Code emulation for reverse engineers

A deep dive into radare2's ESIL

Arnau Gàmez i Montolio

MATHS & CS STUDENT · PRESIDENT @HACKINGLLIURE · ORGANIZER #R2CON

@arnaugamez *arnaugamez.com*

Motivation

- Force myself to **understand basics** of emulation and ESIL (what, why & how)
- Provide an **easy intro to ESIL** for people wanting to understand/get into it
- Show its **application** to different scenarios

Outline

0 **Overview of radare2**

1 Emulation

2 Intermediate languages & ESIL

3 ESIL operation

4 Demos

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)

r2con2020 will be free & online -> rada.re/con/2020

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, [NetIFreeIOpen]BSD, Android, iOS, OSX, QNX, Solaris, Haiku, FirefoxOS.

Runs everywhere
Supports everything

Get radare2

Clone repo
$ git clone https://github.com/radareorg/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

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*** – ***s***eek
- ***px*** – ***p***rint he***x***dump
- ***pd*** – ***p***rint ***d***isasm
- ***wx*** – ***w***rite he***x***pairs
- ***wa*** – ***w***rite ***a***sm
- ***aa*** – ***a***nalyse ***a***ll
- ***ia*** – ***i***nfo ***a***ll
- ***q*** – ***q***uit

Append **?** to any command to get **inline help** and available **subcommands**

Handy tricks

- Append ***j*** (***j~{}***) for ***j***son (indented) 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 Hello World

- Temporary seek with **@**

  Example: pd @ main

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

Debugging

- Starts debugging at dyld, not entrypoint
- Low level debugger, not aiming to replace source code debugging
- Many backends: gdb, r2llvm, r2frida...

Basic practical usage

Debugging options are under **d** (**d**ebug) subcommands

- d***b*** – ***b***reakpoint
- d***c*** – ***c***ontinue
- d***s*** – ***s***tep
- d***su*** – ***s***tep ***u***ntil
- d***so*** – ***s***tep ***o***ver
- d***r*** – ***r***egisters

Scripting with r2pipe

- r2pipe API
  - input -> r2 commands
  - output -> r2 output
  - JSON deserialization into native objects

r2pipe: python example

- Installation
  - pip3 install r2pipe
- Usage
  - import r2pipe
  - ***open()***, ***cmd()***, ***cmdj()***, ***quit()***

Outline

0 Overview of radare2

1 **Emulation**

2 Intermediate languages & ESIL

3 ESIL operation

4 Demos

What is emulation?

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

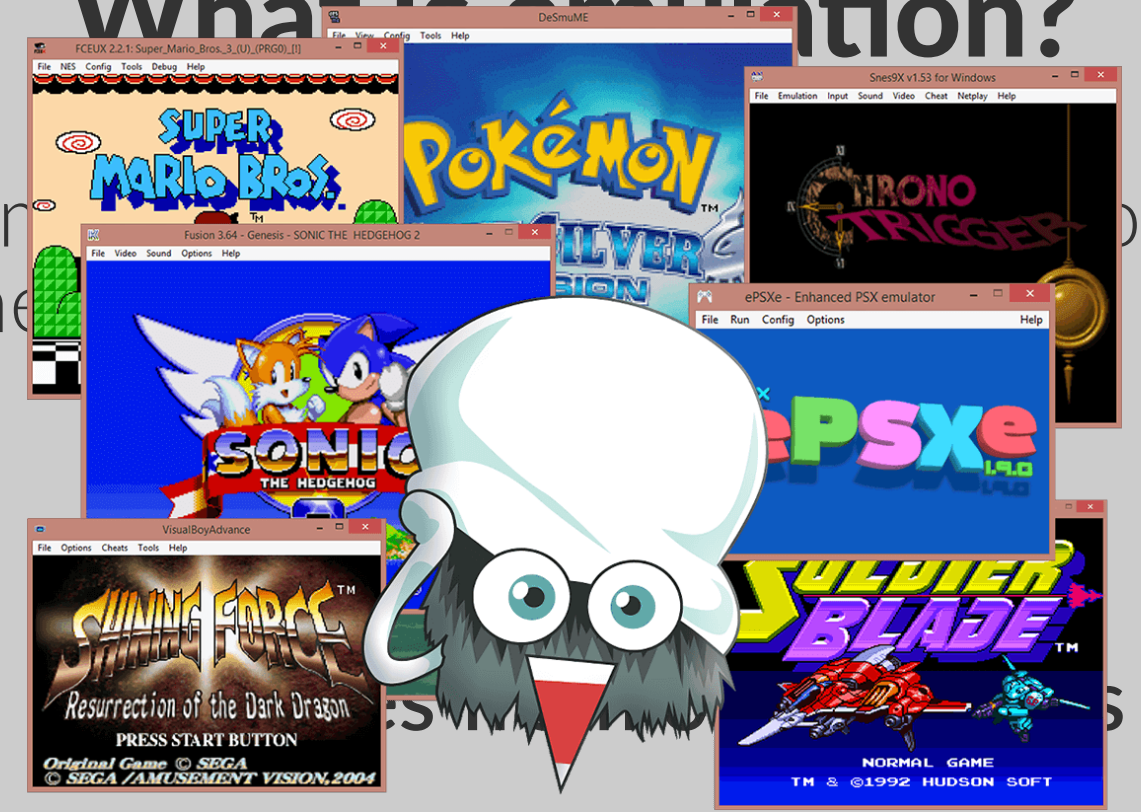
What is emulation?

