Radare Demystified

r2@33C3/2016

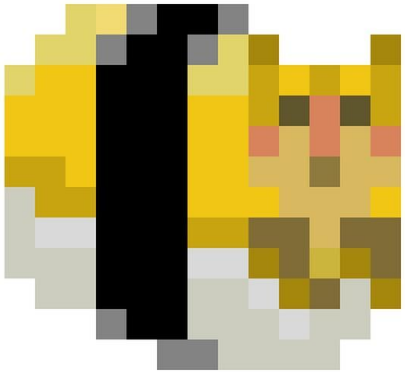
pancake@nopcode.org

Introduction.

Who am I?

Sergi Àlvarez // pancake // @trufae

- Working at **NowSecure** as a Mobile Security Analyst doing R+D.
- Author of **radare(1+2)**, Acr, Valabind and many other open-source tools.
- Messing with Bluetooth, Coding asm video codec optimizations for x86, arm and mips, IoT firmware dev, SexyPanda @Defcon CTF, Forensics, Sysadmin, Web and C developer.



What's Radare?

- Free and OpenSource RE Framework
- Focus on portable, extensible, expressive
- Hobby project started in 2006
- Full rewrite in 2009
- Few contributors until 2013
- Mainly developed by me
- Switching from developer to maintainer
- About 500 users in irc/telegram
- ~6200 followers on Twitter
- First r2con last September in BCN
- 3rd year organizing a Summer Of Code

Stands For ‘Raw Data Recovery’

- Hexadecimal Editor
- Assembler / Disassembler
- Support lot of file formats and archs
- Static / Dynamic Analysis
- Hash / Entropy / BinDiffing
- Debugger / Emulator
- ROP Finder / Payload Generator
- Scripting support for many languages
- Plugins / Package Manager

Very portable

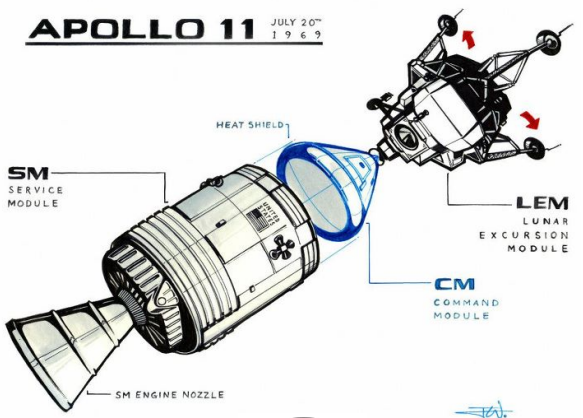
- Linux/Android
- macOS/iOS
- Windows
- QNX

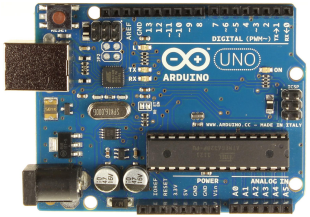
$ rasm2 -L

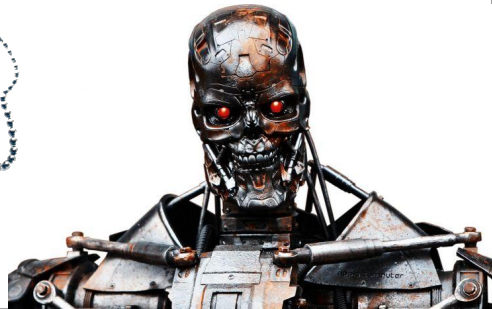
$ rabin2 -L

$ r2pm -s

What can I inspect?







Wait..

Don’t lose the rail


Myths.

