Practical Reverse Engineering using Radare2

Sanoop Thomas

@s4n7h0

Instructor Briefing (2min)

- Sanoop Thomas
  - I write technical contents at devilslab.in
  - Former chapter moderator at Null Mumbai
  - Currently core team & chapter moderator at Null Singapore
  - Author of Halcyon IDE halcyon-ide.org
  - Presented at Nullcon, OWASP India, BlackHat Arsenal, HITBGSEC, ROOTCON
  - Learned from community, so time to give it back.
  - I tweet at `@s4n7h0`

Participants Briefing (10min)

- Name and background
- How long are you attending with nullsg or other null chapters
- What do you do in Information Security
- Your experience specifically in programming, assembly and reverse engineering
- Your expectation from this workshop

Agenda

- Introduction to Radare2
- Reverse Engineering for Beginners – A Crash Course
- Binary Analysis
- Binary Diffing
- Binary Patching
- Shellcode Analysis
- Lab exercises at different levels.

```
.
├── bin
│   ├── lab0
│   ├── lab1
│   ├── lab2
│   ├── lab3
│   ├── lab4
│   └── lab5
└── src
    ├── lab0.c
    ├── lab1.c
    ├── lab2.c
    ├── lab3.c
    ├── lab4.c
    └── lab5.c
```

- This workshop materials will be updated here
  - https://github.com/s4n7h0/Practical-Reverse-Engineering-using-Radare2

#protips

- These thoughts are perfectly fine
  - “I have no idea what to do next.”
  - “Oops! I made wrong assumptions.”
  - “aaggrrhh, it’s taking too much time.”

radare
@radareorg

Following

Love to hear people around the world pronouncing "radare2" :D

RETWEETS 10 LIKES 12

9:49 PM - 15 Dec 2015

10 12

Reply to @radareorg

K. Erik Wolfe @EthernetNinja · 15 Dec 2015
@radareorg is it "raid-air two" or "rad-air two" or "rad-ahr-re two" or....?

radare @radareorg · 16 Dec 2015
@ethernetninja i'm catalan (@trufae) and i pronounce it like it sounds for me, but i don't want any specific pronuncation to be official

Setup

- Use Git
  - `git clone https://github.com/radare/radare2.git`
  - `sys/install.sh`
- New stable releases are published at http://bin.rada.re/

Radare2

- Started as single person project in early 2006.
- **Ra**w **Da**ta **Re**covery – started as a hexeditor with disc search and data recovery features.
- Today it supports more features – disassembler, debugging, graphing, scripting etc.,
- Radare2 is focused on portability and thus it supports a lot of architectures
- Applies *nix theory – “everything is a file”. Meaning, possibly everything can be reversed.

Recognized file formats

- COFF and derivatives, including Win32/64/generic PE
- ELF and derivatives
- Mach-O (Mach) and derivatives
- Game Boy and Game Boy Advance cartridges
- MZ (MS-DOS)
- Java class
- dyld cache dump[19]
- Dex (Dalvik EXecutable)
- Xbox xbe format[20]
- Plan9 binaries
- Winrar virtual machine[21]
- File system like the ext family, ReiserFS, HFS+, NTFS, FAT, ...
- DWARF and PDB file formats for storing additional debug information
- Raw binary

https://en.wikipedia.org/wiki/Radare2

Instruction sets

- Intel x86 family
- ARM architecture
- Atmel AVR series
- Brainfuck
- Motorola 68k and H8
- Ricoh 5A22
- MOS 6502
- Smartcard PSOS Virtual Machine
- Java virtual machine
- MIPS: mipsb/mipsl/mipsr/mipsrl/r5900b/r5900l
- PowerPC
- SPARC Family
- TMS320Cxxx series
- Argonaut RISC Core
- Intel 51 series: 8051/80251b/80251s/80930b/80930s
- Zilog Z80
- CR16
- Cambridge Silicon Radio (CSR)
- AndroidVM Dalvik
- DCPU-16
- EFI bytecode
- Gameboy (z80-like)
- Java Bytecode
- Malebolge
- MSIL/CIL
- Nios II
- SuperH
- Spc700
- Systemz
- TMS320
- V850
- Whitespace
- XCore