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

Run **games from old consoles**

What is emulation?

- Sir of
  the



Why emulation?

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

Outline

0 Overview of radare2

1 Emulation

2 **Intermediate languages & ESIL**

3 ESIL operation

4 Demos

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?

Outline

0 Overview of radare2

1 Emulation

2 Intermediate languages & ESIL

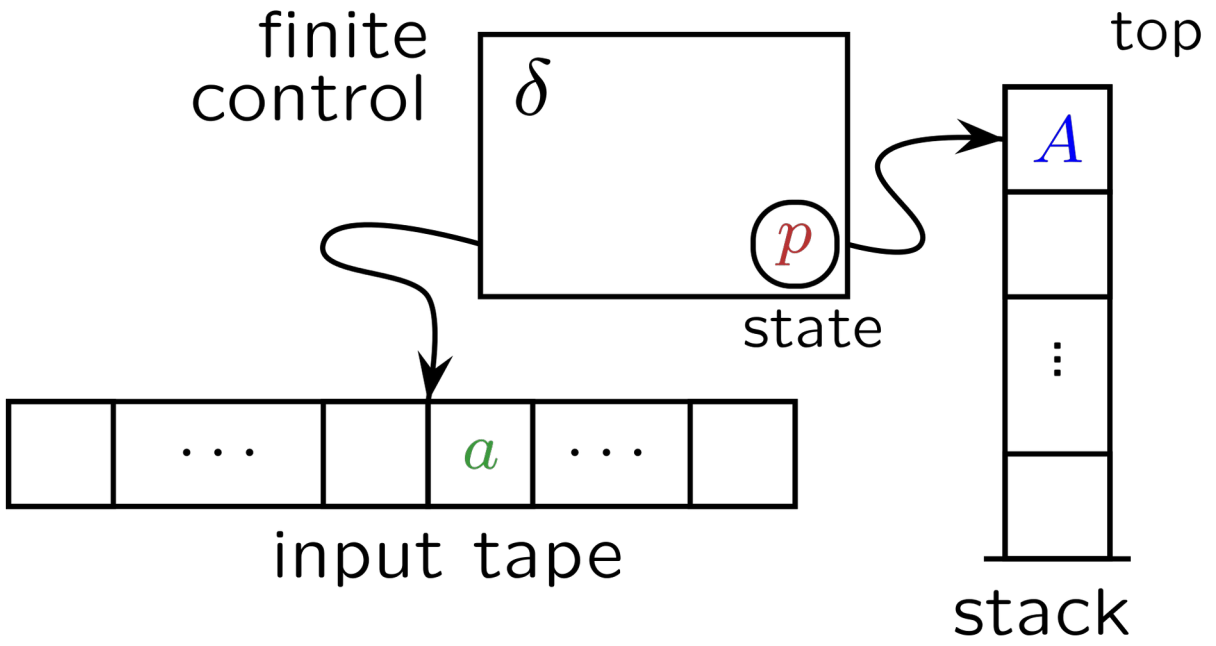
3 **ESIL operation**

4 Demos

ESIL

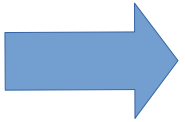
Stack machine on steroids

Stack machines / PDA's

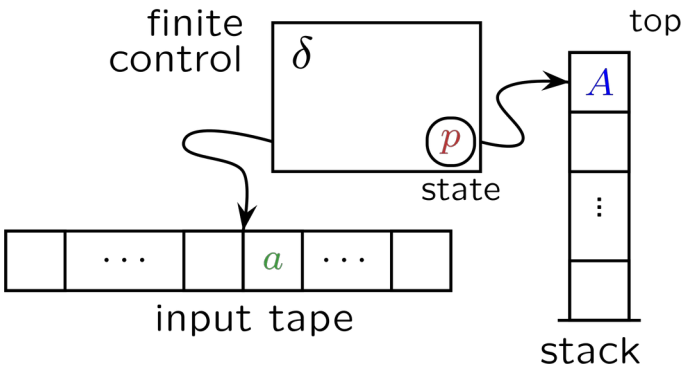


Stack machines / PDA's

- input symbol
- current state
- stack symbol



- state transition
- manipulate stack (push/pop)



Parser idea

```
while not at end of esil_string {
    cur = get_next_element()
    if cur is esil_operation {
        op = get_esil_operation(cur)
        op ()
    } else {
        push (cur)
    }
}
```

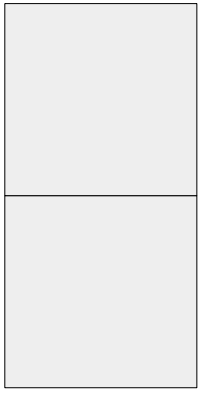
Parser idea

```
while not at end of esil_string {
    cur = get_next_element()
    if cur is esil_operation {
        op = get_esil_operation(cur)
        op ()
    } else {
        push (cur)
    }
}
```