Myths

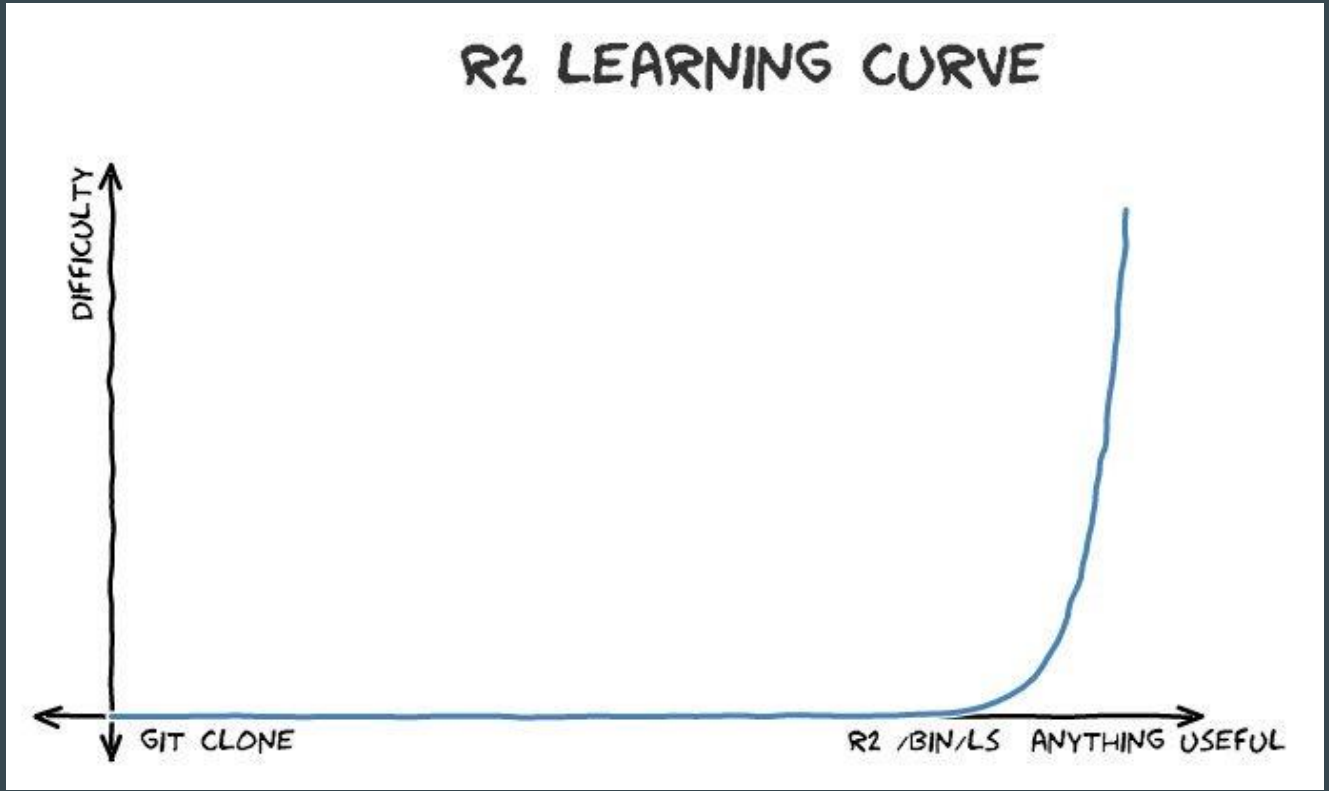
- It’s not Stable
- It’s Difficult
- So Many Commands
- Hard to Remember
- It’s Buggy
- Can’t Decompile
- Broken Debugger
- It’s not Written in Python
- Can’t Assemble
- There’s no Graphs
- I Can’t Pay for it
- Bindings are Not Working
- Nobody Uses it
- Not Documented
- There is no GUI
- Doesn’t Support XXX arch
- It’s Slow
- API not stable

But First.. Let's Make a Poll

- How many of you know radare?

- How many of you use r2?


It’s Difficult.


R2 LEARNING CURVE
DIFFICULTY
GIT CLONE
R2 /BIN/LS
ANYTHING USEFUL

It's Difficult

Learning curve is steep, but from my experience, people need from 2 days to 1 week to get used to it.

(Comparable to Perl, Vim or Git).

