

Radare2 - The Dwarf Fortress of reversing

Who needs a GUI anyway?

Florent (Skia) Jacquet Julien (jvoisin) Voisin

November 18, 2016

GreHack 2016

```
[0x00000000]> pf.  
pf.skia zzzzz name school where r2_since workshops  
[0x00000000]> pf.skia  
    name : 0x00000000 = Florent (Skia) Jacquet  
    school : 0x00000017 = UTBM (looking for internship)  
    where : 0x00000035 = Belfort, France  
    r2_since : 0x00000045 = RSoC 2014  
    workshops : 0x0000004f = First one!  
[0x00000000]> □
```

Who needs the source code anyway?

Playground

```
[0x7f51f92b1cd0 150 /usr/bin/id]> ?0;f tmp;s.. @ rip
0100 0000 0000 0000 bd15 cd80 fc7f 0000 .....
0000 0000 0000 0000 c915 cd80 fc7f 0000 .....
d815 cd80 fc7f 0000 ea15 cd80 fc7f 0000 .....
0816 cd80 fc7f 0000 5416 cd80 fc7f 0000 .....T.....
orax 0x0000003b          rax 0x00000000          rbx 0x00000000
rcx 0x00000000          rdx 0x00000000          r8 0x00000000
r9 0x00000000           r10 0x00000000         r11 0x00000000
r12 0x00000000          r13 0x00000000         r14 0x00000000
r15 0x00000000          rsi 0x00000000         rdi 0x00000000
rsp 0x7ffc80ccfe10      rbp 0x00000000         rip 0x7f51f92b1cd0
rflags I

;-- rip:
4889e7          mov rdi, rsp
e8a83f0000     call 0x7f51f92b5c80
4989c4          mov r12, rax
8b050f312200   mov eax, dword [0x7f51f94d4df0]
5a             pop rdx
488d24c4       lea rsp, [rsp + rax*8]
29c2          sub edx, eax
52             push rdx
4889d6          mov rsi, rdx
4989e5          mov r13, rsp
4883e4f0       and rsp, 0xfffffffffffffffff0
488b3d463322.  mov rdi, qword [0x7f51f94d5040]
```

How to radare2?

Installing

- Shipped with many distributions
- Don't even think about using the package manager!
- Install from git, and `git pull` every day

```
git clone https://github.com/radare/radare2 &&  
cd radare2 && ./sys/install.sh
```

A modular framework

In a randomized order:

- rabin2
- rasm2
- rax2
- radiff2
- rahash2
- rafind2
- rarun2
- **radare2**
- ...

rabin2 - Find informations about binaries

```
$ rabin2 -e file    # Show entrypoints
```

```
$ rabin2 -i file    # Show imports
```

```
$ rabin2 -zz file   # Show strings
```

```
$ rabin2 -g file    # Show everything
```

rasm2 - Assemble/disassemble

Assemble

```
$ rasm2 -a arm -b 32 'mov r0, 0x42' 4200a0e3
```

Disassemble

```
$ rasm2 -a x86 -b 32 -d 4200a0e3'  mov r0, 0x42
```

List available asm plugins

```
$ rasm2 -L
```

Output in C format

```
$ rasm2 -a arm -b 32 'mov r0, 0x42' -C  
"\x42\x00\xa0\xe3"
```

rax2 - Base converter and calculator

```
$ rax2 1977
```

```
0x7b9
```

```
$ rax2 0xfa0 101010b 14
```

```
4000 0x2a 0xe
```

```
$ rax2 -s 72616461726532
```

```
radare2
```

```
$ rax2 "0xfa0+101010b*14"
```

```
4588
```

radiff2 - Unified binary diffing

Code diffing

```
$ radiff2 /bin/true /bin/false
```

Code diffing using graphdiff algorithm

```
$ radiff2 -C /bin/true /bin/false
```

put '-C -A' for analysing before diffing

```
$ radiff2 -g main /bin/true /bin/false
```

Graph diff at given symbol (also try to give offsets: '0x0ff1,0x0ff2')

rahash2 - Block based hashing

Display hashes of the whole file with all algorithms

```
$ rahash2 -a all file
```

Display md5 per block of 1024

```
$ rahash2 -B -b 1024 -a md5 file
```

Display entropy per block of 1024

```
$ rahash2 -B -b 1024 -a entropy file
```

Display md5 of given string

```
$ rahash2 -a md5 -s "string"
```

Search for string

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

Continue to search even when read-error occurs

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

Display results as hexdump

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

rarun2 - Run programs in exotic environments

Sample rarun2 script

```
#!/usr/bin/rarun2
program=./pp400
arg0=10
stdin=foo.txt
chdir=/tmp
clearenv=true
setenv=EGG=eggsy
setenv=NOFUN=nogames
unsetenv=NOFUN
# EGG will be the only env variable
```

Run with

```
$ ./script.rr2 or $ rarun2 script.rr2
```

The radare2 shell

Getting a shell

```
$ r2 -           # Open r2 with a chunk of zero'd memory
$ r2 --          # Open r2 with no file
$ r2 /bin/ls      # Open /bin/ls in r2
$ r2 -d /bin/ls   # Open /bin/ls in debug mode
```

Getting help in the shell

Type ?

