radare2: from forensics to bindiffing

nibble <nibble@develsec.org> (@nibble_ds)

pancake <pancake@nopcode.org> (@trufae)

@radareorg

/Rootəd° CON 2011

3-4-5 Marzo 2011

Madrid

powered by
ANO
.lolcathost.org

Introduction

radare was born as a forensics tool

- 64 bit addressing
- multiple searching methods (aes, bytes, binmask..)
- flags (mark with name, offset and size)
- local and remote io (rap:// w32:// dbg:// ..)

New stuff:

- filesystems and partitions
- zoom mode (overview of file)
- base64 encoding/decoding
- magic templates
- scripting in Vala (fast!)

Demo

Opening a remote disk and search for a string

```
$ sudo r2 -n rap://:9999
```

```
$ r2 -n rap://127.0.0.1:9999//dev/sda
> / hello world
f hit0_0 11 0xfad040
> ./ hello world
> ? hit0_0
0xfad040
> x @ 0xfad040
```

Search methods

Keyword:

- regular expressions (/e)
- text (string, wide string, utf8, ..) (/w)
- hexpair buf + binary mask (/x)

Patterns:

- repeated sequences of bytes (/p)
- expanded AES keys (/A)

Analysis:

- references to addresses (call, jmp, ..) (/a)
- opcodes matching a given expreg (/c)

Signatures

You can create and find hexpair-based templates.

- automatic binary masks based on opcode args
- useful for statically linked bins
- find inlined or dupped symbols

"z is for zignature"

```
> zg ls > ls.zignaturez
```

```
> . ls.zignaturez
> .z/
```

Magic templates

magic(4) is a common library in *NIX systems which uses a db to identify and parse data

```
> pm
data
```

to create our own templates to parse memory data

```
> !vim test.mgc
> pm test.mgc
```

```
$ ls file-*/magic/Magdir
```

Magic example

This is a example of the file format.

```
0 long 0 This is a null reference
0 byte x one %d,
>4 byte x two %d,
>8 string FOO (type is foo)
>8 string BAR (type is bar)
>12 long&0xff >0x70 invalid type
```

Formatted memory

There's also a native formatted print command:

```
> pf [format] [space separated field names]
```

```
[0x04d80480]> pf dis next length string
  next: 0x4d80480: 0x4d80520
length: 0x4d80484: 12
string: 0x4d80488: "backandforth"
```

Scripting

libr/include files are described in swig/vapi/*.vapi

valaswig can translate those vapi files into working bindings for many scripting languages:

- python, perl, ruby, lua, java, guile, go, and vala

* Run from r2 prompt with the #! command
* Run as a standalone program using the r2-swig

Scripting demo

```
[0x8048404]> #!vala
> print ("0x%08llx\n", core.num.get ("entry0"));
0x080498d0
```

```
[0x8048404]> #!python
> core.cmd0 ("pd")
> core.cons.flush ()
0x08049900    0     55                 push ebp
0x08049901    4+    89e5               mov ebp, esp
0x08049903    4     53                 push ebx
0x08049904    8+    83ec04             sub esp, 0x4
```

Filesystems

Supports ext2, ntfs, vfat, reiserfs, ... based on BURG.

```
$ r2 -nw diskimg.ext2
> m ext2 /mnt 0
> md /mnt
foo
> mg /mnt/foo
Hello World
> mo /mnt/foo
offset = 0x37490
size = 12
> ps @ 0x37490:12
Hello World
> w Diiee @ 0x37490
> ms   # mountpoint shell
```

Partitions

Based on GRUB code:

- Supports msdos, gpt, bsd, apple, sun, and more

```
$ r2 -n /dev/sda
> mp msdos 0
0 83 0x087e00 0x0865f9a00
1 82 0x0865f9a00 0x08168d5c00
2 83 0x08168d5c00 0x081ebbc5600
3 83 0x081ebbc5600 0x081ffd62800
```

