3.28480039
🟩 .rdata	30208	0x00413000	0x0041a600	5.08320213
⬛ .reloc	5120	0x00432000	0x00433400	6.47993944
🟪 .rsrc	78336	0x0041e000	0x00431200	7.86195980

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

r2frida

- Use **frida** as backend for memory access and in-process injection
- Install -> r2pm -ci r2frida
- Open -> r2 frida://
- Use -> Prefix with \ (check \?)

r2frida

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

Demo

r2frida

Outline

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

Outline

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