Getting help in the shell

Type ?

```
[0x00000000]> ?
Usage: [.] [times] [cmd] [-grep] [[@iter] addr!size] [>pipe] ; ...
Append '?' to any char command to get detailed help
Prefix with number to repeat command N times (f.ex: 3x)
!%var =valueAlias for 'env' command
| *off=[0x]value      Pointer read/write data/values (see ?v, wx, wv)
| (macro arg0 arg1)   Manage scripting macros
| .[-](m)|f|!sh|cmd|  Define macro or load r2, cparse or rlang file
| = [cmd]              Run this command via rap://
| /                    Search for bytes, regexps, patterns, ..
| ! [cmd]              Run given command as in system(3)
| #!lang [...]         Hashbang to run an rlang script
| a[?]                Perform analysis of code
| b[?]                Get or change block size
| c[?] [arg]           Compare block with given data
| C[?]                Code metadata management
| d[?]                Debugger commands
| e [a=[b]]            List/get/set config evaluable vars
| f [name][sz][at]     Set flag at current address
| g [arg]              Go compile shellcodes with r_egg
| i[?] [file]          Get info about opened file
| k[?] [sdb-query]     Run sdb-query. see k? for help, 'k *', 'k **' ...
| m                   Mountpoints commands
| o[?] [file] ([offset]) Open file at optional address
| p[?] [len]           Print current block with format and length
| P[?]                Project management utilities
| q[?] [ret]           Quit program with a return value
| r[?] [len]           Resize file
| s[?] [addr]          Seek to address (also for '0x', '0x1' == 's 0x1')
| S[?]                Io section manipulation information
| t[?]                Cparse types management
| T[?] [-] [num][msg]  Text log utility
| u[?]                uname/undo seek/write
| V                   Enter visual mode (vcmds=visualvisual  keystrokes)
| w[?] [str]           Multiple write operations
| x[?] [len]           Alias for 'px' (print hexadecimal)
| y[?] [len] [[[@]addr] Yank/paste bytes from/to memory
| z[?]                Signatures management
| ?[??][expr]         Help or evaluate math expression
| ??                 Show available '$' variables and aliases
| @@                 Misc help for '@' (seek), '~' (grep) (see ~??)
| ??                 List and manage core plugins
[0x00000000]> █
```

Getting help in the shell

- Append `?` after every command to get help
Some command support multiple `?` (try `pf???`)
- Every character has a meaning:
`pdf`: `p`rint `d`isassemble `f`unction
- The first character is the most general:
`a`nalyse, `i`nformation, `p`rint, `w`rite...
- Then you get subsets of commands, up to five characters!
(`afvrs`)
- Try also `?@?` to get help about particular `r2` shell syntax

Common command sets

- **a** Analyse
- **s** Seek (move around the file)
- **/** Search
- **i** Informations (rabin2)
- **d** Debugger
- **p** Print
- **w** Write

Some useful commands

- **aaa** Analyse most of the file
- **pdf** Print disassembly of the current function
- **pf** Print formatted data (mostly for dumps and headers)

Visual mode - An interactive view

V in cli mode to enter visual mode

- **p/P** to rotate modes
- **h j k l** to move around
- **o** to seek directly to an offset, a tag, a hit...
- **e** for interactive configuration of r2
- **_** to open HUD and see every object that r2 knows
- **V** opens ASCII graphs, to better analyse functions
- **u** undo last seek

- To perform dynamic analysis:
\$ r2 -d mybin.exe
- Vpp to get to debugger visual mode
- Shortcuts:
 - F2 toggle breakpoint
 - F4 run to cursor
 - F7 single step
 - F8 step over
 - F9 continue

- **Giants**

Try to pass the CD check, and get to the main menu

You'll need to patch the **Giants.exe** binary

- **cARMm-cke**

Make it print 'Key valid'

Crackme in ARMv7, sheet included

- **IOLI-crackme**

Easy challenges for those who begin

Some links

Website

<http://rada.re/>

Blog

<http://radare.today>

Book

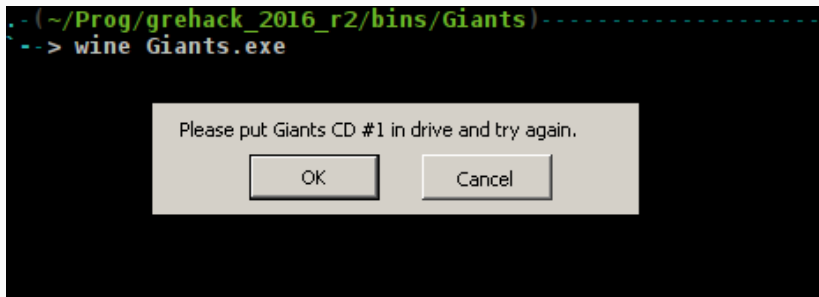
<http://radare.gitbooks.io/radare2book/content>

