radare2: evolution

pancake <pancake@nopcode.org>

@trufae

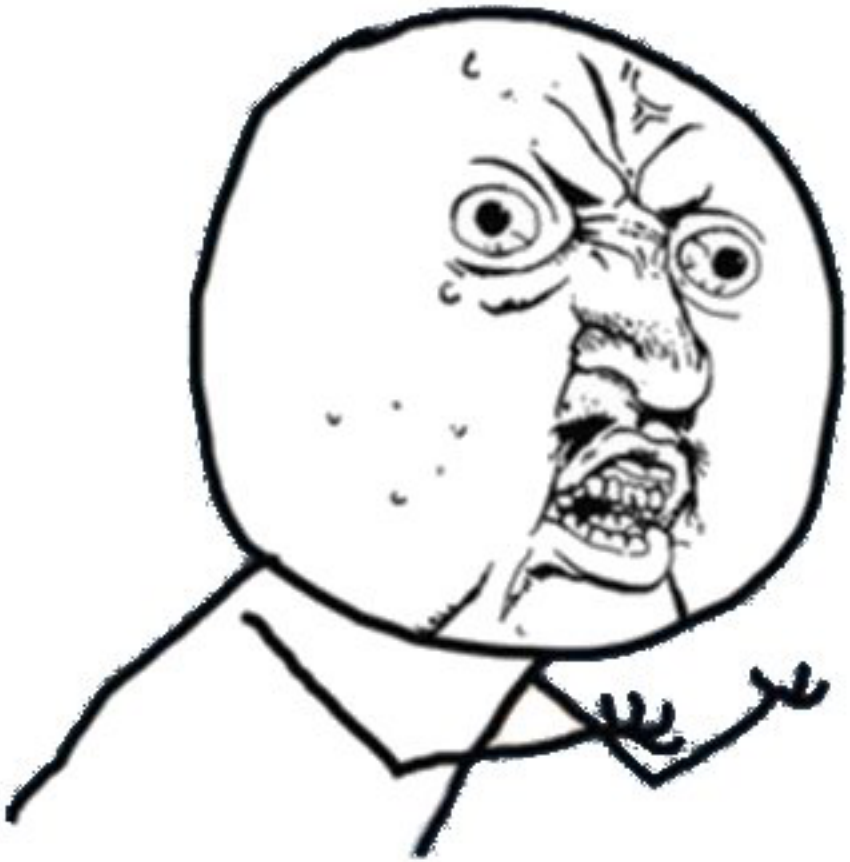
Oh hi!

I don’t plan to make another introduction to what r2 is..

5 years of development brings us many goods

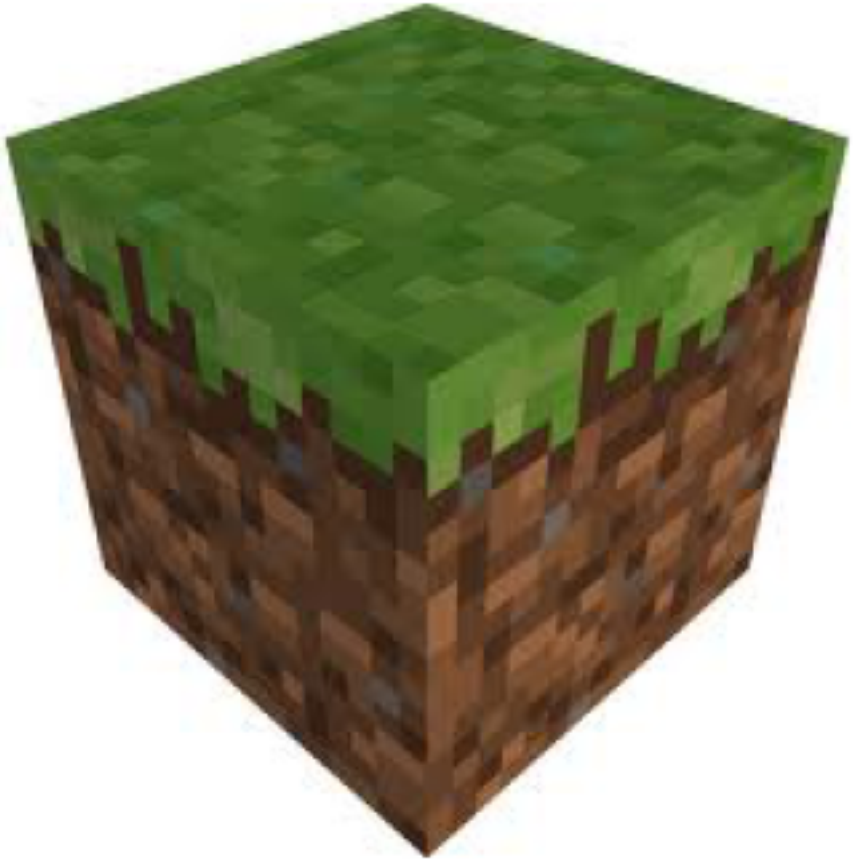
but ...