https://en.wikipedia.org/wiki/Radare2

Utilities

Rax2	Used for calculating and conversion
Rabin2	Used for extracting and analysing binary information.
Rasam2	Used for command line assembler and disassembler
Rahash2	An implementation of a block-based hash tool.
Radiff2	A binary diffing utility.
Rafind2	Used to find byte patterns in files.
Ragg2	Used for generating or compiling shellcodes.
Rarun2	Used for running programs within different environments, different settings etc.
Radare2	The core hex editor, debugger and more
R2	Same as radare2

Rax2

- rax2 10
  - 0xa
- rax2 0xa
  - 10
- rax2 -s 4141
  - AA
- rax2 -S AA
  - 4141
- rax2 -b 01000001
  - A
- rax2 -B A
  - 01000001
- rax2 0x33+3
  - 54
- rax2 -k 0x33+3
  - 0x36
- rax2 -n 0x1234
  - 34120000
- rax2 -N 0x1234
  - \x34\x12\x00\x00
- rax2 -r 0x1234
  - **Try this**
- rax2 -E something
  - **Try this**
- Rax2 can be used inside radare2 console
  - ? 0x80+3
- See help for more options
  - rax2 -h

Rabin2

-A	Architecture
-e	Entry Point
-g	Shows all possible information
-i	Import List
-I	Binary Information
-j	Print result in JSON format
-K	Calculate checksum on sections
-l	Linked libraries
-m	Show source line at address
-M	Shows source line of main
-n	show section, symbol or import named str
-q	Quiet mode output
-s	Shows symbols
-S	Shows sections
-V	Shows binary version
-z	Displays strings
-Z	Shows size of binary

- **Example:**
  - `rabin2 -Ieqqzzj /bin/true`

Binary Analysis

- Use **lab0** to experiment with the following analysis
  - Get the binary information
  - Get the architecture and entry point together
  - Get the strings in normal and quiet mode

Rasm2

- rasm2 -a x86 -b 32 'mov eax, 0xA' -C
  - "\xb8\x0a\x00\x00\x00“
- rasm2 -d 90
  - nop
- rasm2 nop
  - 90
- rasm2 -f lessons/hello.asm
  - b83c0000000f05
- rasm2 -d b83c0000000f05
  - mov eax, 0x3c
  - syscall

