



h-c0n 2020

▶ HackPlayers
c0nference

31-ene y 01-feb

Overcoming fear: reversing with *radare2*

Arnau Gàmez i Montolio | @arnaugamez

▶ Hackplayers c0nference MMXX



ILNI
LA NAVE

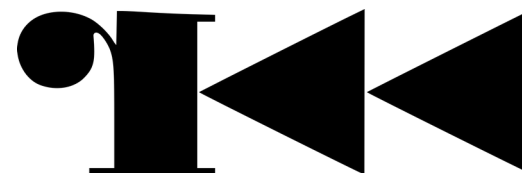


Quien soy



Arnau Gàmez i Montolio - 22 años - Barcelona

- Estudiante - *Maths* & *CS* @ UB
- Presidente - *@HackingLliure*
- Colaborador - *#r2con*





Quien **NO** soy



- Reverser profesional
- Experto en radare2
- Desarrollador de radare2





Motivación



- Desmitificar radare2
- Ofrecer explicaciones sencillas por un usuario no avanzado
- Organizar y compartir conocimiento
- Y... ya me tocaba venir a la h-c0n





Sobre vosotros



- Estudiantes?
- Trabajando en seguridad informática? Ingeniería inversa?
- Conocéis radare2?
- Usáis radare2?





Índice



1 Presentando radare2

2 Comandos & interacción

3 Modos visuales & debugging

4 Configuración & customización

5 Emulación de código con ESIL

6 Extensibilidad & scripting

7 Exploiting

8 Extras

9 Documentación & recursos

10 Conclusiones





Acerca de radare2



- Framework de ingeniería inversa libre y open source
- (Re)escrito en C por pancake
- Hecho desde cero sin dependencias de terceros
- Portable, scriptable, extensible...





Acerca de radare2



- Release cada 6 semanas
- Gran comunidad
- r2con: congreso anual en Barcelona (primera de Setiembre)





Capacidades de radare2



- Desensamblado de binarios de distintas arquitecturas y sistemas operativos
- Análisis de código y datos
- Debugging a bajo nivel y exploiting
- Manipulación de binarios





Capacidades de radare2



- Forense: montar sistemas de archivos, detectar particiones, data carving
- Extracción de métricas para la clasificación de binarios
- Análisis y debugging de kernel





Capacidades de radare2



radare2 tiene soporte para...





Capacidades de radarez



Operating Systems

Windows (since XP), GNU/Linux, OS X, [Net|Free|Open]BSD, Android, iOS, OSX, QNX, Solaris, Haiku, FirefoxOS.

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.





Capacidades de radarez



Corre en *todos* lados
Tiene soporte para *todo*





Instalar radare2



Clonar repositorio

```
$ git clone --depth=1 https://github.com/radareorg/radare2
```

Ir al directorio radare2 creado

```
$ cd radare2
```

Instalar / update *(hace pull automáticamente de la última versión)*

```
$ ./sys/install.sh
```





Instalar radare2



**KEEP
CALM
AND
USE R2
FROM GIT**





Herramientas incluidas



- rax2 -> conversor de base
- rabin2 -> extraer información de binarios
- rasm2 -> (dis)assembler
- rahash2 -> utilidad de crypto/hashing
- radiff2 -> diffing de binarios





Herramientas incluidas



- ragg2 -> crear pequeños binarios (shellcode)
- rarun2 -> configurar entorno de ejecución
- rafind2 -> búsqueda en binarios
- r2pm -> gestor de paquetes de radare2
- radare2 -> herramienta principal (shell)





Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Abriendo la shell de r2



*r2 es un enlace simbólico a **radare2***

Cargar archivo

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

No cargar preferencias de usuario

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

Cargar archivo en modo escritura

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

Alias de r2 malloc://512

```
$ r2 -
```

Cargar archivo en modo debug

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

Abrir r2 sin cargar ningún archivo

```
$ r2 --
```





Comandos básicos



Los comandos en r2 están basados en **mnemónicos**

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





Comandos básicos



Añade **?** a cualquier comando para obtener **ayuda inline** sobre estos y sus **subcomandos** disponibles





Algunos trucos prácticos



- Añade **j** (**j~{}**) para output **j**son (intentado)

Ejemplo: izj, izj~{}

- Añade **q** para **q**uiet output

Ejemplo: izq

- Grep interno con **~**

Ejemplo: iz~string





Algunos trucos prácticos



- Pipe a comandos de la shell nativa

Ejemplo: `iz | less`

- Ejecuta comandos de la shell nativa con el prefijo **!**

Ejemplo: `!echo h-c0n ftw`

- Seek temporal con **@**

Ejemplo: `pd @ main`





Demo



Interacción básica con la shell de radare2



hackplayers.com

Hackplayers conference

@arnaugamez



Práctica



Uso básico de radare2

IOLI crackme0x00

IOLI crackme0x01

- **Objetivo 1:** Conseguir la contraseña correcta
- **Objetivo 2:** Parchear el binario para que acepte cualquier contraseña