U Y NO 1.0 YET?



Minecraft

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.

REgg internals

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);
}
```

Tiny binaries

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, ..

```
$ rabin2 -c elf:cc a.out
$ ./a.out
It's a trap!
```

RMagic

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

```
0       string          \177ELF         ELF
>4      byte            0               invalid class
>4      byte            1               32-bit
>4      byte            2               64-bit
>5      byte            1               LSB
...
```

SDB nosql database

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

SDB nosql database

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 know 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)

Bindings

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)

UI

maybe r2 looks scary, but GUIs are friendly!

UI

- 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

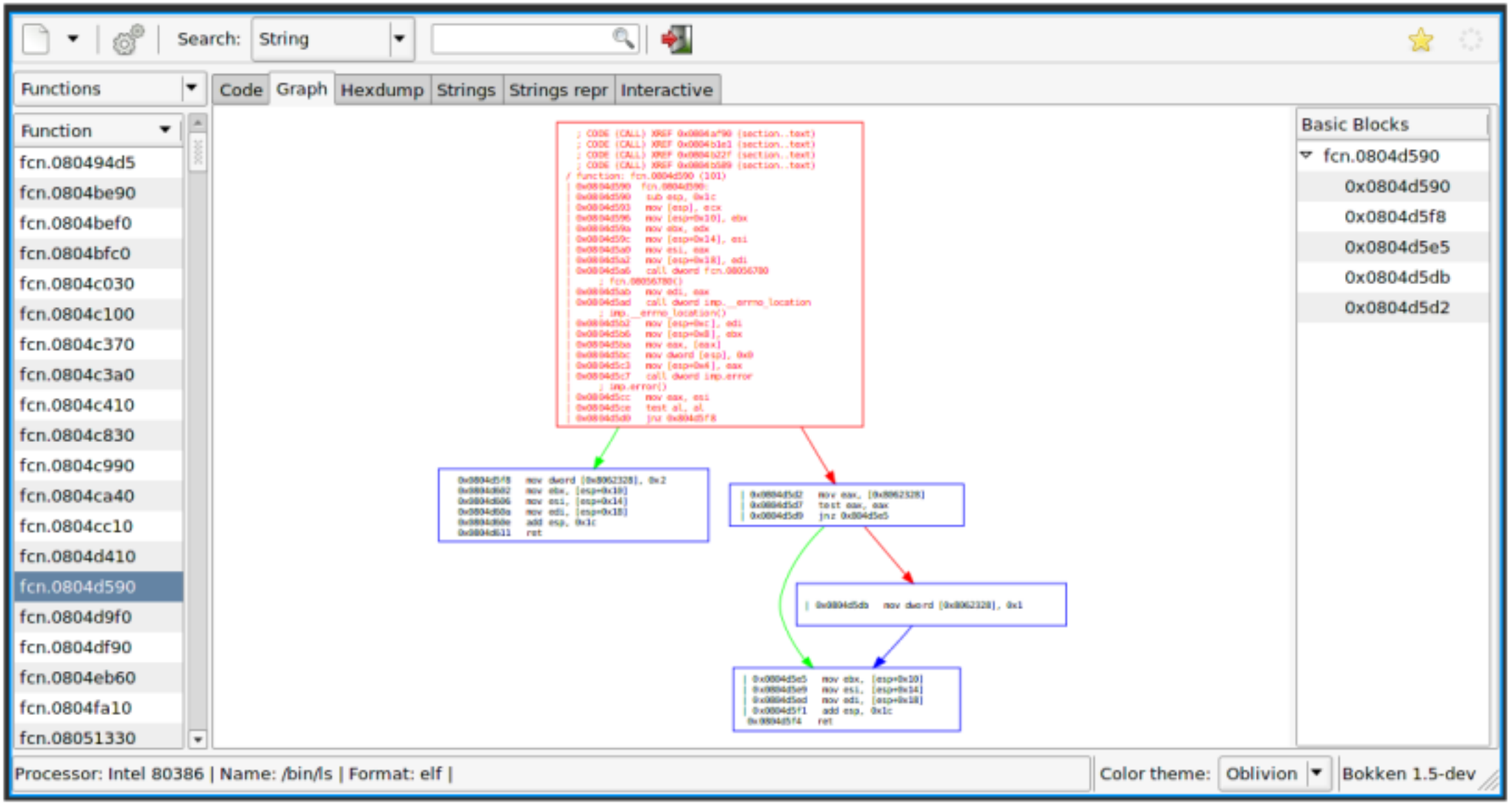
Bokken is a binary analysis UI for pyew and r2 written in python. Still not production ready.

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

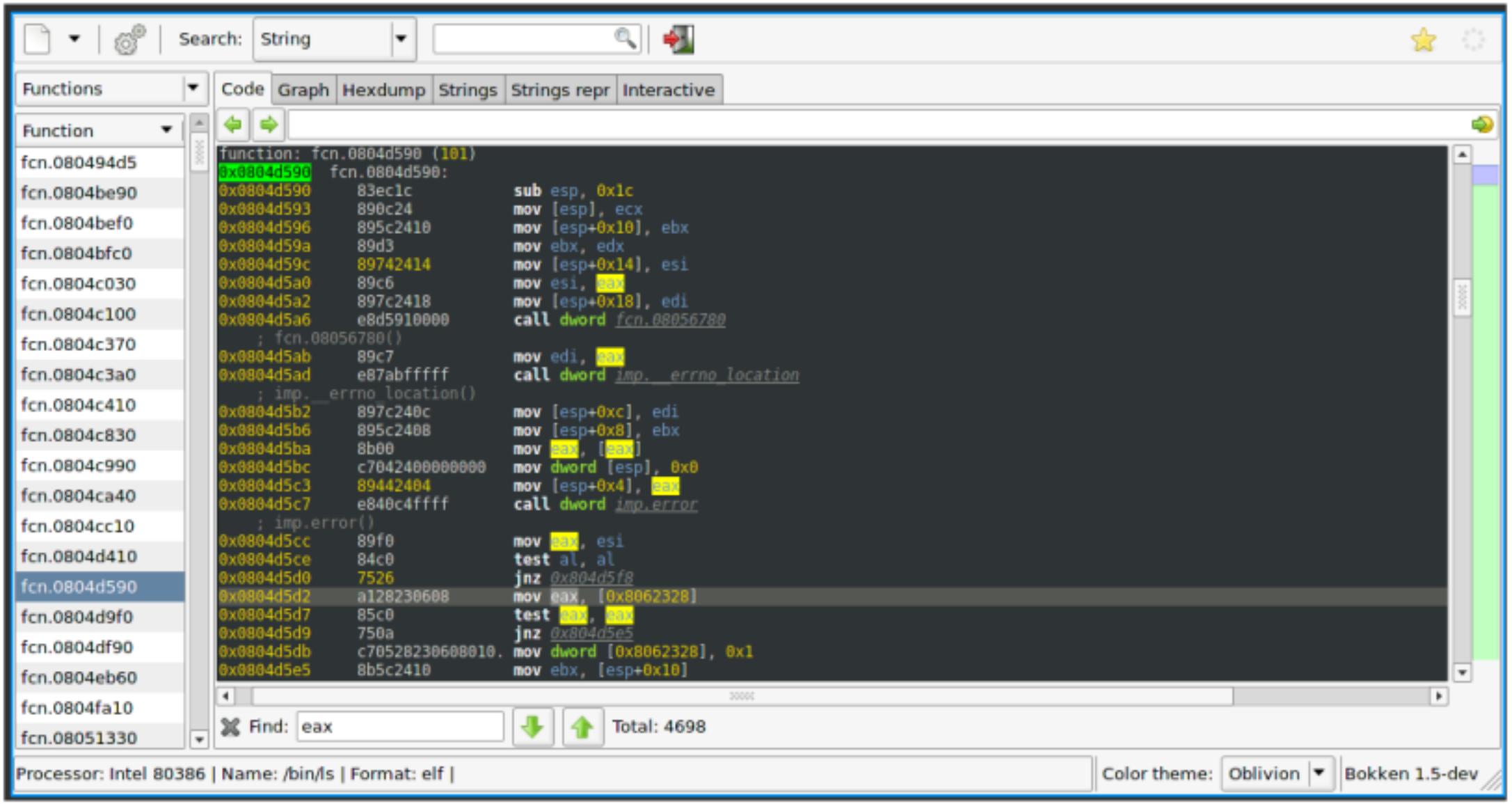
```
http://inguma.eu/projects/bokken/
```