Rahash2

- rahash2 -a all binary_file
  - Computes all hashes for whole binary
- rahash2 -a sha1 binary_file
  - Computes SHA1 for binary
- rahash2 -a entropy binary_file
  - Computes entropy for binary
- rahash2 -B -b 512 -a md5 binary_file
  - Computes MD5 for all 512 blocks in the binary
- rahash2 -B -b 512 -a entropy binary_file
  - Computes entropy for all 512 blocks in the binary

```
s4n7h0@mate:~/lessons/lab1$ rahash2 -B -b 512 -a sha1,entropy lab1
0x00000000-0x000001ff sha1: cd140becb5c9b18cf9c53365548c9786560b8b17
0x00000200-0x000003ff sha1: 742c71fea0b9b328f3610bf293db77c2b45ce88f
0x00000400-0x000005ff sha1: db1e194b009ce51952263795d129ef2380477e46
0x00000600-0x000007ff sha1: f4acea7a145c8ff8d27e1ea3361466855056cd6e
0x00000800-0x000009ff sha1: 5c3eb80066420002bc3dcc7ca4ab6efad7ed4ae5
0x00000a00-0x00000bff sha1: 5c3eb80066420002bc3dcc7ca4ab6efad7ed4ae5
0x00000c00-0x00000dff sha1: 5c3eb80066420002bc3dcc7ca4ab6efad7ed4ae5
0x00000e00-0x00000fff sha1: bcc64f0532fcfd26ec913c791b02ee0028df8c1a
0x00001000-0x000011ff sha1: 51549ed4b1350cc210f871b972474287943c805c
0x00001200-0x000013ff sha1: 39d5c82a9ae1964d34ee986be6eee704bc777db6
0x00001400-0x000015ff sha1: b8f5d449aaac3164187ff34cdcf18465689c5e7b
0x00001600-0x000017ff sha1: 4238cd39bae237c894a2b8bbcd1917ad74ee718e
0x00001800-0x000019ff sha1: 1c89f58e2d29d86db9e85cf6c9ec5fe32b891fa0
0x00001a00-0x00001bff sha1: 9a66ed64248de9553001a4bf31cb08c99d7279e5
0x00001c00-0x00001ca8 sha1: aab83cbcc86b3f39ab22e21dc0c4b93d7739d58a
0x00000000-0x000001ff   3.435249:   42% [##################----------------------------]
0x00000200-0x000003ff   4.545120:   56% [#########################---------------------]
0x00000400-0x000005ff   5.854245:   73% [################################-------------]
0x00000600-0x000007ff   4.528956:   56% [#########################---------------------]
0x00000800-0x000009ff   0.000000:    0% [----------------------------------------------]
0x00000a00-0x00000bff   0.000000:    0% [----------------------------------------------]
0x00000c00-0x00000dff   0.000000:    0% [----------------------------------------------]
0x00000e00-0x00000fff   1.476932:   18% [########------------------------------------]
0x00001000-0x000011ff   4.391732:   54% [########################---------------------]
0x00001200-0x000013ff   2.472092:   30% [#############------------------------------]
0x00001400-0x000015ff   2.030236:   25% [###########----------------------------------]
0x00001600-0x000017ff   2.498944:   31% [#############------------------------------]
0x00001800-0x000019ff   3.086725:   38% [##################----------------------------]
0x00001a00-0x00001bff   5.017376:   62% [###########################------------------]
0x00001c00-0x00001ca8   4.431106:   55% [########################---------------------]
s4n7h0@mate:~/lessons/lab1$
```

Classic “crackme”

- Analyse **lab1** binary
- Identify exploit protection flags
- Calculate entropy for each 512 bytes block
- Find the password

Radiff2

- To compute the distance and similarity
  - radiff2 -s /bin/true /bin/false
- To count the difference
  - radiff2 -c /bin/true /bin/false
- To analyze and check matching functions
  - radiff2 -AC /bin/true /bin/false
- Graph-diffing
  - radiff2 -g main /bin/true /bin/false