Cheat sheet

<https://github.com/pwntester/cheatsheets/blob/master/radare2.md>

Example: patching Giants

The error



Finding the string

```
..(~/Prog/grehack_2016_r2/bins/Giants)----- (master: 1f 15+ 15-)-(skia@Kiffu)-  
--> r2 Giants.exe  
-- THE ONLY WINNING MOVE IS NOT TO PLAY.  
[0x00547358]> iz~CD[0,1,14-42]  
vaddr 0x0055a948 string !"#%&'()*+ -./0123456789: < >?@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_  
vaddr 0x00564f54 string Please put Giants CD #1 In Drive and try again  
vaddr 0x00564f84 string ErrNeedCD1  
vaddr 0x00564f90 string Missing Music Files - Bad Giants CD or Install?  
vaddr 0x00564fd0 string Missing Stream Files - Bad Giants CD or Install?  
vaddr 0x00581558 string CDFS  
vaddr 0x00581568 string %s\..\GIANTSCD.1  
vaddr 0x005815ec string 0123456789ABCDEF  
vaddr 0x0065cb38 string AAAAAAAACD!\t  
vaddr 0x0065cbd8 string AAAAAAAACD!\t  
vaddr 0x0065f7e4 string #CD#.FC.  
[0x00547358]> □
```

Where is that string used?

```
[0x00547358]> iz~Giants CD #1[0,1,14-42]
vaddr 0x00564f54 string Please put Giants CD #1 In Drive and try again
[0x00547358]> /c 564f54
0x004f419b # 5: add eax, 0x564f54b8
0x004f419c # 5: mov eax, str.Please_put_Giants_CD__1_In_Drive_and_try_again
[0x00547358]> □
```

A bit of assembly

```
[0x00547358]> pd 25 @ 0x004f419c - 0x20
0x004f417c      5f      pop edi
0x004f417d      5e      pop esi
0x004f417e      c3      ret
0x004f417f      e8fc6c0400 call 0x53ae80
0x004f4184      85c0     test eax, eax
0x004f4186      7527     jne 0x4f41af
0x004f4188      68844f5600 push str.ErrNeedCD1 ; str.ErrNeedCD1 ; "ErrNeedCD1" @ 0x564f84
0x004f418d      e85e960100 call 0x50d7f0
0x004f4192      83c404   add esp, 4
0x004f4195      3d844f5600 cmp eax, str.ErrNeedCD1 ; "ErrNeedCD1" @ 0x564f84
;-- hit0_0:
;-- hit0_1:
0x004f419a      7505     jne 0x4f41a1
0x004f419c      b8544f5600 mov eax, str.Please_put_Giants_CD_1_In_Drive_and_try_again ; "Please put Giants CD #1 In Drive and try again" @ 0x564f54
0x004f41a1      50      push eax
0x004f41a2      e859f50200 call 0x523700
0x004f41a7      83c404   add esp, 4
0x004f41aa      33c0     xor eax, eax
0x004f41ac      5f      pop edi
0x004f41ad      5e      pop esi
0x004f41ae      c3      ret
0x004f41af      5f      pop edi
0x004f41b0      b801000000 mov eax, 1
0x004f41b5      5e      pop esi
0x004f41b6      c3      ret
0x004f41b7      90      nop
0x004f41b8      90      nop
[0x00547358]>
```

Patch with

wao jz @ 0x004f4186

radare2, for fame, glory and shells

- Julien (jvoisin) Voisin
- dustri.org
- websec.fr
- I know some english¹

¹As demonstrated this morning.

Disclaimer

- The challenges are public
- This part of the workshop will be an interactive walkthrough
- Ask questions!

openCTF 2016 - apprentice_www

- OpenCTF 2016²
- During DefCon24, it was pretty fun.
- This is a trivial challenge

²<http://openctf.com/>

pdf @ main

print the **d**isassembly of a whole **f**unction **@** at the location of the **main** symbol.