Will pop and use previously pushed symbols as operands

Visual animation

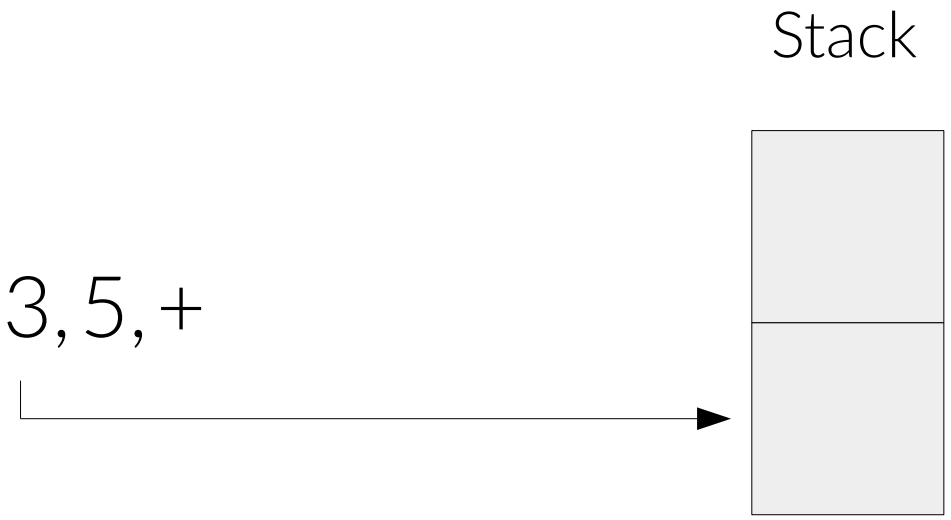
Stack



3, 5, +

Not end of ESIL string → “3” symbol not an operation → So push to stack

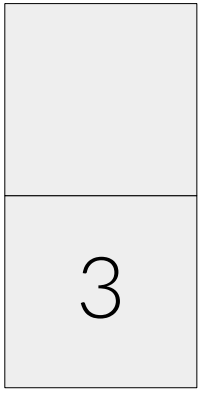
Visual animation



Visual animation

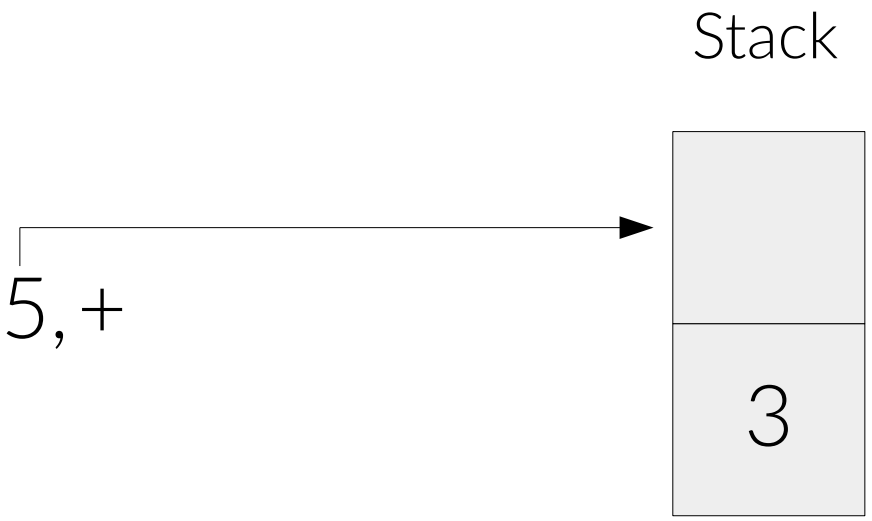
5, +

Stack

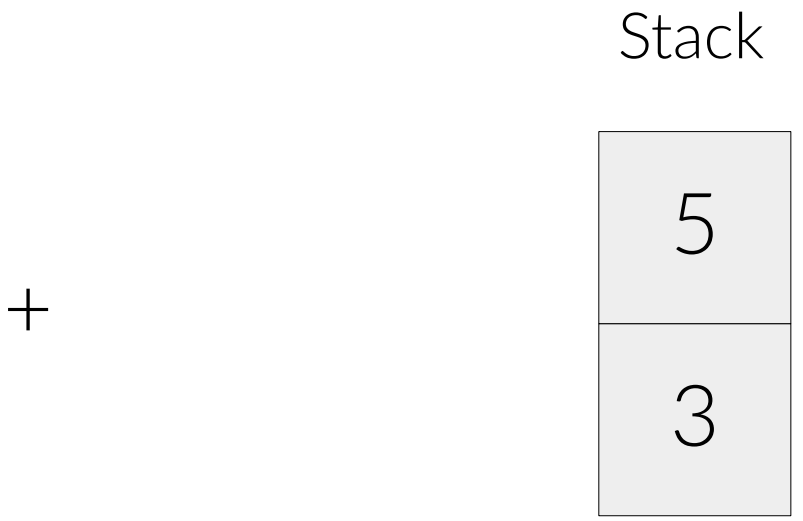


Not end of ESIL string → “5” symbol not an operation → So push to stack

Visual animation

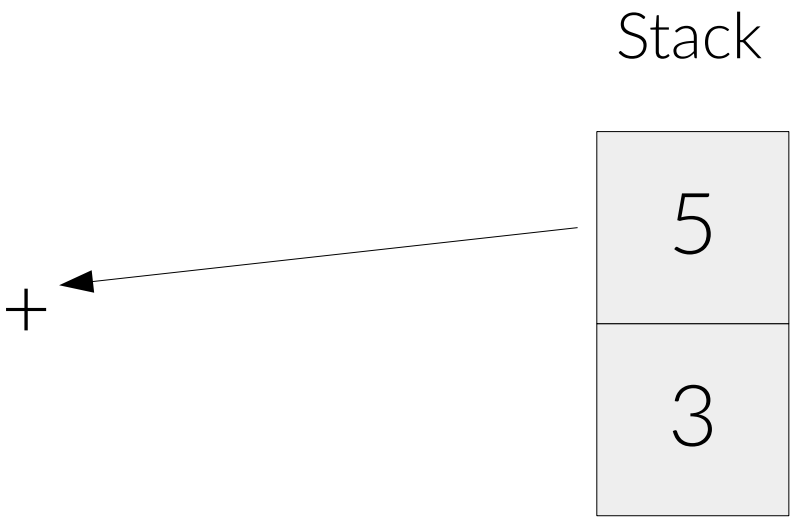


Visual animation