Image source: radare.gitbooks.io

Rafind2

- Show hexdump of string search hits
  - rafind2 -X -s findme example.bin
- Show hexdump of hex search hits
  - rafind2 -X -x ffff example.bin
- Show strings of string search hits
  - rafind2 -Z -s Congrats lesson1
- Show strings of hex search hits
  - rafind2 -Z -x 4e6f lesson1

```
s4n7h0@mate:~/lessons/crack$ rafind2 -X -x ffff lesson1
0x2ba
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x000002ba  ffff ff85 c074 05e8 3a00 0000 83c4 085b  .....t..:......[
0x000002ca  c300 0000 0000 ff35 04a0 0408 ff25 08a0  .......5.....%..
0x000002da  0408 0000 0000 ff25 0ca0 0408 6800 0000  .......%....h...
0x000002ea  00e9 e0ff ffff ff25 10a0 0408 6808 0000  .......%....h...
0x000002fa  00e9 d0ff ffff ff25 fc9f 0408 6690       .......%....f.
0x2ed
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x000002ed  ffff ffff 2510 a004 0868 0800 0000 e9d0  ....%....h......
0x000002fd  ffff ffff 25fc 9f04 0866 9000 0000 0000  ....%....f......
0x0000030d  0000 0031 ed5e 89e1 83e4 f050 5452 68c0  ...1.^.....PTRh.
0x0000031d  8404 0868 6084 0408 5156 680b 8404 08e8  ...h`...QVh.....
0x0000032d  bfff ffff f466 9066 9066 9066 9066       .....f.f.f.f.f
0x000002ef Sequencial hit ignored.
0x2fd
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x000002fd  ffff ffff 25fc 9f04 0866 9000 0000 0000  ....%....f......
0x0000030d  0000 0031 ed5e 89e1 83e4 f050 5452 68c0  ...1.^.....PTRh.
0x0000031d  8404 0868 6084 0408 5156 680b 8404 08e8  ...h`...QVh.....
0x0000032d  bfff ffff f466 9066 9066 9066 9066 9066  .....f.f.f.f.f.f
0x0000033d  9066 908b 1c24 c366 9066 9066 9066       .f...$.f.f.f.f
0x000002ff Sequencial hit ignored.
0x32e
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0000032e  ffff fff4 6690 6690 6690 6690 6690 6690  ....f.f.f.f.f.f.
0x0000033e  6690 8b1c 24c3 6690 6690 6690 6690 6690  f...$.f.f.f.f.f.
0x0000034e  6690 b81f a004 082d 1ca0 0408 83f8 0676  f......-.......v
0x0000035e  1ab8 0000 0000 85c0 7411 5589 e583 ec14  ........t.U.....
0x0000036e  681c a004 08ff d083 c410 c9f3 c390       h.............
0x3d1
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x000003d1  ffff ffc6 051c a004 0801 c9f3 c366 90b8  .............f..
0x000003e1  109f 0408 8b10 85d2 7505 eb93 8d76 00ba  ........u....v..
0x000003f1  0000 0000 85d2 74f2 5589 e583 ec14 50ff  ......t.U.....P.
```

Ragg2

```
$ cat hello.r
/* hello world in r_egg */
write@syscall(4);
exit@syscall(1);
main@global(128) {
    .var0 = "helloworld!";
    write(1,.var0, 12);
    exit(0);
}
$ ragg2 -O -F hello.r
$ ./hello
helloworld!
```

- Padding with 80 “A”s
  - ragg2 -p A80 -r
- Padding with a specific double word
  - ragg2 -p A80 -d 76:0x32323232 -r
- Generate a shell code
  - ragg2 -i exec
- Generate x86 32bit shell code in C format
  - ragg2 -a x86 -b 32 -i exec -z

Rarun2

- #!/usr/bin/rarun2

  program=./binary

  arg1=first_argument

  stdin=foobar.txt

  chdir=/tmp

  #chroot=.

- rarun2 program=binary arg1=first_argument

- For example,
  - nc -l 9999
  - rarun2 program=/bin/ls connect=localhost:9999

Radare2/R2

- Debugging mode (-d)
- Seek (s)
- Analyse (a)
- Search (/?)
- Print (p)
- Write (w)
- **Visualise mode (v)**
- Switch print mode (p/P)
- Navigate through symbols/objects (n/N)
- Seek to (o)
- Move (hjkl)
- Undo (u)