Demo (fun fact)



FLARE-On 2019

Challenge 2: Overlong

*Unlucky strings
radare2 FTW*



hackplayers.com

Hackplayers conference

@arnaugamez



Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Modo visual



- Accede al modo visual con el comando **V**
 - Rotar (print) modo con el comando **p**
 - Pulsa **?** para obtener ayuda del modo visual
 - Usa **:** para ejecutar comandos r2





Vista de grafo



- Accede a la vista de grafo con el comando **VV**
 - Seguir el flujo de las funciones
 - Es necesario estar *seeked* en una función
 - Navega con **hjkl**
 - Zoom in/out con **+/-**





Vista en paneles



- Accede a la vista en paneles con el comando **V!**
 - Muy útil para debugging
 - Paneles por defecto bastante útiles
 - Personaliza al gusto la vista de los paneles





Debugging



- Las opciones de debugging se encuentran como subcomandos de **d** (debug)
- Empieza el debugging en el dyld, no en el entrypoint
- Debugger a bajo nivel. No aspira a reemplazar debugging a nivel de código fuente
- Distintos backends: gdb, r2llvm, r2frida...





Debugging



- **db** – **b**reakpoint
- **dc** – **c**ontinue
- **ds** – **s**tep
- **dsu** – **s**tep **u**ntil
- **dso** – **s**tep **o**ver
- **dr** – **r**egisters





Demo



Exploración del modo visual, vista de grafo y vista en paneles



hackplayers.com

Hackplayers conference

@arnaugamez



Práctica



Debugging en radare2

IOLI crackme0x02

- **Objetivo 1:** Conseguir la contraseña correcta
- **Objetivo 2:** Modificar memoria/registros para que acepte cualquier contraseña

(Recordad los modos y vistas descritos)





Índice



- | | |
|--|-------------------------------------|
| 1 Presentando radare2 | 6 Extensibilidad & scripting |
| 2 Comandos & interacción | 7 Exploiting |
| 3 Modos visuales & debugging | 8 Extras |
| 4 Configuración & customización | 9 Documentación & recursos |
| 5 Emulación de código con ESIL | 10 Conclusiones |





Configuración



- Variables de configuración **e**valuables
- Usar comando **e** (y subcomandos) para configurar (y tunear) radare2
- Listar las variables de configuración
 - Show values: **e**
 - Show description: **e??**





Configuración



- Búscalas: ***e??~whatever***
- Lista los valores que puede tomar: ***e conf.var = ?***
- Asigna un nuevo valor: ***e conf.var = new_value***





Algunos ejemplos útiles



Usar chars UTF-8

`e scr.utf8 = true`

Habilitar pseudo sintaxis

`e asm.pseudo = true`

Esquinas curvas

`e scr.utf8.curvy = true`

Usar mayúsculas

`e asm.ucase = true`

Descripción del opcode

`e asm.describe = true`

Habilitar cache (r/w)

`e io.cache = true`





Algunos ejemplos útiles



- Añadir configuraciones (comandos con **e**) al archivo **~/.radare2rc** para que se carguen por defecto
 - La flag -N previene que se cargue la configuración personalizada
- Explora y modifica visualmente las variables de configuración con **Ve**





Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Emulación: ¿Qué?



- Simular la ejecución de código de la **misma o distinta CPU**



Ejecuta **juegos de consolas antiguas**





Emulación: ¿Por qué?



- **Entender** un snippet de código específico
- **Evitar los riesgos** de la ejecución nativa de código
- Asistir al **debugging** y **análisis de código**
- Explorar **ejecutables no nativos**





Lenguajes intermedios (IL)



*"Lenguaje de una **máquina abstracta** diseñada para ayudar a realizar el análisis de un programa informático" -- wikipedia*



Vital para (de)compilación





ESIL: ¿Qué?



- **E**valuable **S**trings **I**ntermediate **L**anguage
- Conjunto de instrucciones reducido
- Basado en notación polaca inversa (stack)
- Diseño pensado para **emulación y evaluación**, no para ser leído por humanos





ESIL: ¿Qué?



- Infinita memoria y conjunto de registros
- Alias "nativos" para los registros
- Posibilidad de implementar **instrucciones personalizadas** y llamar a funciones externas





ESIL: ¿Por qué?



- Necesidad de emulación en r2land
- Fácil de generar, parsear y modificar
- Extensible
- ¿Por qué no?





