radare2: evolution

pancake <pancake@nopcode.org>
@trufae
I don’t plan to make another introduction to what r2 is..

5 years of development brings us many goods

but ...
U Y NO 1.0 YET?
The main reason why we don’t have r2-1.0 yet:
What’s new?

- lot of bugfixes
- tiny bins
- r_egg lib
- sdb nosql
- optimizations
- new ports
- new bindings
- ui
REgg

eggs are small pieces of code with a specific purpose that aim to be injected somewhere else.

– shellcodes are eggs to provide a shell
– eggs should be relocatable
– eggs must not violate segment perms

r_egg api provides a simple compiler that generates relocatable code for intel–32/64 and arm platforms.

Work in progress support for ROP compilation.
r_egg compiles a C–like language into assembly which is passed to r_asm to generate blobs. Those blobs are then pushed to r_bin in order to create a native system binary.

```
$ ragg2 -f elf -b 32 -o hi hi.r
$ ./hi
Hello World!
```

r_egg is also used by ‘r2 –p’ to apply rapatches (see lacon2k10 talk)
REgg example

#!/usr/bin/ragg2 -X
main();

// OSX syscall definitions
write@syscall(4);
exit@syscall(1);
syscall() {
  : mov eax, `.arg'
  : push eax
  : int 0x80
}

main@global(128, 128) {
  write (1, "hello world\n", 12);
  exit(0);
}

RBin api now supports to create tiny binaries:

- elf – x86–32/64, arm  (84 bytes)
- mach0 – x86–32/64, arm   (232 bytes)
- pe – windows (x86–32)   (97 bytes)

Only text and experimental data sections supported.
Future support for CLASS, DEX, pe64, p9bins, ..

```bash
$ rabin2 -c elf:cc a.out
$ ./a.out
It’s a trap!
```
Magic signatures are now native and portable, no more deps.

Search and examine complex data structures using the file(1) magic database syntax.

Code imported from OpenBSD
  – adapted to r2 api
  – fix and report segfault

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>string</td>
<td>\177ELF</td>
<td>ELF</td>
</tr>
<tr>
<td>&gt;4</td>
<td>byte</td>
<td>0</td>
<td>invalid class</td>
</tr>
<tr>
<td>&gt;4</td>
<td>byte</td>
<td>1</td>
<td>32-bit</td>
</tr>
<tr>
<td>&gt;4</td>
<td>byte</td>
<td>2</td>
<td>64-bit</td>
</tr>
<tr>
<td>&gt;5</td>
<td>byte</td>
<td>1</td>
<td>LSB</td>
</tr>
</tbody>
</table>

...
NoSQL is trendy.. so let’s ride it!

sdb is a minimalist memcache–like key value database written by me in C.

notes:
- same speed as memcache in tcp/plain
- smaller memory footprint
- on–disk storage with modified cdb
- api for client and server and bindings
- usable as a library instead of networked
- string based. no binary data allowed
- atomic storage, no data corruption
Good things about key value databases:

- allow tree node iterations (like in cassandra)
  - nested hashtables
- data structures can be implemented on top of it
  - linked lists, hashtables, arrays, graphs, trees
- no schemas, no initial configuration
  - flexible as long as you design in runtime
- constant request time
- fault-tolerant and easily scalable (DHS)
SDB nosql database

Used to describe syscall description tables.

I plan to use it as standard database for r2

- debug and source information
- binary information
- code analysis
- flags
- comments
- ...

Optimizations

Doing huge code analysis tasks

- Working with huge bins is now less painful :)
- Fixed lot of bugs thanks to Valgrind and OpenBSD
- sdb allows to store big stuff with fast access and few mem
- r_th is a wrapper for pthread and w32 thread apis (r2 -t)
- Using related data structures and caching results
- valgrind, oprofile, dtrace, gprof, ...
Farming!
Farming!

As long as I’ve been mostly coding alone I decided to write a build farm to:

- ease the build and install to new users
- looks like ./configure ; make ; make install is too hard
- python bindings installer with dependency facilities
- automatic report of compile errors for multiple platforms

- gnu/linux–x86–32/64, arm
- OSX
- android–x86–32/64
- mingw32/64
New ports

r2 is now known to work on the following platforms:

- android-x86/arm – native using NDK
- meego – Harmattan Nokia n950 / n9
- windows 64 bit – mingw64
- osx lion – support for PIE bins
- GNU/kFreeBSD – thanks debian
- *bsd – thanks openbsd! (Edd Barret)
New languages have been added to the bindings family:

- newlisp and guile (yay! parenthesis!)
- c++ (basic object oriented class facilities)
- javascript (v8gear)
- gir (gobject introspection runtime)
maybe r2 looks scary, but GUIs are friendly!
- ragui development has been stopped for a year
- @hteso is writing a python-based gui known as bokken
- split development by high/low level designs is good

- many people is using the r2 api for their own projects
  - v8 javascript code analysis engine on top of r_anal
  - malware signature search with r_search and r_sign
  - rop gadget search tool
  - ...

Bokken is a binary analysis UI for pyew and r2 written in python. Still not production ready.

Hexadecimal viewer, disassembler, graphs, search for keywords, bytes, etc.. and code analysis

http://inguma.eu/projects/bokken/

hg clone http://inguma.eu/repos/bokken
Bokken shot
Ragui is closed source app written in GtkAML and Vala

- Works on all major platforms (win, mac, lin)
- 4 modes: editor, disassembler, debugger, forensics
- Still work in progress and not ready to use
- UI for mounting filesystems and dumping files
- Code graph viewer using self-written graph library
- Will be released at some point
- Betatestes must contact @radareorg on Twitter
Raqui shot
Call for developers!
Future plans

- Keep refactoring the core and making the libs better
- Enhance UIs (call for developers!)
- Focus on optimizations and speed
- Better debugger for ios/osx, w32 and gdb
- Add support for Bochs and windbg
- Support for classes in r_bin
- Enhance dalvik platform support
Questions?