Correct Password

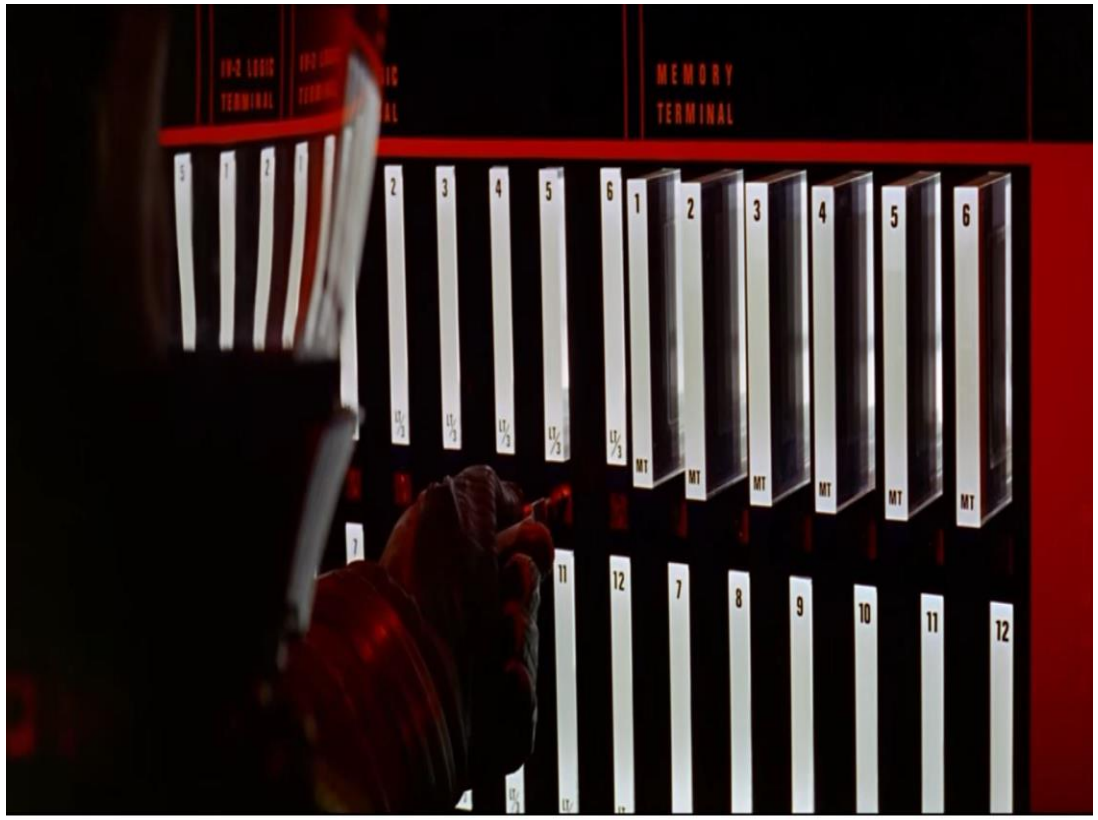
- Analyse **lab2** binary
- Find string patterns in the binary
- Analyse the program flow, graph (use r2)
- Crack the program using correct password

Always “good code”

- Analyse **lab3** binary
- Fetch binary protection flags in r2 console
- Analyse the program flow, graph (use r2)
- Patch the program as lab3_patch to print always good code
- Do binary diff for original and patch files