```
hg clone http://inguma.eu/repos/bokken
```

Bokken shot


Search:
String
Functions
Code
Graph
Hexdump
Strings
Strings repr
Interactive
Function
fcn.080494d5
fcn.0804be90
fcn.0804bef0
fcn.0804bfc0
fcn.0804c030
fcn.0804c100
fcn.0804c370
fcn.0804c3a0
fcn.0804c410
fcn.0804c830
fcn.0804c990
fcn.0804ca40
fcn.0804cc10
fcn.0804d410
fcn.0804d590
fcn.0804d9f0
fcn.0804df90
fcn.0804eb60
fcn.0804fa10
fcn.08051330
Basic Blocks
fcn.0804d590
0x0804d590
0x0804d5f8
0x0804d5e5
0x0804d5db
0x0804d5d2
Processor: Intel 80386 | Name: /bin/ls | Format: elf |
Color theme:
Oblivion
Bokken 1.5-dev

Bokken shot


Search: String
Functions
Code
Graph
Hexdump
Strings
Strings repr
Interactive
Function
fcn.080494d5
fcn.0804be90
fcn.0804bef0
fcn.0804bfc0
fcn.0804c030
fcn.0804c100
fcn.0804c370
fcn.0804c3a0
fcn.0804c410
fcn.0804c830
fcn.0804c990
fcn.0804ca40
fcn.0804cc10
fcn.0804d410
fcn.0804d590
fcn.0804d9f0
fcn.0804df90
fcn.0804eb60
fcn.0804fa10
fcn.08051330
function: fcn.0804d590 (101)
0x0804d590 fcn.0804d590:
0x0804d590 83ec1c sub esp, 0x1c
0x0804d593 890c24 mov [esp], ecx
0x0804d596 895c2410 mov [esp+0x10], ebx
0x0804d59a 89d3 mov ebx, edx
0x0804d59c 89742414 mov [esp+0x14], esi