ESIL



Stack machine con esteroides



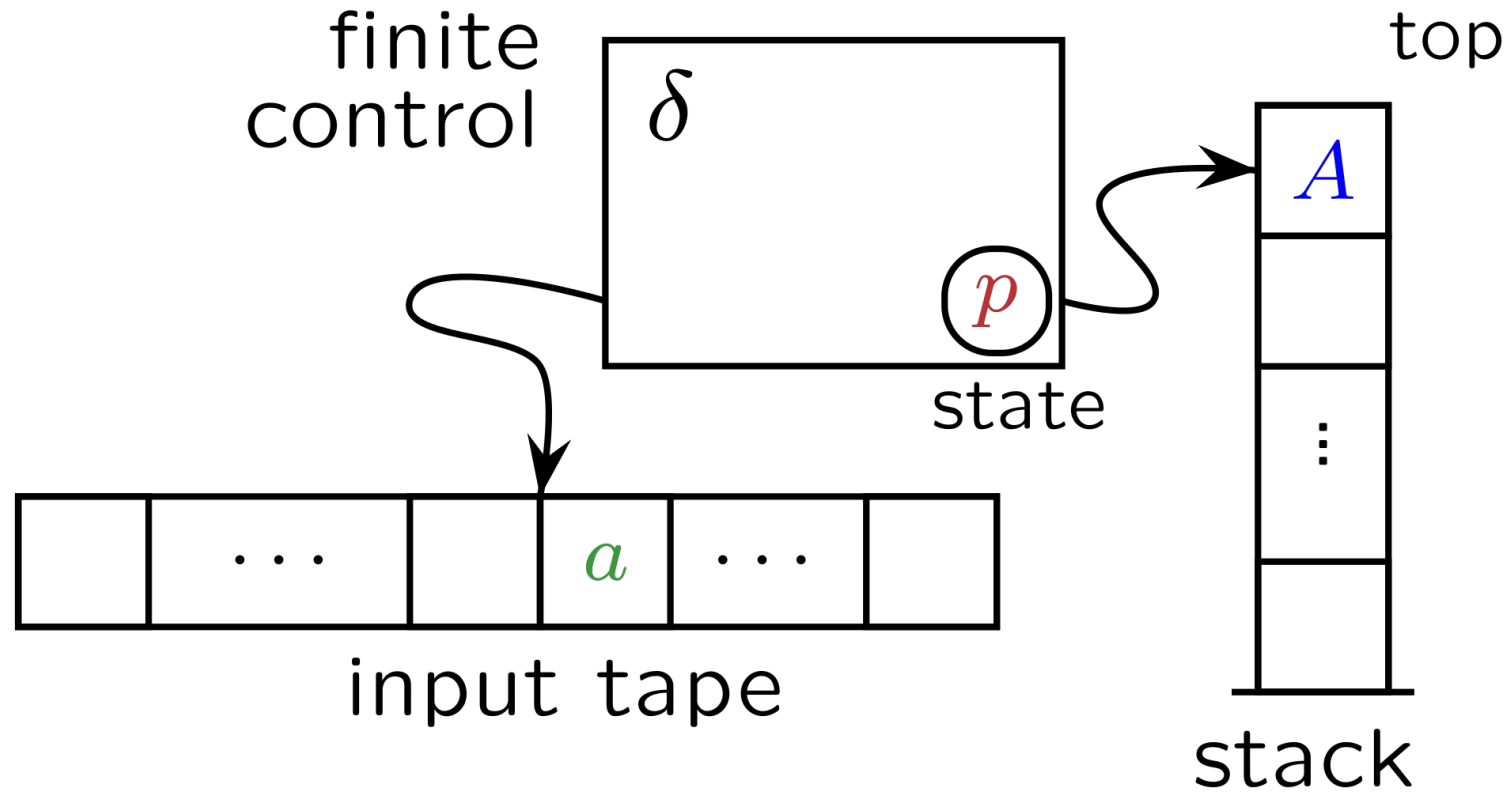
hackplayers.com

Hackplayers conference

@arnaugamez



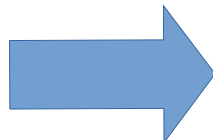
Stack machines - PDAs

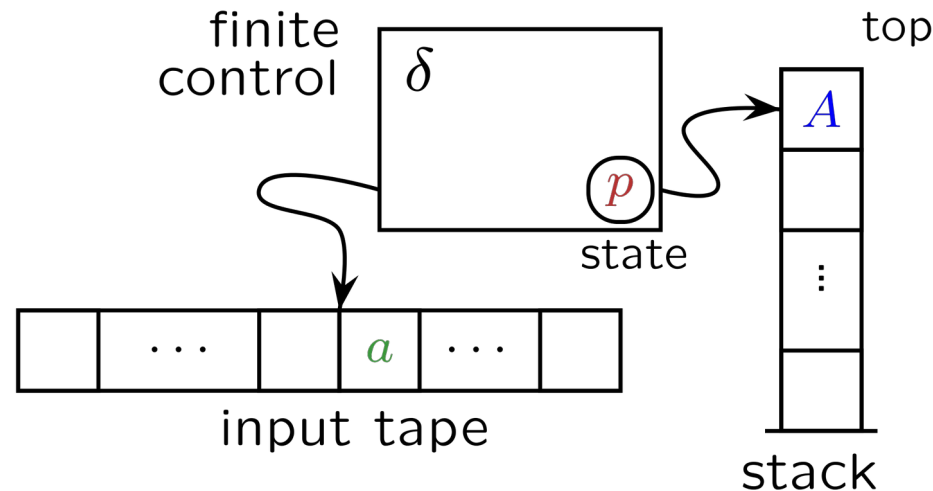




Stack machines - PDAs



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





Animación visual



3, 5, +

Stack





Animación visual



3, 5, +



Stack





Animación visual



5, +

Stack





Animación visual





Animación visual