Deactivate HAL

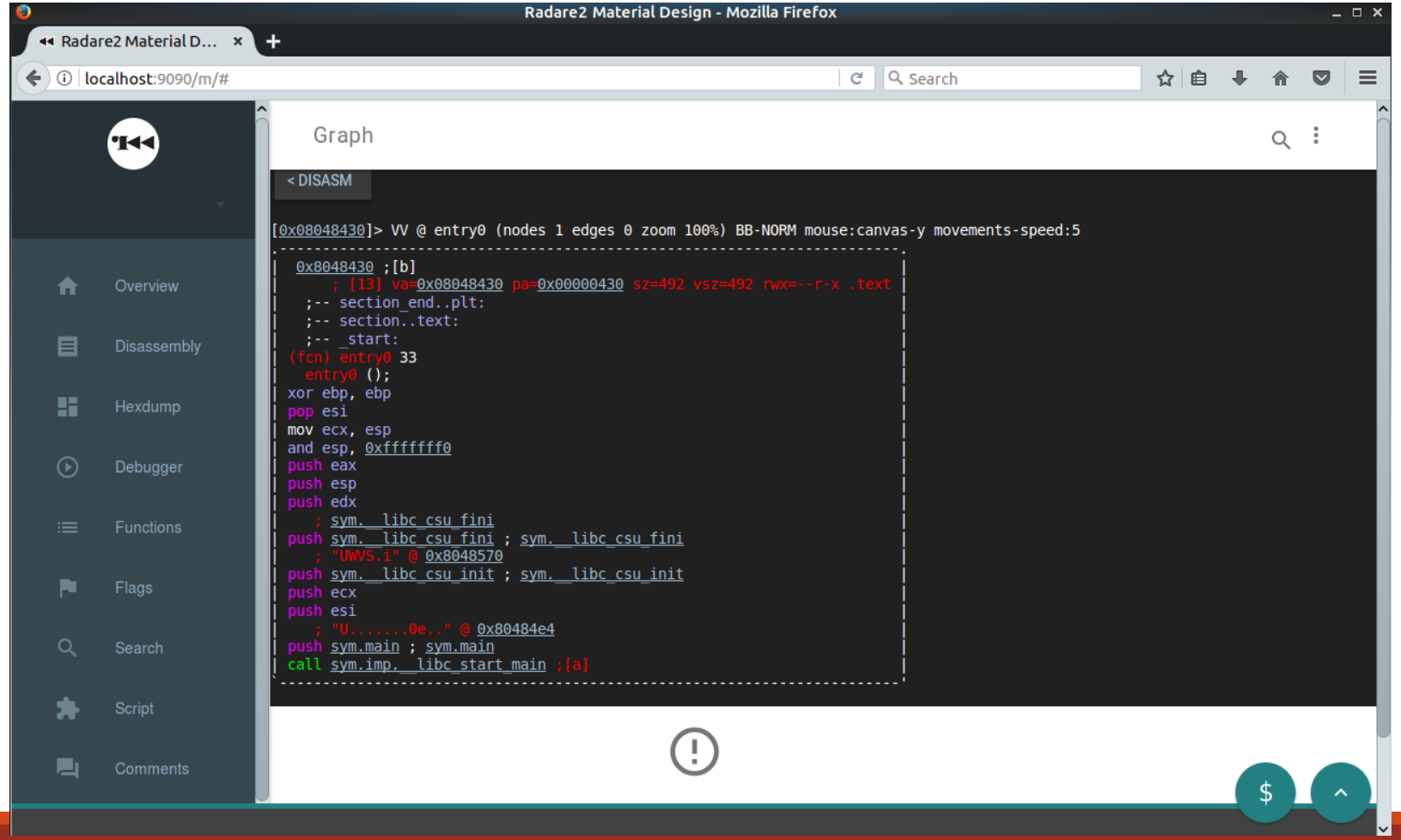
- Analyse **lab4** binary
- Control the program to deactivate HAL
- Create lab4_patch and also diff