pdf @ main

```

[0x08048450]> pdf @ main
;-- main:
(fcn) sym.main 90
    sym.main ();
    ; var int local_4h @ esp+0x4
    ; var int local_1ch @ esp+0x1c
0x0804863e    55                push ebp
0x0804863f    89e5             mov ebp, esp
0x08048641    83e4f0          and esp, 0xffffffff
0x08048644    83ec20          sub esp, 0x20
0x08048647    c744241c0100.  mov dword [esp + local_1ch], 1
0x0804864f    a140a00408     mov eax, dword [obj.stdin]
0x08048654    c74424040000.  mov dword [esp + local_4h], 0
0x0804865c    890424          mov dword [esp], eax
0x0804865f    e87cfdffff     call sym.imp.setbuf ; void setbuf(FILE *stream, char*);
0x08048664    a160a00408     mov eax, dword [obj.stdout]
0x08048669    c74424040000.  mov dword [esp + local_4h], 0
0x08048671    890424          mov dword [esp], eax
0x08048674    e867fdffff     call sym.imp.setbuf ; void setbuf(FILE *stream, char*);
0x08048679    c704241e0000.  mov dword [esp], 0x1e
0x08048680    e87bfdffff     call sym.imp.alarm
0x08048685    c704243e8604.  mov dword [esp], sym.main
0x0804868c    e8bcfeffff     call sym.setup
0x08048691    e801ffff       call sym.butterflySwag ; long double erfl(long double x);
0x08048696    c9              leave
0x08048697    c3              ret
[0x08048450]> █

```

pdf @ sym.setup

```
[0x08048450]> pdf @ sym.setup
(fcn) sym.setup 74
    sym.setup (int arg_8h);
        ; var int local_10h @ ebp-0x10
        ; var int local_ch @ ebp-0xc
        ; arg int arg_8h @ ebp+0x8
        ; var int local_4h @ esp+0x4
        ; var int local_8h @ esp+0x8
        ; CALL XREF from 0x0804868c (sym.main)
0x0804854d      55                push ebp
0x0804854e      89e5              mov ebp, esp
0x08048550      83ec28            sub esp, 0x28
0x08048553      c745f0000000     mov dword [ebp - local_10h], 0
0x0804855a      eb33              jmp 0x804858f
0x0804855c      8b4508            mov eax, dword [ebp + arg_8h] ; [0x8:4]=0
0x0804855f      2508000408        and eax, 0x8048000
0x08048564      8945f4            mov dword [ebp - local_ch], eax
0x08048567      8b45f0            mov eax, dword [ebp - local_10h]
0x0804856a      c1e00c            shl eax, 0xc
0x0804856d      0145f4            add dword [ebp - local_ch], eax
0x08048570      8b45f4            mov eax, dword [ebp - local_ch]
0x08048573      c74424080700     mov dword [esp + local_8h], 7
0x0804857b      c74424040010     mov dword [esp + local_4h], 0x1000 ; [0x1000:4]=0x8049f14 obj._DYNAMIC
0x08048583      890424            mov dword [esp], eax
0x08048586      e865feffff        call sym.imp.mprotect
0x0804858b      8345f001          add dword [ebp - local_10h], 1
        ; JMP XREF from 0x0804855a (sym.setup)
0x0804858f      837df002          cmp dword [ebp - local_10h], 2 ; [0x2:4]=0x101464c
0x08048593      7ec7              jle 0x804855c
0x08048595      c9                leave
0x08048596      c3                ret
[0x08048450]> █
```