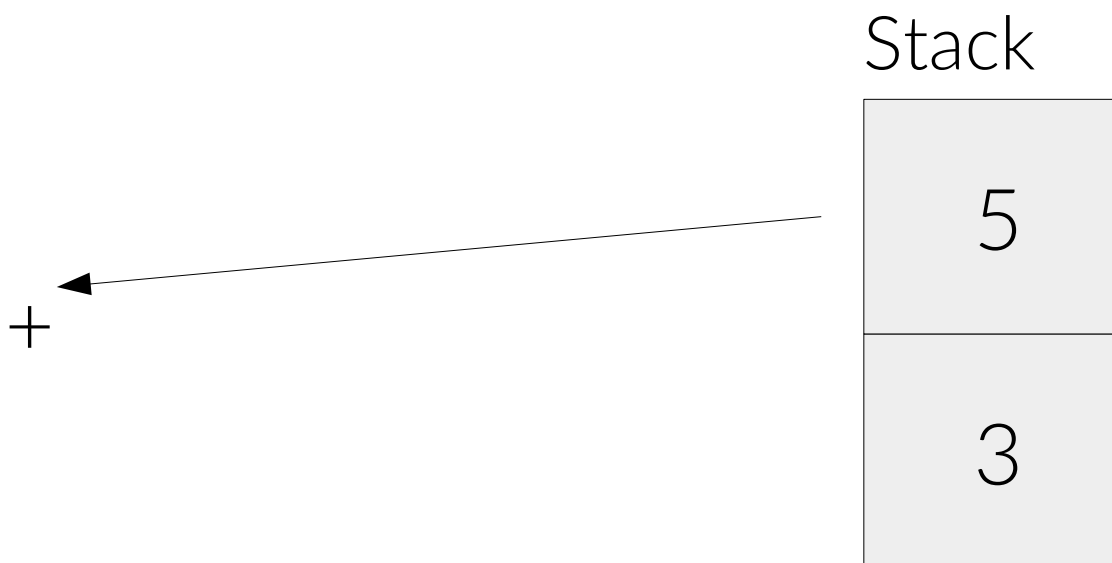
+

Stack



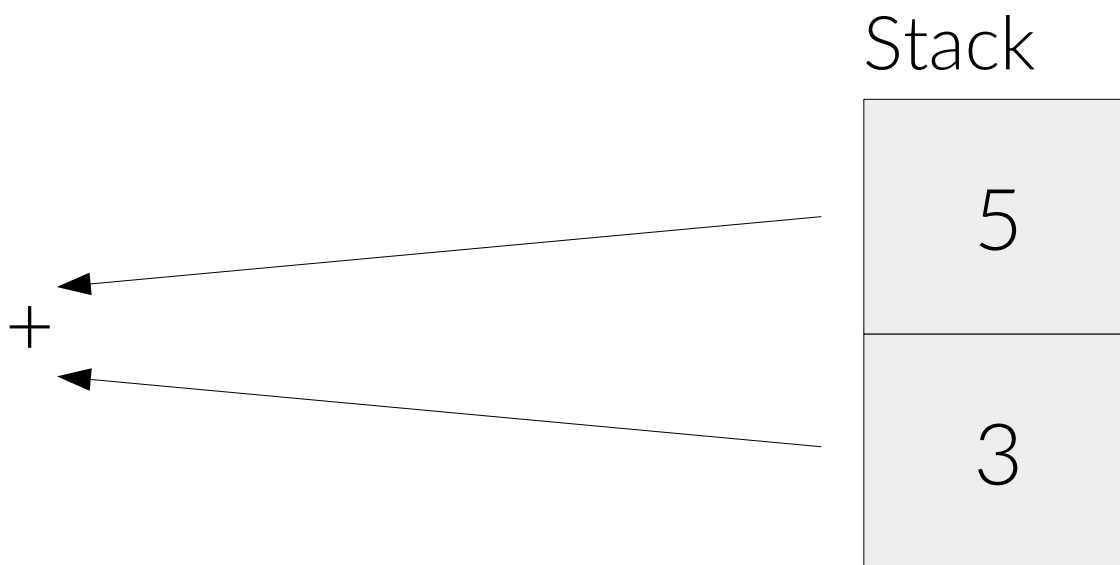


Animación visual





Animación visual

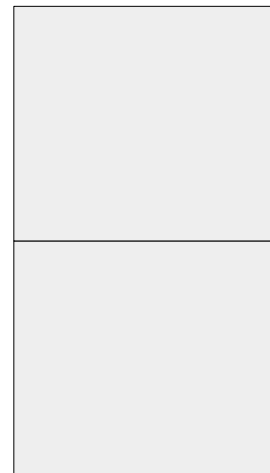




Animación visual



Stack





Ejemplo



ae 3,5,+





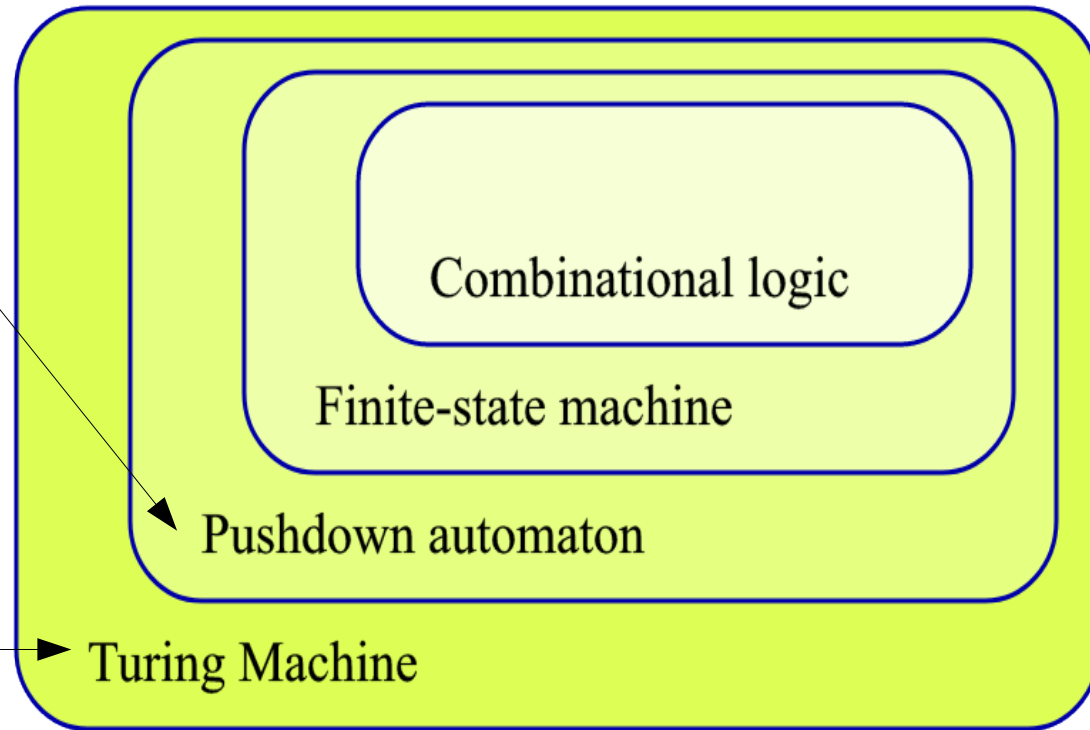
Expandiendo stack machines



Estamos aquí



Queremos
llegar aquí



cc @condr3t





Expandiendo stack machines



¿Como?





Expandiendo stack machines



¿Como?



ESTEROIDES

(aka cheating)





Esteroideos Vol.1



- Añadir operaciones de **acceso aleatorio**
- Añadir operaciones de **control de flujo**





Esteroideos Vol.2



- Acceso a **registros**
- Añadir una "**cinta extra**" con acceso aleatorio (memoria virtual, VM stack)

x	y	z	a	b	c	...
---	---	---	---	---	---	-----





Uso a la práctica



Las opciones de ESIL se encuentran como subcomandos de **ae** (**a**nalysis **e**sil)

- 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





Operandos y flags de ESIL



Los podéis encontrar con *ae??*

(descripcion y ejemplos)





