radare

Overview

In short: Advanced commandline hexadecimal editor

In long:

- Multiarchitecture/multiplatform and extensible hex editor with disassembler and lowlevel debugger
- Abstracted IO access
- Scripting capabilities
- Batch mode
- Code analysis with interfactive graphs
- Binary diffing with deltas
- Binary searchs with binary masks

Targets

Forensics (RAw DAta Recovery) search with binmask

Reversing (automatic and interactive code analysis)

Binary manipulation (audit binary protections)

Binary diffing (with delta support)

Pattern find and identification (aes keys, repeated bytes)

Debugging (lin/bsd/osx/w32 @ x86/ppc/arm/mips)

...

radare2

Need for a redesign

Set of 32 independent libraries

Aims to fully reimplement radare1 in modules

Bypass limitations of the monolitich design of radare1

Massive pluginization of functionalities

Some scriptting rules will change

radare2 structure

```
            +--------+
          .-| config |
         /  +--------+
  +------+  +------+ +------+ +------+
  | core |--| cons | | asm  | | diff |
  +------+  | line | | bin  | | sign |
    |    \  +------+ | anal | | hash |    ,_____.
  +----+  \          +---.--+ +--.---+   +._____.+
  | io |   +-------------'------/        |       |
  +----+   | cmd, search, print |<------>| flags |
    |      +------.-------------\        | meta  |
[ lib ]     +----'-----------+ +-`-----+ +._____.+
    |    .----| debug, bp, vm | | lang  |
    |    |    | reg, syscall  | | macro |
+------'--+ |  var, trace   | +-------+
| plugins | +---------------+     |
+-----.---+                       |
      `-------------------------'
```

radare2 status

Today we release r1-1.2.2 and r2-0.1

First release of radare2

Codename: Seaking

```
radare |####--------------| 20%
radiff |##----------------| 15%
rabin  |#########---------| 50%
rasm   |###############---| 80%
rax    |#########---------| 50%
```

http://www.radare.org/get/radare-1.2.2.tar.gz

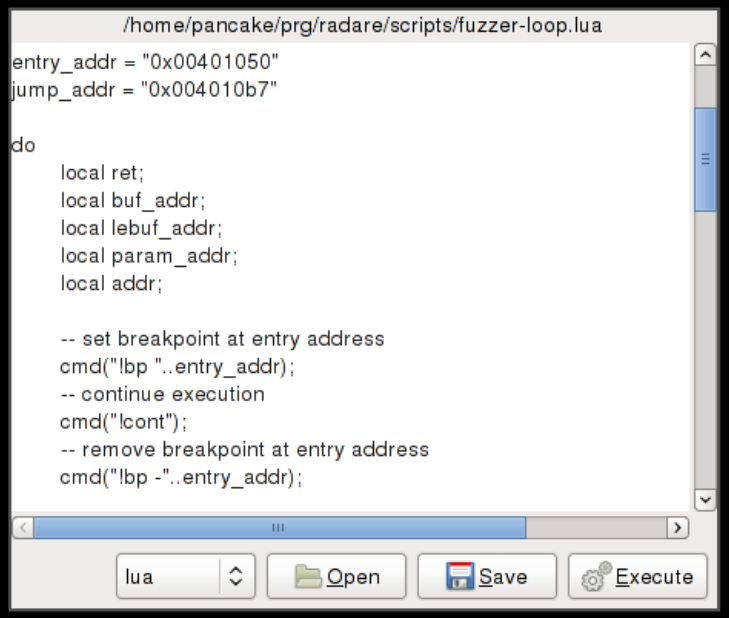
http://www.radare.org/get/radare2-0.1.tar.gz

radare scripting: basics

Native scripting:
- Cryptic (or mnemonic) +fun
- Macro-based with lot of sugar
- Actions are radare commands

Language bindings:
- Perl, Python, Ruby, LUA, …
- API based on radare commands

In radare2:
- Full access to the internal API from the script bindings
- Looks for an automated way for generating bindings



radare scripting: macros and sugar

Syntax sugar enables multiple actions in a single line.

```
pr 128 @ esp > stack ; dump 128 bytes of stack
wx 90@@hit ; write 0x90 at every flag matching 'hit'
3ds   ; run 3 times 'ds' command (alias for debug !step)
?[4:$$]~[0] ; get 4 bytes from $$ (curseek) and grep 1st col
!echo byte=`?[1:esi]~[0]` ; print first byte where esi points
pd 20 @ eip ; Disassemble 20 opcodes at eip
```

Radare commands can be grouped in macros to be used as functions with dynamic argument replacement.

```
(do-step num,!step $0,.!regs*,!dregs,pd 5 @eip,x 128 @esp)
.(do-step 10)  ; '.' command is for interpretting
```

radare scripting: iterators

Iterators are macros used with the '@@' suffix.

```
"(for-functions,()`C*~CF[3]#$@`)"
pdf @@ .(for-functions)
```

A null-macro means 'return from macro'.

"" quotes a command to avoid interpretting internal chars

Macro commands separated by commas

`` runs a subcommand and concatenates the result

C* lists all code metadata information

~CF[3] greps for lines matching CF and gets column 4

grep line number defined by next expression

$@ virtual variable inside macros that gives the number of times the macro has been called as iterator.

radare scripting: jpeg recovery

```
def recover_exif(addr):
  eval_set("search.to","$$")
  seek(-200)
  seek_search("Exif")
  byte = get_byte("$$")
  if byte == 0x45:
    seek(-6)
    write_to_files("dump", "2M")

def recover_iter(str):
  r.cmd("/ CASIO COMPUTER CO")
  hits[] = flag_list("hit0_")
  for hit in hits:
    recover_exit(hit[addr])
```

```
(recover-exif,
  e search.to=$$
  s -200
  s/ Exif
  ? [1:$$]-0x45
  ?!()
  s -6
  wT dump 2M)
; run the macro!
/ CASIO COMPUTER CO
.(recover-exif) @@ hit0_
```

radare scripting: code analysis

Running this macro while stepping in debugger adds comments to mark branches as likely/unlikely.

```
(step-post-anal
 ?z`ao@oeip~type = cond`
  ??()
 ?eip-`ao@oeip~jump =[2]`
  ??CC likely@oeip
  ??()
 CC unlikely@oeip)
e cmd.prompt=.(step-post-anal)
```

```
[0xB7F92FCB]> ao@oeip
pas = jz sub_0xb7f93028
index = 0
size = 2
stackop = unknown(0)
type = conditional-jump
bytes = 7440
offset = 0xb7f92fe6
ref = 0x00000000
jump = 0xb7f93028
fail = 0xb7f92fe8
```

?z (true if zero length string)
??() return from macro if previous conditional matches
CC likely @ oeip ; adds a comment ('likely') at oeip (old eip address)

radare debugger

One of the IO plugins enables radare to attach to processes and work with its memory like if it was a plain file.

Commands are sent via the system() io hook of the plugin

Support for ptrace (linux/bsd/osx), w32 and some mach(osx)

Remoting is done with socket connections:

```
radare listen://:9999
radare connect://172.26.3.22:9999/dbg:///bin/ls
```

Commands run in local, io and debug commands networked

radare demo

... demo here

Questions?

http://www.radare.org

ktxby!


SEAKING
ARS GRATIA ARTIS
FUCK
YEAH
WWW.FUCKYEAHSEAKING.COM

ktxbv!