- So many commands
- Hard to remember

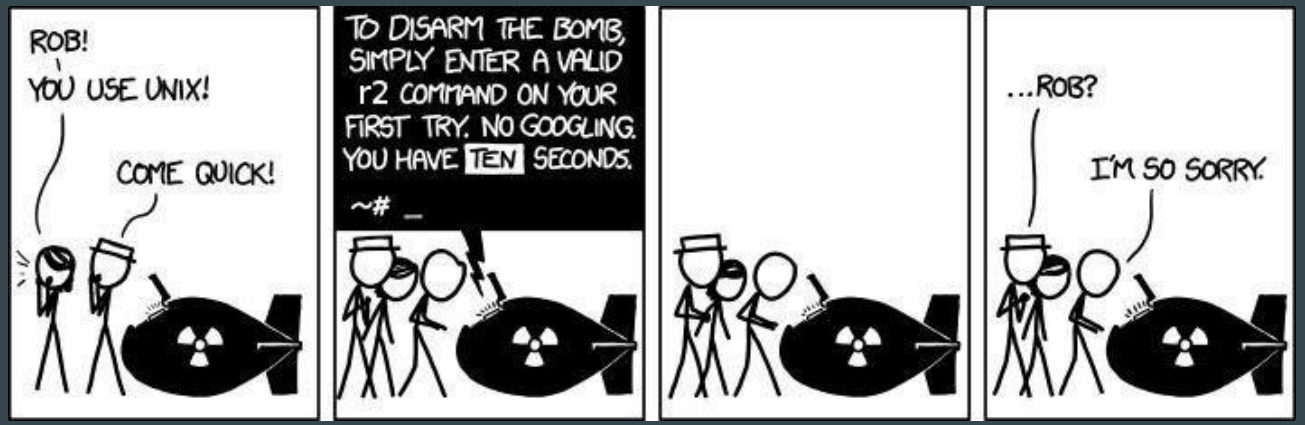
In the other hand...

- Very Expressive and Flexible
- Easy to Modify and Extend
- Build Tools on Top of r2

So Many Commands

The amount of commands you have to remember is pretty little.

- Mnemonic Commands (short in length, each letter have a meaning)
- Unix-like shell (pipes, redirections, grep, less, json-indent, ...)
- Orthogonal (mix commands and modifiers to express your wishes)



So Many Commands

Just remember 5 commands to do most of the tasks:

- s - seek
- pd/px ~ print disasm / hexdump
- wa/wx ~ write assembly / hexpairs
- v ~ enter visual mode
- q ~ quit

Command Modifiers: ., ?, @, ~, >, |, @@, ~!, “, \

Advanced Commands: i, af, agf, dr, S=

Hard To Remember

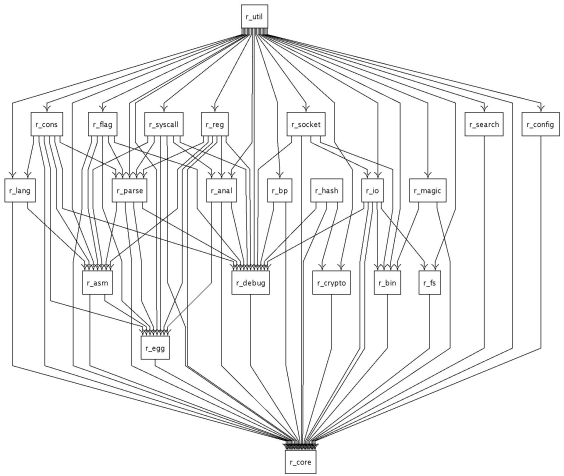
Commands follow simple mnemonic rules:

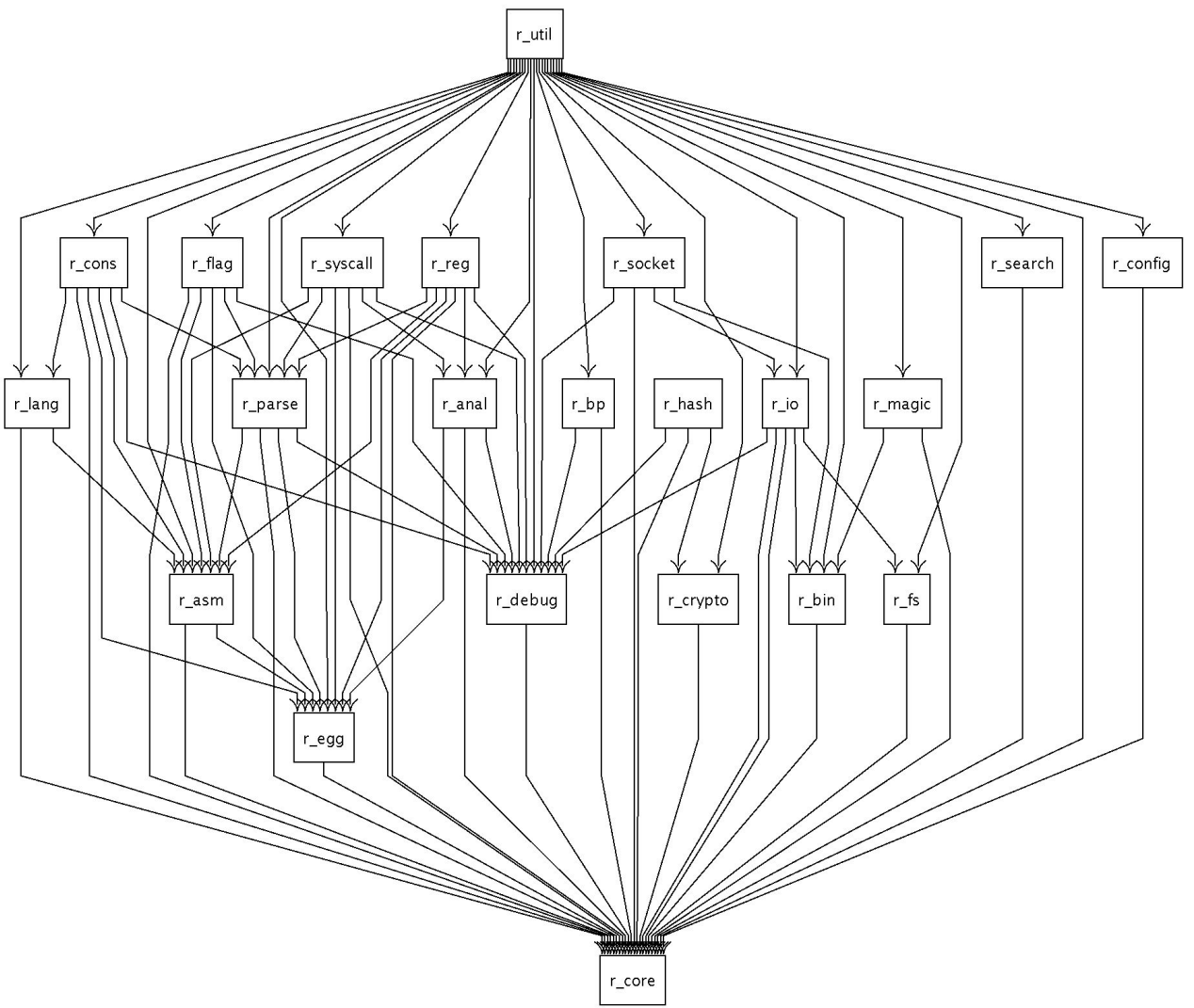
- Each char in the command is a subcommand of the previous one.
  - px -> print hexdump
  - pd -> print disasm
  - af -> analyze function
  - is -> info symbols
- Append ‘?’ to the command to get help about it
- Prefix with ‘.’ to interpret the output as commands
- Backticks are also supported x `?v 33`
- Temporary seek with @