Práctica



Uso básico de ESIL

IOLI crackme0x02

IOLI crackme0x03

- **Objetivo:** Conseguir la contraseña correcta usando emulación con ESIL





Demo



Desofuscación de código con ESIL

Defender.exe



hackplayers.com

Hackplayers conference

@arnaugamez



Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Estructura

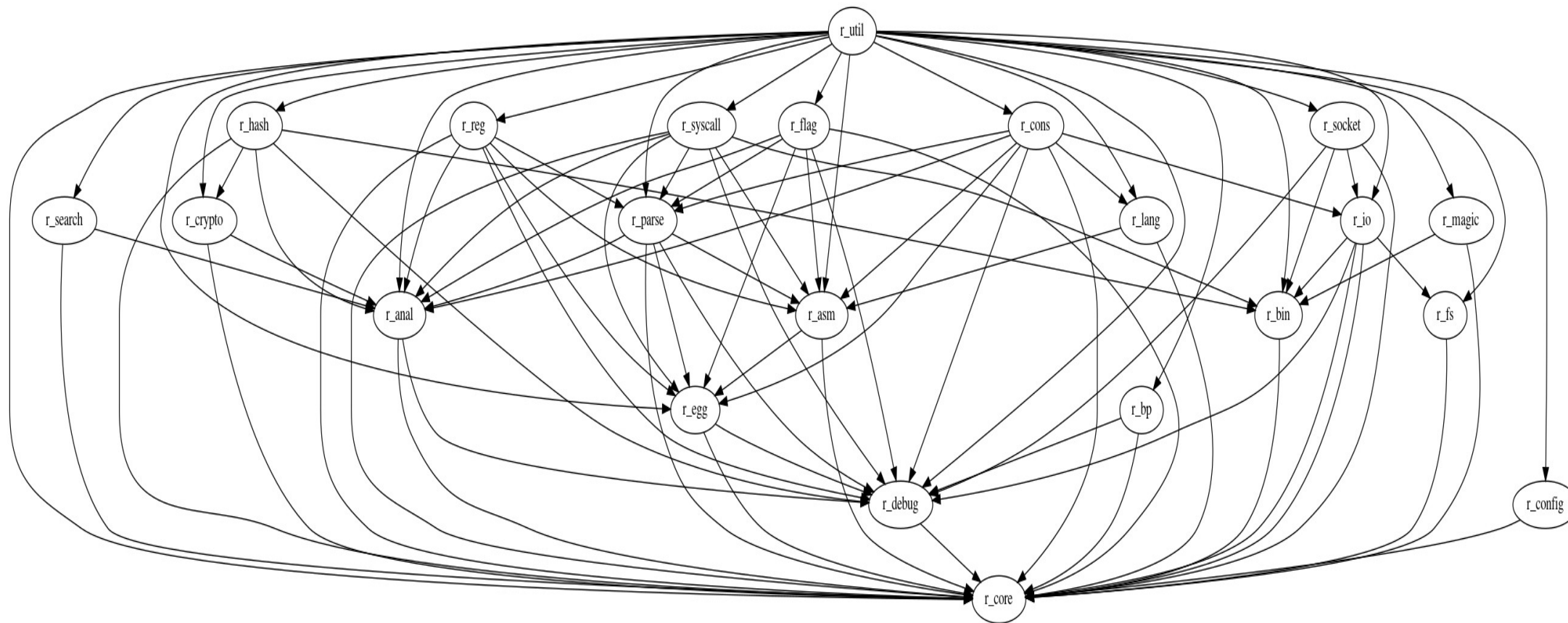


- radare2 está compuesto por distintas librerías
- Programas *standalone* (herramientas incluídas) construídas en base a una o varias de estas librerías





Estructura



make depgraph.png





Estructura



- `libr/` -> módulos
 - `[lib]/p` -> plugins para cada módulo
- `binr/` -> binarios
- `shlr/` -> código ripeado de terceros





Plugins



- Plugins

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





Plugins



- Instalar/gestionar plugins via r2pm
 - Inicializar pkg manager -> **r2pm init**
 - Instalar plugin -> **r2pm -i [plugin]**
- Check *man r2pm*





Scripting



- Bindings para muchos lenguajes:
 - Java
 - Go
 - NodeJS
 - Python
 - ...





Scripting



- r2pipe API
 - input -> r2 commands
 - output -> r2 output
 - Deserialización de JSON a objetos nativos





Scripting: python



- Instalación
 - `pip(3) install r2pipe`
- Uso
 - `import r2pipe`
 - `open()`, `cmd()`, `cmdj()`, `quit()`





Demo



Deserialización de JSON a objetos nativos



hackplayers.com

Hackplayers conference

@arnaugamez



Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7 Exploiting**
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Exploiting



- Buscar strings -> / [string]
- Buscar ROP gadgets -> /R
- Encontrar xrefs a funciones -> axt [offset]
- Encontrar secciones w/x sections -> iS





Exploiting



- Listar (libc) imports -> is~imp
- De Bruijn pattern -> ragg2 -P [size] -r
- Encontrar offset del pattern -> wopO [value]
- Craftear shellcode -> ragg2 -a [arch]
-b [bits] code.c





Exploiting



- Referencias:
 - <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/>





Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Cutter



- GUI oficial de radare2
- Desarrollado en C++ & QT
- Releases junto a las releases de radare2
- Check <https://cutter.re>