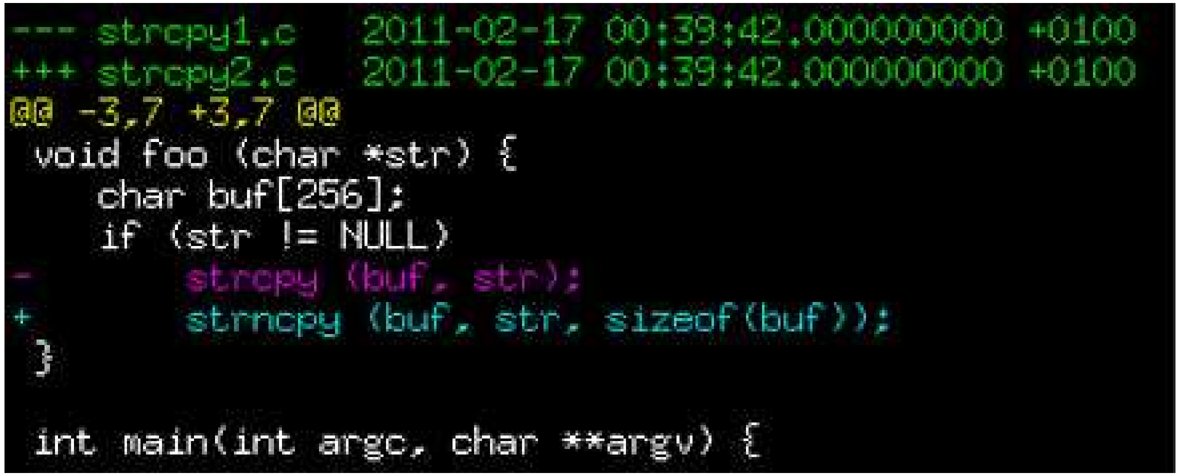
Bindiffing

- What is bindiffing?

- Why is this useful?
  - Patched bins
  - Analyze backdoored bins
  - Find new functions (maybe non-documented)
  - Locate different implementations between functions in similar bins

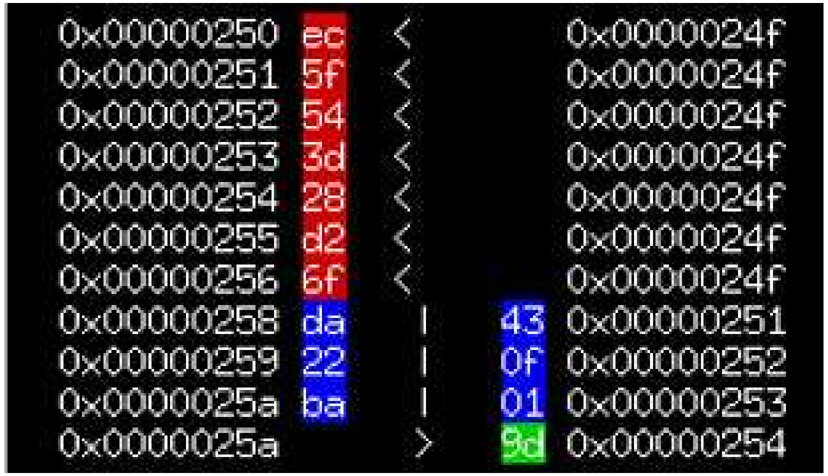
Plain text diffing vs Binary diffing

- Text/Code is written in a natural way for humans
- Can be splitted by lines
- Doesn´t exist dependencies/references between one line and another
- One "instruction" is always coded the same
- There isn´t intrinsic data to extract for each line



```
--- strcpy1.c    2011-02-17 00:39:42.000000000 +0100
+++ strcpy2.c    2011-02-17 00:39:42.000000000 +0100
@@ -3,7 +3,7 @@
 void foo (char *str) {
     char buf[256];
     if (str != NULL)
-        strcpy (buf, str);
+        strncpy (buf, str, sizeof(buf));
 }

 int main(int argc, char **argv) {
```



```
0x00000250 ec  <        0x0000024f
0x00000251 5f  <        0x0000024f
0x00000252 54  <        0x0000024f
0x00000253 3d  <        0x0000024f
0x00000254 28  <        0x0000024f
0x00000255 d2  <        0x0000024f
0x00000256 6f  <        0x0000024f
0x00000258 da   |    43 0x00000251
0x00000259 22   |    0f 0x00000252
0x0000025a ba   |    01 0x00000253
0x0000025a      >    9d 0x00000254
```

Troubles

- Discard useless data
  - Padding
  - Uninitialized data
  - Useless sections/segments
- Tokenization
  - Several Options: Fcns, BBs, Opcodes, Bytes
  - Combination
- Deltas
- Presentation

Steps

1.- Code Analysis (Do you remember RAnal? ;)