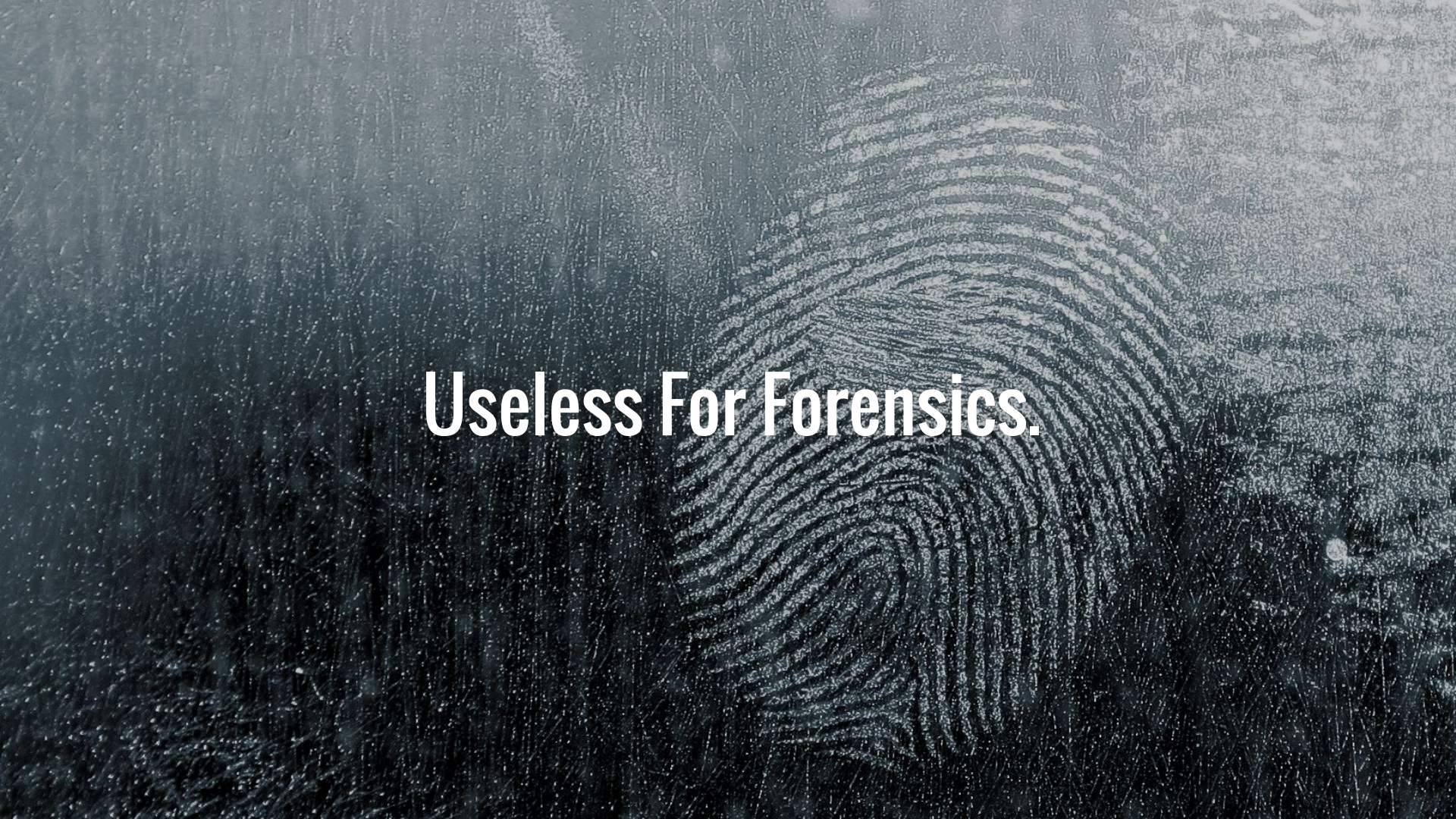
Structured

- libr/ modules with internal dependencies
- p/ plugins for each module
- binr/ programs
- shlr/ 3rd party ripped code

- APIs in continuous evolution (refactoring)
- Bindings automatically generated
- Core/R2 provides all functionality of other parts




r_util
r_cons
r_flag
r_syscall
r_reg
r_socket
r_search
r_config
r_lang
r_parse
r_anal
r_bp
r_hash
r_io
r_magic
r_asm
r_debug
r_crypto
r_bin
r_fs
r_egg
r_core


Useless For Forensics.