0x0804d5a0 89c6 mov esi, eax
0x0804d5a2 897c2418 mov [esp+0x18], edi
0x0804d5a6 e8d5910000 call dword fcn.08056780
; fcn.08056780()
0x0804d5ab 89c7 mov edi, eax
0x0804d5ad e87abfffff call dword imp.__errno_location
; imp.__errno_location()
0x0804d5b2 897c240c mov [esp+0xc], edi
0x0804d5b6 895c2408 mov [esp+0x8], ebx
0x0804d5ba 8b00 mov eax, [eax]
0x0804d5bc c7042400000000 mov dword [esp], 0x0
0x0804d5c3 89442404 mov [esp+0x4], eax
0x0804d5c7 e840c4ffff call dword imp.error
; imp.error()
0x0804d5cc 89f0 mov eax, esi
0x0804d5ce 84c0 test al, al
0x0804d5d0 7526 jnz 0x804d5f8
0x0804d5d2 a128230608 mov eax, [0x8062328]
0x0804d5d7 85c0 test eax, eax
0x0804d5d9 750a jnz 0x804d5e5
0x0804d5db c70528230608010. mov dword [0x8062328], 0x1
0x0804d5e5 8b5c2410 mov ebx, [esp+0x10]
Find: eax
Total: 4698
Processor: Intel 80386 | Name: /bin/ls | Format: elf |
Color theme: Oblivion
Bokken 1.5-dev

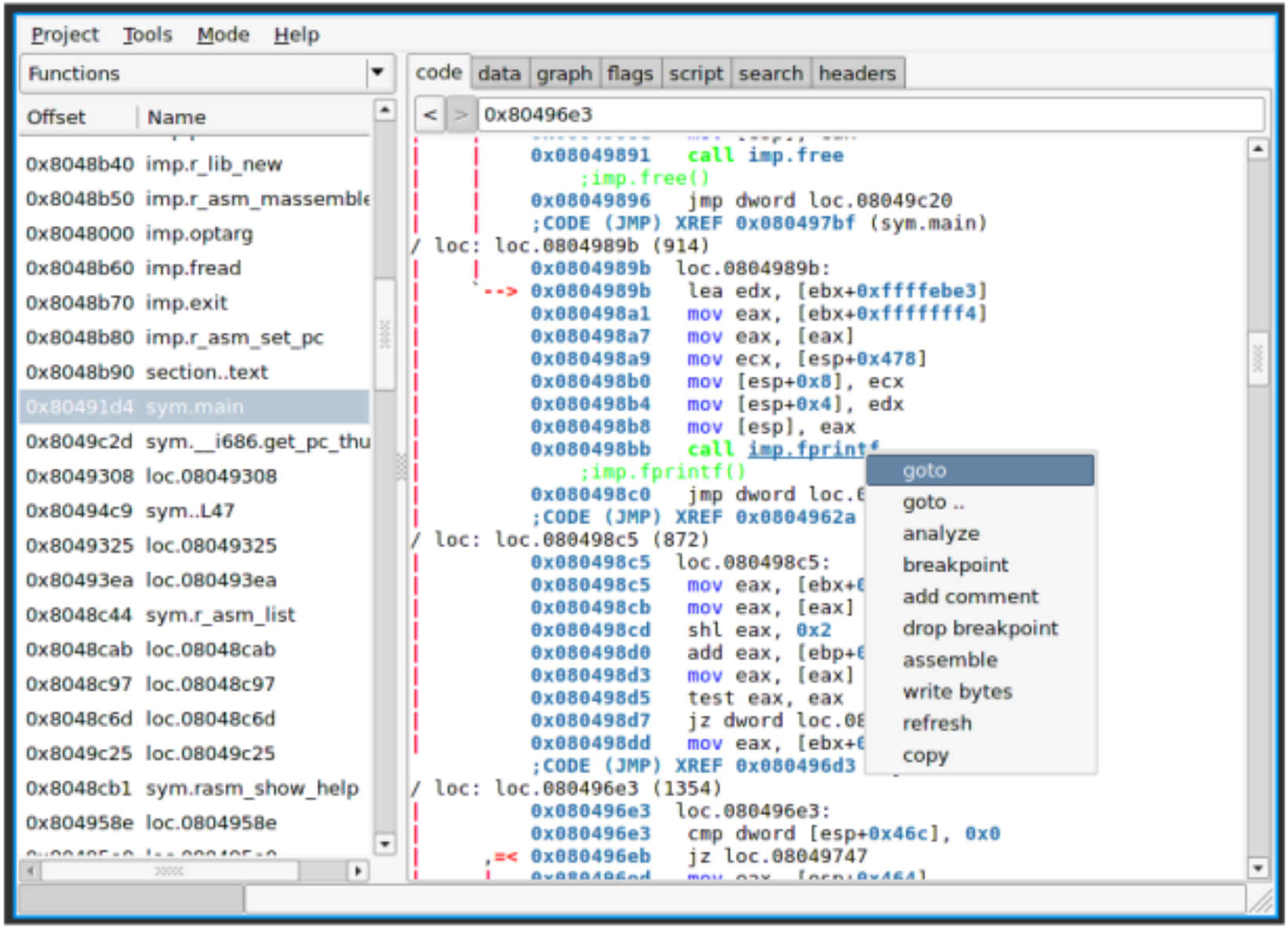
Demo

Ragui

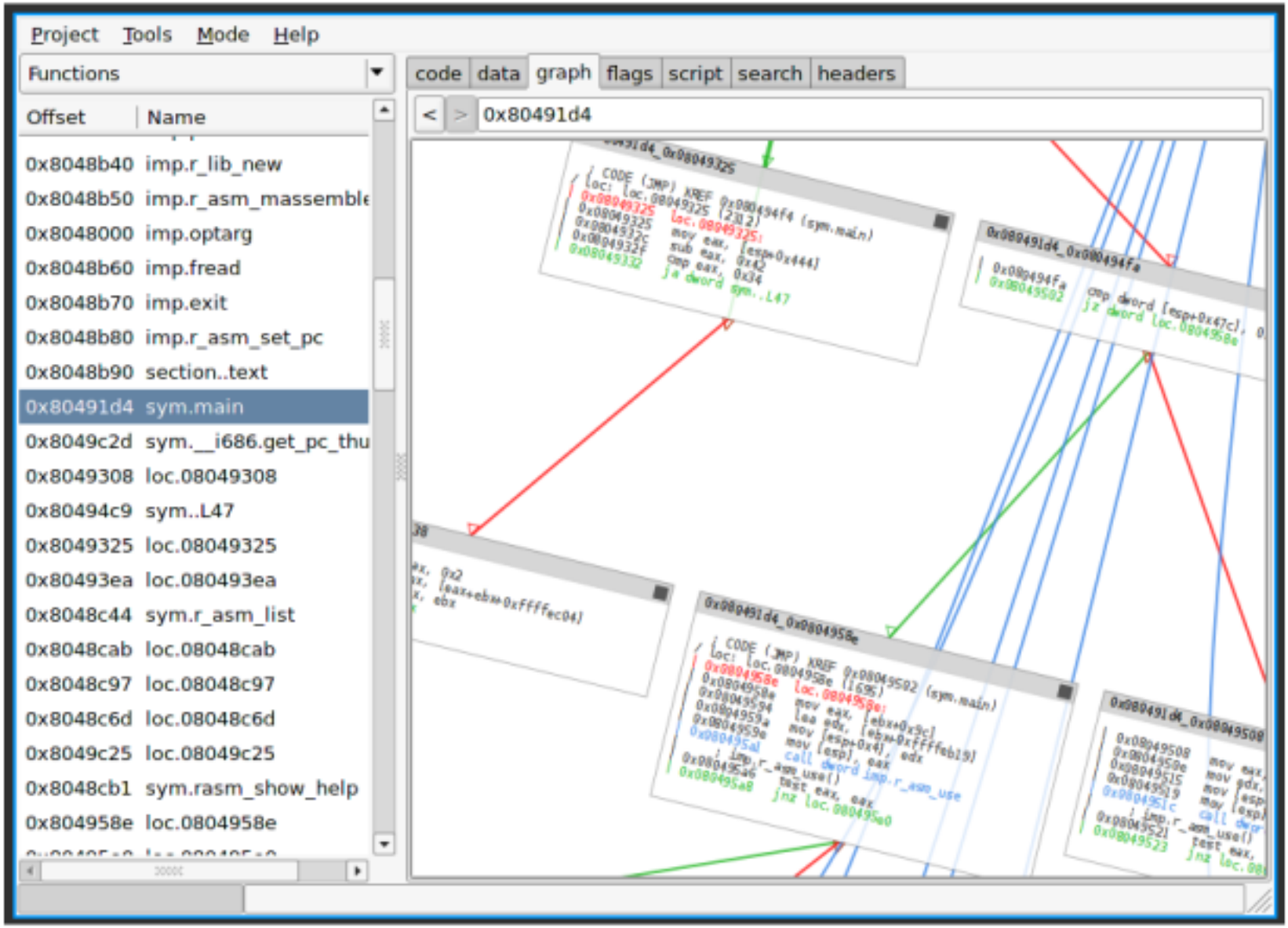
Ragui is closed source app written in GtkAML and Vala

- 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
- Betatesters must contact @radareorg on Twitter

Ragui shot