Cutter



File Edit View Windows Debug Help

Graph (fcn.00002630)

Disassembly (unsynced)

Hexdump (unsynced)

Functions

Name	Size
-entry.fini0	65
-entry.init0	153
-entry0	46
-fcn.00002000	27
-fcn.00002130	41
-fcn.00002200	863
-fcn.00002630	162
-fcn.000026e0	268
-fcn.000027f0	4876
-fcn.0000344e	3643
-fcn.000034e1	4630
-fcn.0000355c	4544
-fcn.000035ce	4544
-fcn.00003b00	6933
-fcn.00004240	193
-fcn.00004470	1063
-fcn.00004950	222
-fcn.00004ac0	82
-fcn.00004b20	246
-fcn.00004cd0	41
-fcn.00004d20	56
-fcn.00004d60	136
-fcn.00004df0	102
-fcn.00004e60	96
-fcn.00004ec0	60
-fcn.00004f90	66

Quick Filter

30 Items

Graph Overview

Dashboard Graph (fcn.00002630) Strings Imports Search

(fcn) fcn.00002630 162
fcn.00002630 (int32_t arg1);
; arg int32_t arg1 @ rdi
push rbx
test rdi, rdi
je 0x26ae

mov rbx, rdi
mov esi, 0x2f
call qword [reloc.strchr]
test rax, rax
je 0x269e

lea r8, [rax + 1]
mov rdx, r8
sub rdx, rbx
cmp rdx, 6
jle 0x269e

lea rsi, [rax - 6]
mov ecx, 7
repz cmpsb byte [rsi], byte ptr [rdi]
seta dl
sbb dl, 0
test dl, dl
jne 0x269e

mov ecx, 3
lea rdi, [0x00006380]
mov rsi, r8
mov rbx, r8
repz cmpsb byte [rsi], byte ptr [rdi]
seta dl
sbb dl, 0

lea rdi, [0x00006080]
call qword [reloc.bindtextdomain]
lea rdi, [0x00006080]
call qword [reloc.textdomain]
lea rdi, [0x00002500]
call fcn.000050c0
mov rbx, qword [rbx + 8]
lea rsi, str.help
mov rdi, rbx
call qword [reloc.strcmp]
test eax, eax
je 0x20f3
lea rsi, str.version
mov rdi, rbx
call qword [reloc.strcmp]
test eax, eax
je 0x20c5
xor eax, eax
pop rbx
ret
xor r9d, r9d
lea r8, str.Jim.Meyering
mov rcx, qword [str.8.31]
lea rdx, str.GNU.coreutils
lea rsi, str.true
mov rdi, qword [obj.stdout]
call fcn.00004950
jmp 0x20c1
xor edi, edi
call fcn.00002200
nop word [rax + rax]

endbr64
xor ebp, ebp
mov r9, rdx
pop rsi
mov rdx, rsp
and rsp, 0xffffffffffffff0
push rax
push rsp
lea r8, [r8]
lea rcx, [rcx]
lea rdi, [rdi]

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
00 ff 15 31 6f 00 00 48 6d 35 43 48 00 00 48 8d 3d 13 40 00 00 ff 15 5d 6e 00 00 48 6d 3d 05 4e
00 00 ff 15 50 6e 00 00 78 8d 3d 11 04 00 00 67 e8 2b 30 00 00 48 8b 5b 03 48 6d 35 23 40 00 00
48 89 df ff 15 27 60 00 00 85 c0 74 46 48 8d 35 16 40 00 00 48 89 df ff 15 93 60 00 00 85 c0 74
84 31 c0 5b c3 45 31 c9 4c 8d 05 05 40 00 00 48 8b 0d 3a 6f 00 00 48 8d 15 a7 3f 00 00 48 8d 35
20 3f 00 00 48 8b 3d 9d 6f 00 00 67 e8 5f 28 00 00 eb ce 31 ff e8 06 01 00 00 66 0f 1f 44 00 00
f3 0f 1e fa 31 ed 49 89 d1 5e 48 89 e2 48 83 e4 f0 50 54 4c 8d 05 96 2f 00 00 48 8d 0d 1f 2f 00 00
00 48 8d 3d 18 ff ff ff ff 15 02 6e 00 00 f4 90 48 8d 3d 49 6f 00 00 48 8d 05 42 6f 00 00 48 39
f8 74 15 48 8b 05 76 6d 00 00 48 85 c0 74 09 ff e0 0f 1f 80 00 00 00 c3 0f 1f 80 00 00 00 00
48 8d 3d 19 6f 00 00 48 8d 35 12 6f 00 00 48 29 fe 48 c1 fe 03 48 89 f0 48 c1 e8 3f 48 01 c6 48
d1 fe 74 14 48 8b 05 4d 6e 00 00 48 85 c0 74 08 ff e0 66 0f 1f 44 00 00 c3 0f 1f 80 00 00 00 00
f3 0f 1e fa 80 3d fd 6e 00 00 75 33 55 48 83 3d 3a 6e 00 00 48 89 e5 74 0d 48 8b 3d 46 6e
00 00 ff 15 28 6e 00 00 e8 63 ff ff ff c6 05 04 6e 00 00 01 5d c3 66 2e 0f 1f 84 00 00 00 00 00
c3 66 2e 0f 1f 84 00 00 00 00 0f 1f 40 00 03 0f 1e fa e9 67 ff ff 0f 1f 80 00 00 00 00 00