Forensics

Besides being the original aim of the tool, forensics is probably not one of their strong points.

The evolution of the project depends on user's interest and contributors, in order to support more filesystems, better introspection on data structures, etc

But still, r2 have some really valuable features on this field...

Forensics

- Open disk devices on all platforms as well as memory dumps
- r2k allows to read/write physical memory (r2pm -i r2k-linux; r2k://)
- Find patterns and analyze the results (/x)
- Mount filesystems and understand partitions (m)
- Carve dumps to identify known file headers with pm and /m
- Show data in structures (parsing .h files or format oneliners)
- Compute incremental or per-block checksums
- Pluggable IO to work with local or remote resources (see r2 -L)
- Support gzip:// ewf:// and other common forensics file formats

Forensics

(demo)

Carve Dump to find magic headers and extract them

Mount Filesystem

Show data structures


Analysis Is Slow.

Analysis By Default

- Blocking Operation that takes too long
- Doesn’t work for big binaries
- Takes a lot of time
- Doesn’t find all the functions
- The rule of ‘a’ after ‘a’
- Many analysis options and commands
- Choose carefully

http://radare.today/posts/analysis-by-default/


WAIT A MOMENT
I'M SURE AAE WILL FINISH SOON

Faster Analysis

- Go straight to the problem
- 90% of the time you don’t need a complete analysis
- Find refs much faster than any other tool
- Improving on every release
- Avoid the use of generic/dumb analysis
- Know your tools and choose wisely

- Generic loop for all archs (pluggable)
- Multiple iterations with different algorithms

Analysis Demo

(demo)

Analyze Functions, Check Code Coverage

Find references of strings, Patch Call

Use ESIL to emulate code and find computed references


Undocumented.

Documentation

As long as the project scope is pretty huge, there are tons of hidden functionalities and original uses of every single command people feel lost and disoriented and start asking for documentation. But the truth is..

- It’s already documented in C
- Inline help in every command by just appending a question mark
- I wrote a book for r1, and Maijin updated it for r2
- Tons of talks, slides, blog posts, youtube tutorials, ..

http://rada.re/r/docs.html


Cannot Decompile.

Can’t Decompile

Decompilation is hard, so it’s delegated to 3rd party tools

- Use retdec (r2pm plugin and online service)
- Radeco (started in the GSoC, wip, still not usable)
- Boomerang was supported for r1.
- Snowman Decompiler (r2pm -i r2snow)

Better Disasm

But r2 have some good disassembler capabilities

- Colorized instructions by type
- Variables/Arguments analysis
- Immediate replacement (and relative substitutions)
- Pseudo-Disassembly (add eax, 3 -> eax += 3)
- Summary (refs of strings and calls)
- AsmEmu (emulate code and add comments at right)
- Interactive Ascii-Art basic block and call control-flow Graphs

Better Disasm

(demo)


Not Stable.

It's Not Stable

Stability depends on

- The amount of crashes
- Commands and APIs changes.

So...