pdf @ sym.butterflySwag

```

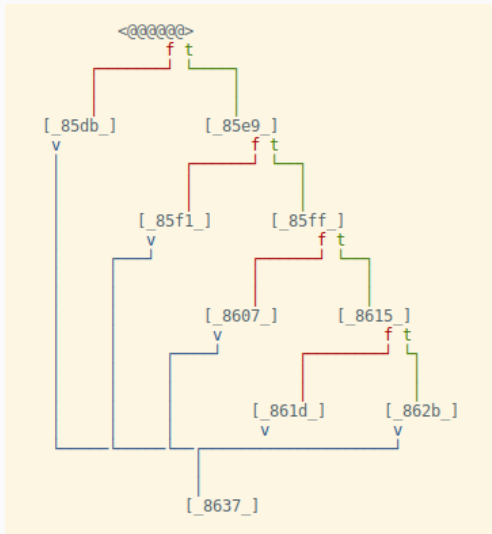
[0x00040597]> pdf
- (fcn) sym.butterflySwag 167
  sym.butterflySwag ();
    ; var int local_10h @ ebp-0x10
    ; var int local_ch @ ebp-0xc
    ; var int local_4h @ esp+0x4
    0x00040597 55          push ebp
    0x00040598 89e5        mov ebp, esp
    0x0004059a 83ec28      sub esp, 0x28
    0x0004059d 8d45f0      lea eax, [ebp - local_10h]
    0x000405a0 894d404     mov dword [esp + local_4h], eax
    0x000405a4 c70424308704. mov dword [esp], 0x00040730
    0x000405ab e890feffff  call sym.imp._isoc99_scanf; int scanf(const char *format);
    0x000405b0 8d45f4      lea eax, [ebp - local_ch]
    0x000405b3 894d2404     mov dword [esp + local_4h], eax
    0x000405b7 c70424308704. mov dword [esp], 0x00040733
    0x000405be e87dfeffff  call sym.imp._isoc99_scanf; int scanf(const char *format);
    0x000405c3 8b45f4      mov eax, dword [ebp - local_ch]
    0x000405c6 0fb6c0      movzx eax, al
    0x000405c9 8945f4      mov dword [ebp - local_ch], eax
    0x000405cc 8b45f0      mov eax, dword [ebp - local_10h]
    0x000405cf 8b55f4      mov edx, dword [ebp - local_ch]
    0x000405d2 8810        mov byte [eax], dl
    0x000405d4 8b45f4      mov eax, dword [ebp - local_ch]
    0x000405d7 85c0        test eax, eax
    0x000405d9 75be        jne 0x000405e9
    0x000405db c70424308704. call sym.imp.puts, str.That_which_does_not_kill_us_makes_us_stronger.
    0x000405e2 e829feffff  call sym.imp.puts ; int puts(const char *s);
    0x000405e7 eb1e        jmp 0x00040637
    0x000405e9 8b45f4      mov eax, dword [ebp - local_ch]
    0x000405ec 83f001      cmp eax, 1
    0x000405ef 75be        jne 0x000405ff
    0x000405f1 c70424608704. mov dword [esp], str.All_truly_great_thoughts_are_conceived_by_walking.
    0x000405f8 e813feffff  call sym.imp.puts ; int puts(const char *s);
    0x000405fd eb38        jmp 0x00040637
    0x000405ff 8b45f4      mov eax, dword [ebp - local_ch]
    0x00040602 83f004      cmp eax, 4
    0x00040605 770e        ja 0x00040615
    0x00040607 c704249c8704. mov dword [esp], str.Without_music_life_would_be_a_mistake.
    0x0004060e e8fdfffeffff call sym.imp.puts ; int puts(const char *s);
    0x00040613 eb22        jmp 0x00040637
    0x00040615 8b45f4      mov eax, dword [ebp - local_ch]
    0x00040618 83f009      cmp eax, 9
    0x0004061b 770e        ja 0x0004062b
    0x0004061d c70424c48704. mov dword [esp], str.He_who_has_a_why_to_live_can_bear_almost_any_how.
    0x00040624 e8e7fdfffeffff call sym.imp.puts ; int puts(const char *s);
    0x00040629 eb0c        jmp 0x00040637
    0x0004062b c70424f88704. mov dword [esp], str.When_you_look_into_an_abyss_the_abyss_also_looks_into_you.
    0x00040632 e800fdfffeffff call sym.imp.puts ; int puts(const char *s);
    0x00040637 8b45f4      mov eax, 0
    0x0004063c c9          leave
    0x0004063d c3          ret
[0x00040597]>

```

pdf @ sym.butterflyswag | grep -e call -e '<' -e '>'

```
[0x08048597]> pdf @ sym.butterflySwag | grep -e call -e '<' -e '>'
0x080485ab e890feffff call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485be e87dfeffff call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485d9 750e jne 0x80485e9
0x080485e2 e829feffff call sym.imp.puts ; int puts(const char *s);
0x080485e7 eb4e jmp 0x8048637
0x080485e9 8b45f4 mov eax, dword [ebp - local_ch]
0x080485ef 750e jne 0x80485ff
0x080485f8 e813feffff call sym.imp.puts ; int puts(const char *s);
0x080485fd eb38 jmp 0x8048637
0x080485ff 8b45f4 mov eax, dword [ebp - local_ch]
0x08048605 770e ja 0x8048615
0x0804860e e8fdfeffff call sym.imp.puts ; int puts(const char *s);
0x08048613 eb22 jmp 0x8048637
0x08048615 8b45f4 mov eax, dword [ebp - local_ch]
0x0804861b 770e ja 0x804862b
0x08048624 e8e7feffff call sym.imp.puts ; int puts(const char *s);
0x08048629 eb0c jmp 0x8048637
0x0804862b c70424f88704 mov dword [esp], str.When_you_look_into_an_abyss_the_abyss_also_looks_into_you.
0x08048632 e8d9feffff call sym.imp.puts ; int puts(const char *s);
0x08048637 b800000000 mov eax, 0

[0x08048597]> █
```