Project Tools Mode Help
Functions
Offset Name
0x8048b40 imp.r_lib_new
0x8048b50 imp.r_asm_massemble
0x8048000 imp.optarg
0x8048b60 imp.fread
0x8048b70 imp.exit
0x8048b80 imp.r_asm_set_pc
0x8048b90 section..text
0x80491d4 sym.main
0x8049c2d sym.__i686.get_pc_thu
0x8049308 loc.08049308
0x80494c9 sym..L47
0x8049325 loc.08049325
0x80493ea loc.080493ea
0x8048c44 sym.r_asm_list
0x8048cab loc.08048cab
0x8048c97 loc.08048c97
0x8048c6d loc.08048c6d
0x8049c25 loc.08049c25
0x8048cb1 sym.rasm_show_help
0x804958e loc.0804958e
code data graph flags script search headers
0x80496e3
0x08049891 call imp.free
;imp.free()
0x08049896 jmp dword loc.08049c20
;CODE (JMP) XREF 0x080497bf (sym.main)
loc: loc.0804989b (914)
0x0804989b loc.0804989b:
0x0804989b lea edx, [ebx+0xffffebe3]
0x080498a1 mov eax, [ebx+0xfffffff4]
0x080498a7 mov eax, [eax]
0x080498a9 mov ecx, [esp+0x478]
0x080498b0 mov [esp+0x8], ecx
0x080498b4 mov [esp+0x4], edx
0x080498b8 mov [esp], eax
0x080498bb call imp.fprintf
;imp.fprintf()
0x080498c0 jmp dword loc.
;CODE (JMP) XREF 0x0804962a
loc: loc.080498c5 (872)
0x080498c5 loc.080498c5:
0x080498c5 mov eax, [ebx+
0x080498cb mov eax, [eax]
0x080498cd shl eax, 0x2