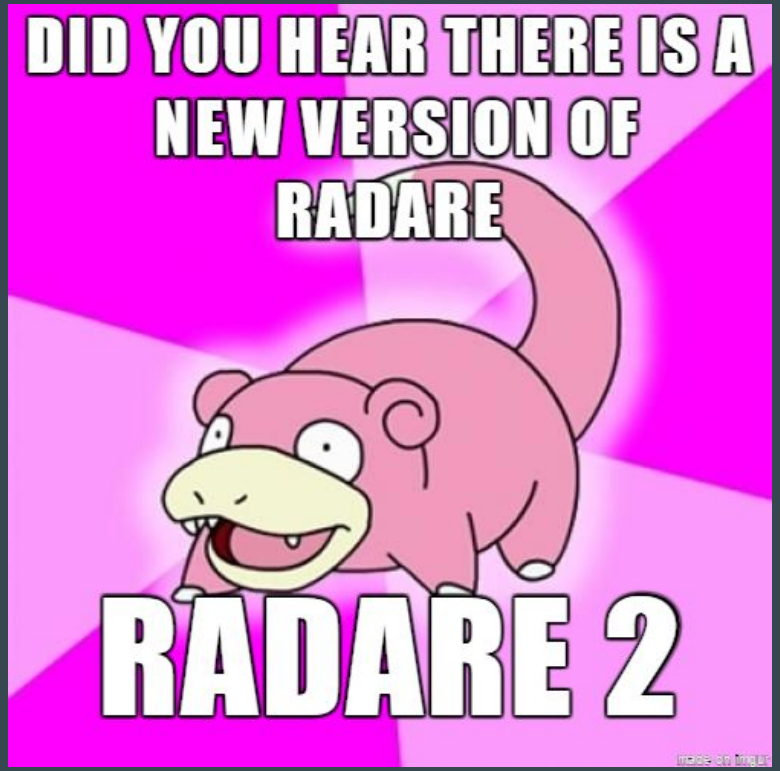
- We are now 1.x (announced in r2con)
- Most commands are not going to change
- JSON output eases parsing for automating with r2pipe
- The C API is quite stable, but there are continuous refactorings

Feature X is Broken

- Most common complains are already fixed in Git.
- Security bugs are fixed in less than 24h (usually 1-2h)
- Aim to follow the rule of “you see it you fix it”
- If not, write tests and fill issues in GitHub
- Very active project with lot of eventual contributors

Release Soon Release Often (every 6 weeks)

The Debian Case



Testsuite

- Follows the RDD pattern, late for TDD, continuous refactorings
- Runs on Linux and macOS in Travis and Jenkins (slow for AppVeyour)
- Fuzzing is part of the development process
- Using valgrind, asan, clang-analyzer, scan.coverity

Not Written in Python

C is not the perfect language, it's easy to make mistakes, but Python is not the solution. Maybe Rust fits better with the philosophy of the project.

- 3 bindings for Python Swig, r2pipe, CTypes
- Support MIASM, IO, RBin and RAsm plugins

- Statically typed languages catch errors at compile time
- More tools available to profile, debug, optimize
- Faster, native and smaller footprint, transpiles to js!
- Enough for 90% of the problems faced in r2land


There is no GUI.

Terminals are scary

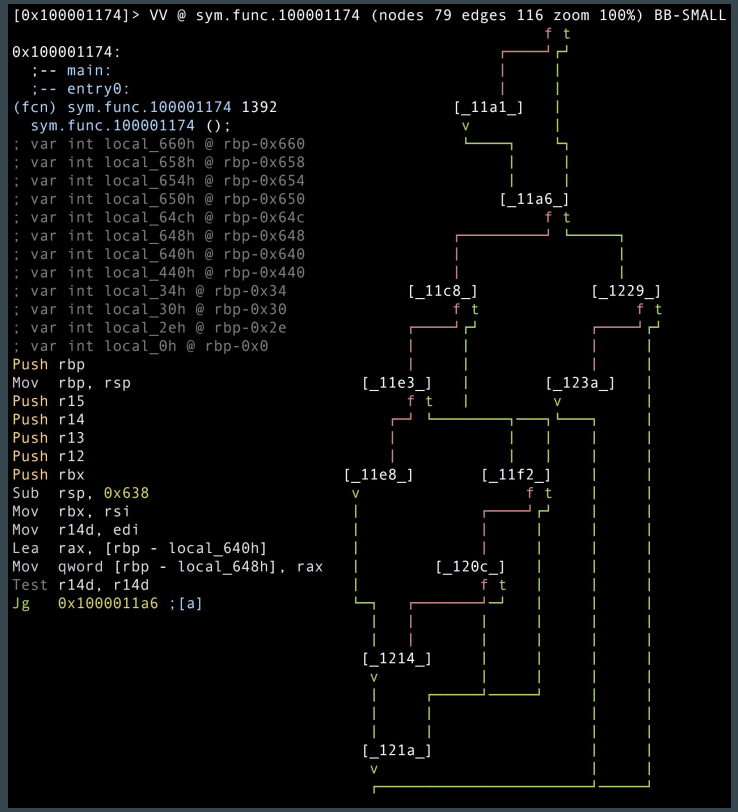
Writing UIs is boring and commandline tools are faster to develop and more flexible. But lazy minds usually like to wheel and click around instead of typing commands.