hackplayers.com

Hackplayers conference

@arnaugamez



Decompilación



- r2dec
 - Conversor de asm a pseudo-C escrito en JS
 - <https://github.com/wargio/r2dec-js>
- r2retdec
 - Bridge entre r2 & retdec
 - <https://github.com/securisec/r2retdec>





Decompilación



- radeco
 - *Aspira(ba) "the r2 decompiler"*
 - Escrito en Rust. Usa ESIL como input
 - Archivado, almenos de momento :(
 - <https://github.com/radareorg/radeco>





Decompilación



- r2ghidra-dec
 - Integración del decompiler de ghidra en r2
 - Basado solo en el decompiler de ghidra (en C++), por lo que no requiere tener ghidra instalado
 - Viene por defecto en las nuevas versiones de Cutter





r2frida



- Plugin para usar **frida** como backend para acceso a memoria e inyección in-process
- Instalar -> `r2pm -ci r2frida`
- Cargar -> `r2 frida://`
- Uso -> Con prefijo `\` (check `\?`)





r2frida



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



hackplayers.com

Hackplayers conference

@arnaugamez



Demo



Idea general de r2frida



hackplayers.com

Hackplayers conference

@arnaugamez



Práctica



Reto tipo CTF

Sonda (thegame @ HackUPC2019)

- **Objetivo:** Conseguir la flag (aviso: es bastante rara)
- **Nota:** Combinad y probad distintos recursos que hemos ido viendo: vista por grafos, ESIL, scripting, decompilación...





Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Documentación escrita



- "Ya está documentado en C"
- Libro oficial de radare2
 - <https://radare.gitbooks.io/radare2book>
 - Actualizado regularmente con las contribuciones de la comunidad





Más recursos escritos



- radare2 explorations
 - <https://monosource.gitbooks.io/radare2-explorations>
- Blogs
 - <http://radare.today>
 - <https://megabeets.net>





Otros recursos



- Charlas grabadas
 - r2con2016
 - r2con2017
 - r2con2018
 - r2con2019 (en proceso...)
 - Otras muchas decenas en YouTube y otros.





Tips extras



- Recuerda añadir ? para ayuda *inline*
- Quick trick
 - Búsqueda interactiva en la ayuda -> ?*~...
- Quick trick++
 - **alias r2help="r2 -q -c '?*~...' -"**





Soporte



- IRC
 - #radare @ irc.freenode.net
- Telegram
 - <https://t.me/radare>

IRC & Telegram están bridged





Índice



- 1** Presentando radare2
- 2** Comandos & interacción
- 3** Modos visuales & debugging
- 4** Configuración & customización
- 5** Emulación de código con ESIL
- 6** Extensibilidad & scripting
- 7** Exploiting
- 8** Extras
- 9** Documentación & recursos
- 10** Conclusiones





Conclusiones



- radare2 no es *tan* difícil
 - Comandos basados en mnemónicos
 - UNIX-like shell
 - Menos de 10 comandos para realizar la mayoría de tareas
 - Ayuda *inline* añadiendo ?





Conclusiones



- Hay muchas formas distintas de contribuir a proyectos open source como radare2
 - Código
 - Documentación
 - Reportar issues
 - Usar y compartir radare2



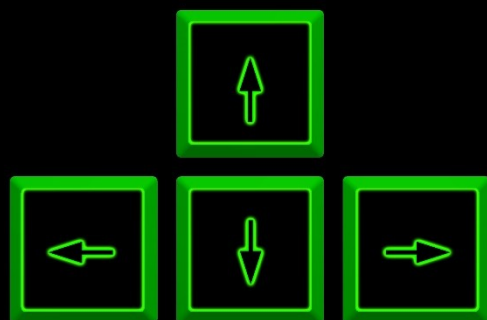


Invitación



- r2con2020
 - De la comunidad para la comunidad
 - Trainings y charlas.
 - Del 2 al 5 Setiembre @ Barcelona (fechas por confirmar)





> h - c@n

Hackplayers c@nference

Sending SIGKILL to all processes.

Please stand by while rebooting the system.

[64857.521348] sd 0:0:0:0: [sda] Synchronizing SCSI cache

[64857.522838] Restarting system.

—

www.h-c@n.com