- Visual mode
- View graph
- rotate p/Print modes

pd 20 @ sym.butterflySwag

```

[0x08048450]> pd 20 @ sym.butterflySwag
(fcn) sym.butterflySwag 167
    sym.butterflySwag ();
        ; var int local_10h @ ebp-0x10
        ; var int local_ch @ ebp-0xc
        ; var int local_4h @ esp+0x4
        ; CALL XREF from 0x08048691 (sym.main)
0x08048597      55          push ebp
0x08048598      89e5        mov ebp, esp
0x0804859a      83ec28      sub esp, 0x28          ; '('
0x0804859d      8d45f0      lea eax, [ebp - local_10h]
0x080485a0      89442404    mov dword [esp + local_4h], eax
0x080485a4      c70424308704. mov dword [esp], 0x8048730 ; [0x8048730:4]=0x25007525
0x080485ab      e890feffff  call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485b0      8d45f4      lea eax, [ebp - local_ch]
0x080485b3      89442404    mov dword [esp + local_4h], eax
0x080485b7      c70424338704. mov dword [esp], 0x8048733 ; [0x8048733:4]=0x6425
0x080485be      e87dfeffff  call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485c3      8b45f4      mov eax, dword [ebp - local_ch]
0x080485c6      0fb6c0      movzx eax, al
0x080485c9      8945f4      mov dword [ebp - local_ch], eax
0x080485cc      8b45f0      mov eax, dword [ebp - local_10h]
0x080485cf      8b55f4      mov edx, dword [ebp - local_ch]
0x080485d2      8b10        mov byte [eax], dl
0x080485d4      8b45f4      mov eax, dword [ebp - local_ch]
0x080485d7      85c0        test eax, eax
0x080485d9      750e        jne 0x80485e9
[0x08048450]>

```

So what?

- The `.text` and `.bss` segments are `RWX`
- We can write `one` byte at an arbitrary location.

How do we pop a shell now?

pd 20 @ sym.butterflySwag

```

[0x08048450]> pd 20 @ sym.butterflySwag
(fcn) sym.butterflySwag 167
    sym.butterflySwag ();
        ; var int local_10h @ ebp-0x10
        ; var int local_ch @ ebp-0xc
        ; var int local_4h @ esp+0x4
        ; CALL XREF from 0x08048691 (sym.main)
0x08048597      55          push ebp
0x08048598      89e5        mov ebp, esp
0x0804859a      83ec28      sub esp, 0x28          ; '('
0x0804859d      8d45f0      lea eax, [ebp - local_10h]
0x080485a0      89442404    mov dword [esp + local_4h], eax
0x080485a4      c70424308704. mov dword [esp], 0x8048730 ; [0x8048730:4]=0x25007525
0x080485ab      e890feffff  call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485b0      8d45f4      lea eax, [ebp - local_ch]
0x080485b3      89442404    mov dword [esp + local_4h], eax
0x080485b7      c70424338704. mov dword [esp], 0x8048733 ; [0x8048733:4]=0x6425
0x080485be      e87dfeffff  call sym.imp.__isoc99_scanf; int scanf(const char *format);
0x080485c3      8b45f4      mov eax, dword [ebp - local_ch]
0x080485c6      0fb6c0      movzx eax, al
0x080485c9      8945f4      mov dword [ebp - local_ch], eax
0x080485cc      8b45f0      mov eax, dword [ebp - local_10h]
0x080485cf      8b55f4      mov edx, dword [ebp - local_ch]
0x080485d2      8b10        mov byte [eax], dl
0x080485d4      8b45f4      mov eax, dword [ebp - local_ch]
0x080485d7      85c0        test eax, eax
    < 0x080485d9      750e        jne 0x80485e9
[0x08048450]> █

```

The Plan

- Patch the `jne` at `0x080485da`
- Use the infinite loop to write our shellcode
- Jump on our shellcode

Patching the jump

- `e io.cache = 1`
- `wx 74c2 @ 0x080485d9`
- `pd 20 @ sym.butterflySwag`

Patching the jump

```
[0x08048597]> e io.cache = true
[0x08048597]> wx 74c2 @ 0x080485d9
[0x08048597]> pd 20 @ sym.butterflySwag
(fcn) sym.butterflySwag 167
sym.butterflySwag ();
    ; var int local_10h @ ebp-0x10
    ; var int local_ch @ ebp-0xc
    ; var int local_4h @ esp+0x4
    ; CALL XREF from 0x08048691 (sym.main)
0x08048597      55          push ebp
0x08048598      89e5        mov ebp, esp
0x0804859a      83ec28      sub esp, 0x28 ; '('
0x0804859d      8d45f0      lea eax, [ebp - local_10h]
0x080485a0      89442404    mov dword [esp + local_4h], eax
0x080485a4      c70424308704. mov dword [esp], 0x8048730 ; [0x8048730:4]=0x25007525
0x080485ab      e890feffff  call sym.imp._isoc99_scanf; int scanf(const char *format);
0x080485b0      8d45f4      lea eax, [ebp - local_ch]
0x080485b3      89442404    mov dword [esp + local_4h], eax
0x080485b7      c70424338704. mov dword [esp], 0x8048733 ; [0x8048733:4]=0x6425
0x080485be      e87dfeffff  call sym.imp._isoc99_scanf; int scanf(const char *format);
0x080485c3      8b45f4      mov eax, dword [ebp - local_ch]
0x080485c6      0fb6c0      movzx eax, al
0x080485c9      8945f4      mov dword [ebp - local_ch], eax
0x080485cc      8b45f0      mov eax, dword [ebp - local_10h]
0x080485cf      8b55f4      mov edx, dword [ebp - local_ch]
0x080485d2      8810      mov byte [eax], dl
0x080485d4      8b45f4      mov eax, dword [ebp - local_ch]
0x080485d7      85c0      test eax, eax
0x080485d9      74c2      je 0x804859d
[0x08048597]> □
```

- `ragg2 -b 32 -i exec -z`
- `ragg2 -b 32 -i exec -z | rasm2 -d -b 32 -`

Fill the `exploit.py` template!