0x080498d0 add eax, [ebp+
0x080498d3 mov eax, [eax]
0x080498d5 test eax, eax
0x080498d7 jz dword loc.0
0x080498dd mov eax, [ebx+
;CODE (JMP) XREF 0x080496d3
loc: loc.080496e3 (1354)
0x080496e3 loc.080496e3:
0x080496e3 cmp dword [esp+0x46c], 0x0
0x080496eb jz loc.08049747
goto
goto ..
analyze
breakpoint
add comment
drop breakpoint
assemble
write bytes
refresh
copy

Ragui shot


Project
Tools
Mode
Help
Functions
Offset
Name
0x8048b40 imp.r_lib_new
0x8048b50 imp.r_asm_massemble
0x8048000 imp.optarg
0x8048b60 imp.fread
0x8048b70 imp.exit
0x8048b80 imp.r_asm_set_pc
0x8048b90 section..text
0x80491d4 sym.main
0x8049c2d sym.__i686.get_pc_thu
0x8049308 loc.08049308
0x80494c9 sym..L47
0x8049325 loc.08049325
0x80493ea loc.080493ea
0x8048c44 sym.r_asm_list
0x8048cab loc.08048cab
0x8048c97 loc.08048c97
0x8048c6d loc.08048c6d
0x8049c25 loc.08049c25
0x8048cb1 sym.rasm_show_help
0x804958e loc.0804958e
code
data
graph
flags
script
search
headers
0x80491d4

Demo

Call for developers!

Future plans

- Keep refactoring the core and making the libs better
- Enhace 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
- Enhace dalvik platform support

Questions?