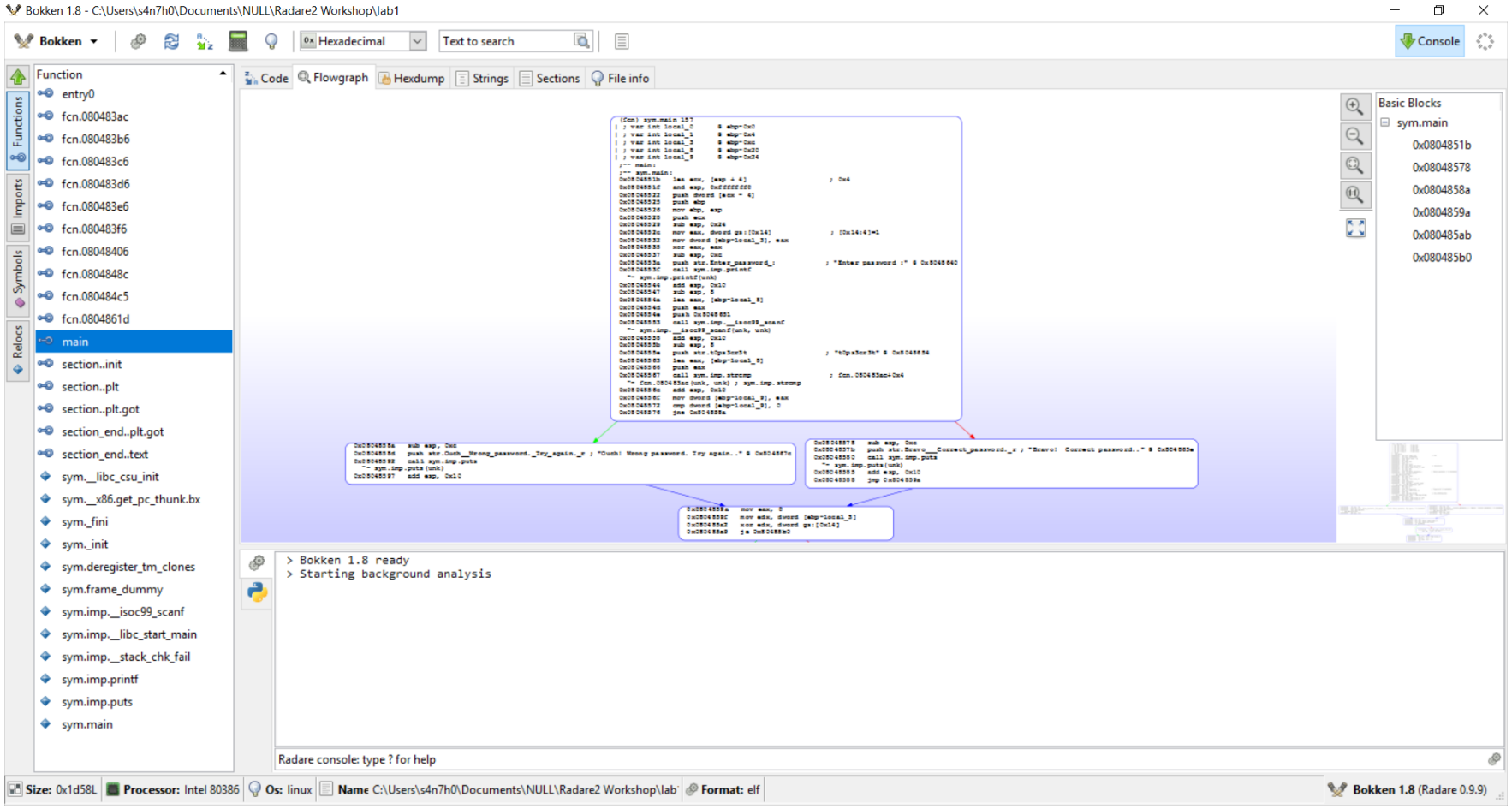
Shellcode Analysis

- Analyse the **lab5** binary
- Find the shellcode
- Identify what shellcode can do

Radare2 Web UI



Bokken


Bokken 1.8 - C:\Users\s4n7h0\Documents\NULL\Radare2 Workshop\lab1
Bokken
Hexadecimal
Text to search
Console
Function
entry0
fcn.080483ac
fcn.080483b6
fcn.080483c6
fcn.080483d6
fcn.080483e6
fcn.080483f6
fcn.08048406
fcn.0804848c
fcn.080484c5
fcn.0804861d
main
section..init
section..plt
section..plt.got
section_end..plt.got
section_end..text
sym.__libc_csu_init
sym.__x86.get_pc_thunk.bx
sym._fini
sym._init
sym.deregister_tm_clones
sym.frame_dummy
sym.imp.__isoc99_scanf
sym.imp.__libc_start_main
sym.imp.__stack_chk_fail
sym.imp.printf
sym.imp.puts
sym.main
Functions
Imports
Symbols
Relocs
Code
Flowgraph
Hexdump
Strings
Sections
File info
Basic Blocks
sym.main
0x0804851b
0x08048578
0x0804858a
0x0804859a
0x080485ab
0x080485b0
> Bokken 1.8 ready
> Starting background analysis
Radare console: type ? for help
Size: 0x1d58L
Processor: Intel 80386
Os: linux
Name C:\Users\s4n7h0\Documents\NULL\Radare2 Workshop\lab
Format: elf
Bokken 1.8 (Radare 0.9.9)

References

- http://rada.re/
- http://radare.today/
- https://radare.gitbooks.io/radare2book/content/introduction/intro.html
- https://github.com/pwntester/cheatsheets/blob/master/radare2.md