DefCamp 2015 - exp200

- DefCamp 2015³
- Awful CTF, but Romania was fun
- Simple challenge
- No ASLR⁴

³<http://DefCamp.ro>

⁴`sysctl -w kernel.randomize_va_space=0`

Surprise popquizz

Are you familiar with the concepts of:

- Stack

Surprise popquizz

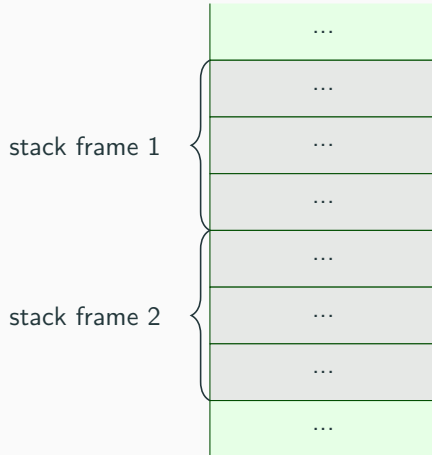
Are you familiar with the concepts of:

- Stack
- ROP

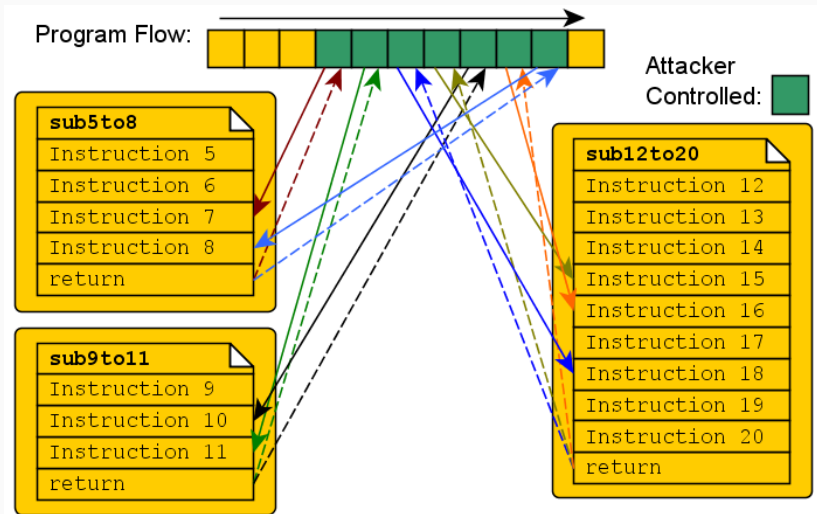
Are you familiar with the concepts of:

- Stack
- ROP
- ROP on x64

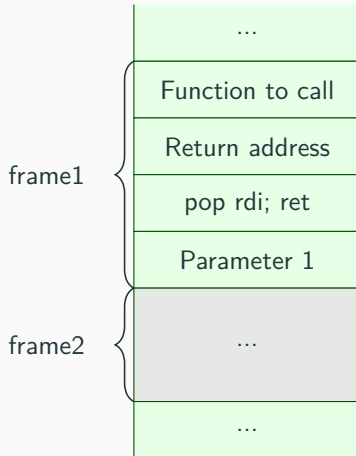
What is a stack



ROP



Rop chain



```

[0x004005bd]> pdf @ main
;-- main:
(fcn) sym.main 117
sym.main ();
; var int local_8h @ rbp-0x8
0x004005bd 55          push rbp
0x004005be 4889e5      mov rbp, rsp
0x004005c1 4883ec10    sub rsp, 0x10
0x004005c5 41b900000000 mov r9d, 0
0x004005cb 41b8ffffff   mov r8d, 0xffffffff
0x004005d1 b922000000   mov ecx, 0x22
0x004005d6 ba07000000   mov edx, 7
0x004005db be00020000   mov esi, 0x200
0x004005e0 bf00000010   mov edi, 0x10000000
0x004005e5 e896feffff   call sym.imp.mmap
0x004005ea 488945f8     mov qword [rbp - local_8h], rax
0x004005ee 488d45f8     lea rax, [rbp - local_8h]
0x004005f2 ba00020000   mov edx, 0x200
0x004005f7 4889c6      mov rsi, rax
0x004005fa bf00000000   mov edi, 0
0x004005ff b800000000   mov eax, 0
0x00400604 e887feffff   call sym.imp.read ; ssize_t
0x00400609 ba01000000   mov edx, 1
0x0040060e be00020000   mov esi, 0x200
0x00400613 bf00000010   mov edi, 0x10000000
0x00400618 e8a3feffff   call sym.imp.mprotect
0x0040061d 488b45f8     mov rax, qword [rbp - local_8h]
0x00400621 4889c2      mov rdx, rax
0x00400624 b800000000   mov eax, 0
0x00400629 ffd2        call rdx
0x0040062b b800000000   mov eax, 0
0x00400630 c9          leave
0x00400631 c3          ret
[0x004005bd]>