- Find functions and bb´s (recursively)
- Extract data from opcodes

2.- Fingerprint BB´s

3.- Fingerprint Fcn´s based on BB´s

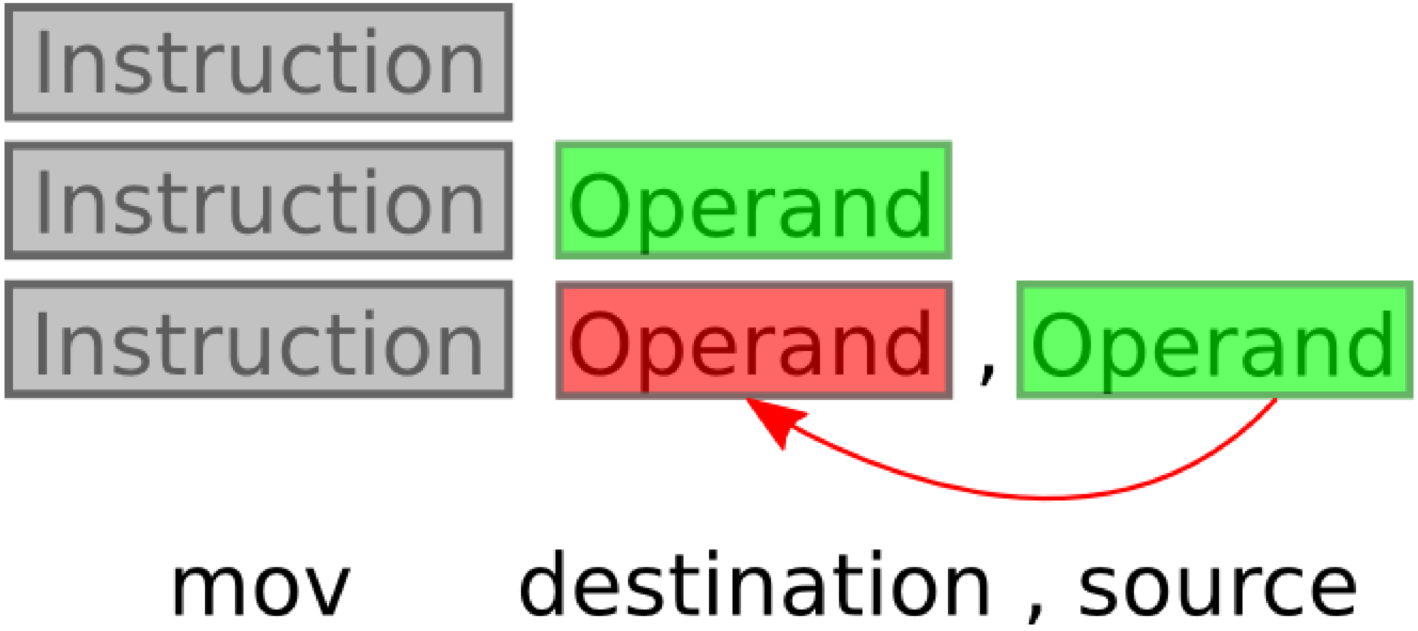
4.- Function matching by name (exports)

5.- Function matching based on fingerprints

6.- BB matching

Fingerprinting

- Use of Binary masks
- RAnal info
- Graph based metrics



BB/Fcn Diffing

Levenshtein distance relative to entity size

```
  Minimum number of edits needed to transform one string into
  the other
```

Example:

```
  "rooted" vs "roted"     -> d = 1
  "rooted" vs "r-ooted"   -> d = 1
  "rooted" vs "r-oted"    -> d = 1
  "rooted" vs "rooted---" -> d = 3
  "rooted" vs "-roo--ted" -> d = 3
```

Demos

- Demo 1: Simple diff
- Demo 2: Diff between similar apps
- Demo 3: Backdoored bin

And… a little surprise

ragui: the ui

It´s not yet ready for daily use..

- work in progress
- based on GNOME technologies
- runs on Windows/OSX/Linux/BSD without changes
- show screenshots and demo

Questions?

/Rooted°CON 2011

3-4-5 Marzo 2011
Madrid

radare2: from forensics to bindiffing

nibble <nibble@develsec.org> (@nibble_ds)

pancake <pancake@nopcode.org> (@trufae)

@radareorg

/Rootəd° CON 2011

3-4-5 Marzo 2011

Madrid

powered by

ANO

.lolcathost.org