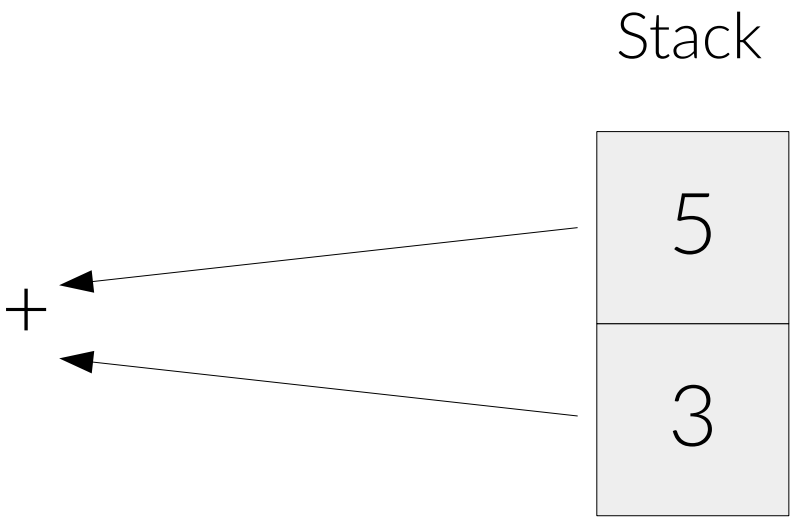
Not end of ESIL string → “+” symbol is an operation → So…

Visual animation



Pop values from stack

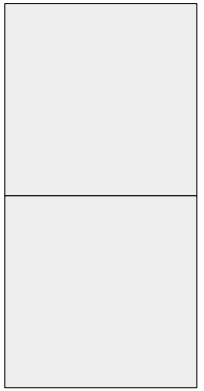
Visual animation



Pop values from stack → Use them as operands

Visual animation

Stack



Pop values from stack → Use them as operands → Perform operation

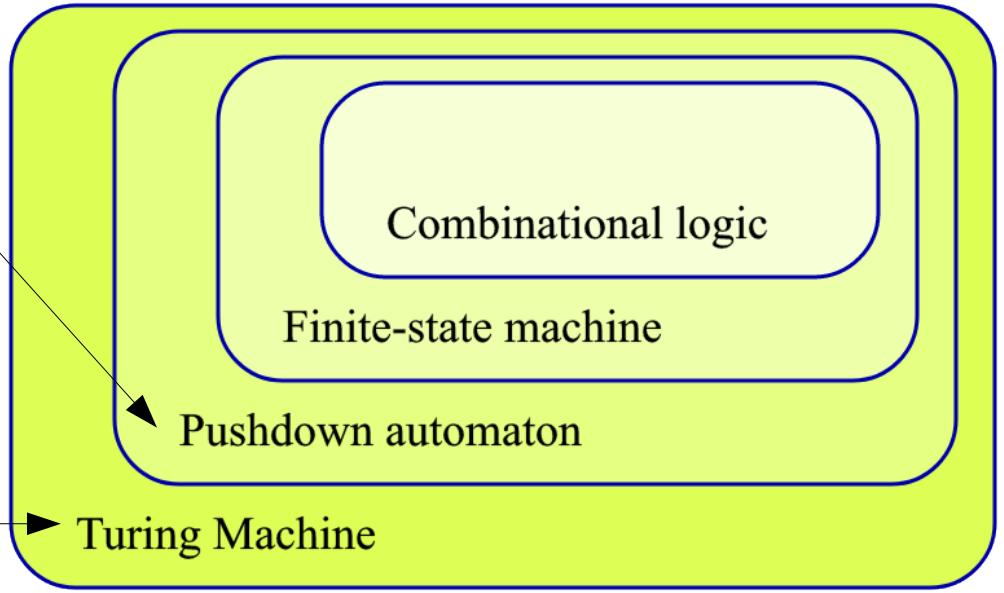
Example

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 ***$***

- $z – zero flag
- $c – carry flag
- …

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

Outline

0 Overview of radare2

1 Emulation

2 Intermediate languages & ESIL

3 ESIL operation

4 **Demos**

Demo

Defeat simple crackme

cc @pof @jvoisin

Demo

Brute force correct input with ESIL & r2pipe

Full writeup at my website

Demo

Deobfuscate encrypted code

cc @superponible

More cool things with ESIL

- r2con2019
  - ESIL applied for Graphs and Analysis (YouTube)
  - Faking Windows Data Structures for ESIL to parse (YouTube)

EOF

HACKING LLIURE

@HackingLliure

@radareorg

me ‹at› arnaugamez ‹dot› com

@arnaugamez