```

1. `mmap` a `0x200` bytes area
2. read out input in it
3. `mprotect` is a `read-only`
4. `call` the aforementioned area

What do we control?

```
jvoisin@kaa 18:22 ~/ctf/defcamp2015/e200 r2 ./exp200 -d
Process with PID 9408 started...
= attach 9408 9408
bin.baddr 0x00400000
Assuming filepath /home/jvoisin/ctf/defcamp2015/e200/exp200
asm.bits 64
-- I did it for the pwnz.
[0x7ffff7dd8ca0]> dc
AAAAAAAAABBBBBBBB
child stopped with signal 11
[+] SIGNAL 11 errno=0 addr=0x00000000 code=128 ret=0
= attach 9408 1
[0x00400629]> pd 4 @ rip
;-- rip:
0x00400629 ffd2 call rdx
0x0040062b b800000000 mov eax, 0
0x00400630 c9 leave
0x00400631 c3 ret
[0x00400629]> dr edx
0x41414141
[0x00400629]> □
```

What do we control?

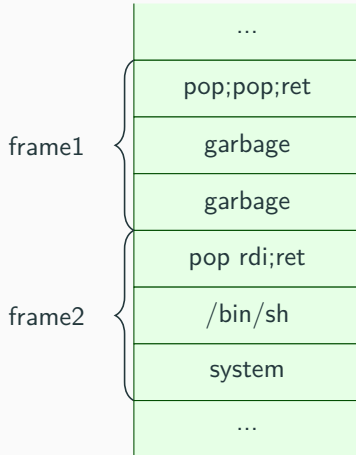
```
[0x00400629]> pxQ 64 @ rsp
0x7fffffff020 0x00007fffffff0110 r13
0x7fffffff028 0x4141414141414141 rdx
0x7fffffff030 0x4242424242424242
0x7fffffff038 0x4343434343434343
0x7fffffff040 0x0000000000000000a section.+10
0x7fffffff048 0x00007fffffff0118 r13+8
0x7fffffff050 0x0000000100000000
0x7fffffff058 0x00000000004005bd main
[0x00400629]> □
```

How can we ROP our way to a shell?

Plan of action

1. Pop `r13` from the stack
2. `call` with push its return address on the stack: pop it too
3. Pop `/bin/sh` into `rdi`
4. Call `system`
5. Victory dance.

Lazy solution



Find ROP gadgets

```
[0x004004d0]> e rop.len = 3
[0x004004d0]> "/R pop"
0x00400513      7702  ja 0x400517
0x00400515      5d    pop rbp
0x00400516      c3    ret

0x00400521      5d    pop rbp
0x00400522      bf50106000 mov edi, 0x601050
0x00400527      ffe0  jmp rax

0x00400550      7502  jne 0x400554
0x00400552      5d    pop rbp
0x00400553      c3    ret

0x00400582      5d    pop rbp
0x00400583      c605c60a200001 mov byte [rip + 0x200ac6], 1
0x0040058a      f3c3  ret

0x004005ad      ffd0  call rax
0x004005af      5d    pop rbp
0x004005b0      e97bffffff jmp 0x400530

0x004006a0      415e  pop r14
0x004006a2      415f  pop r15
0x004006a4      c3    ret

0x004006a1      5e    pop rsi
0x004006a2      415f  pop r15
0x004006a4      c3    ret

0x004006a3      5f    pop rdi
0x004006a4      c3    ret

[0x004004d0]> █
```

We've got a `pop rdi;ret` and a `pop;pop;ret`.

Fill the `exploit.py` template!

Conclusion

Conclusion

- Using radare2 is like using vim in Dwarf Fortress

Conclusion

- Using radare2 is like using vim in Dwarf Fortress
- Please complain on `#radare2` on freenode

Conclusion

- Using radare2 is like using vim in Dwarf Fortress
- Please complain on `#radare2` on freenode
- Also remember that this software comes with no brain included. Please use your own.

Conclusion

- Using radare2 is like using vim in Dwarf Fortress
- Please complain on `#radare2` on freenode
- Also remember that this software comes with no brain included. Please use your own.

Conclusion

- Using radare2 is like using vim in Dwarf Fortress
- Please complain on `#radare2` on freenode
- Also remember that this software comes with no brain included. Please use your own.

Question?