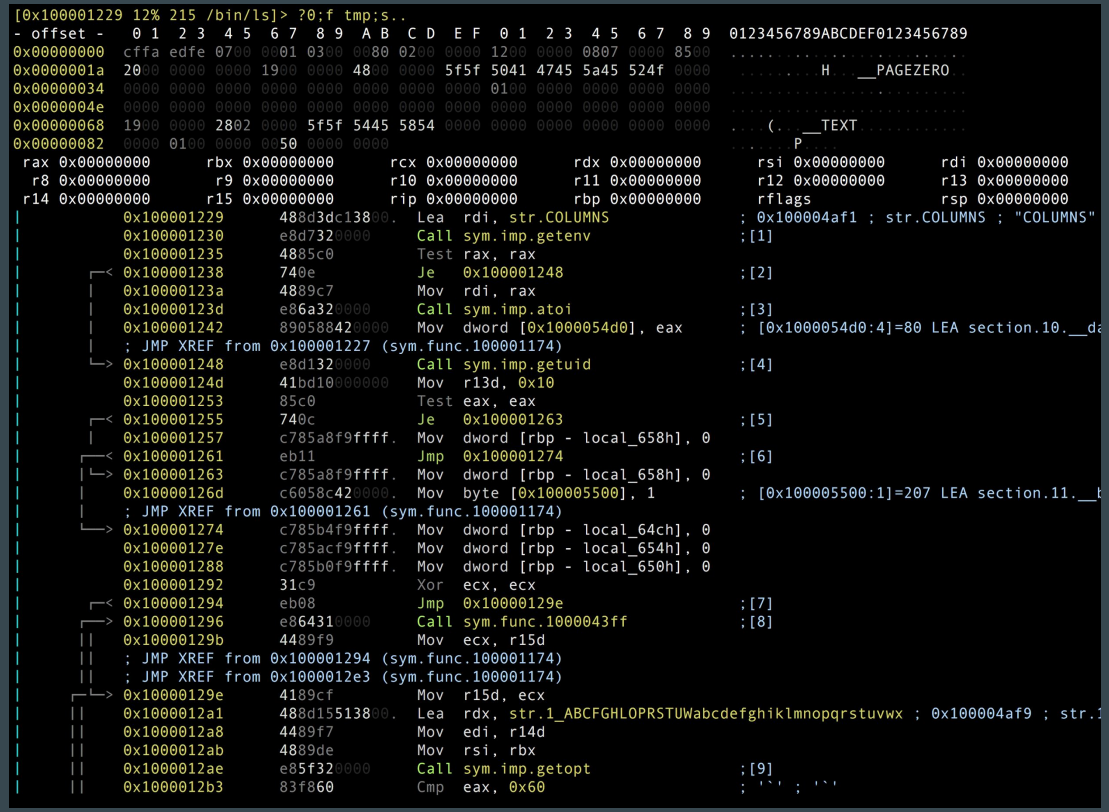
We care about users, but r2 is not a GUI, other projects fill the gap.

Only RE tool of choice for blind people (we have at least 2 users!), text-to-speech and braile device support works fine with r2.

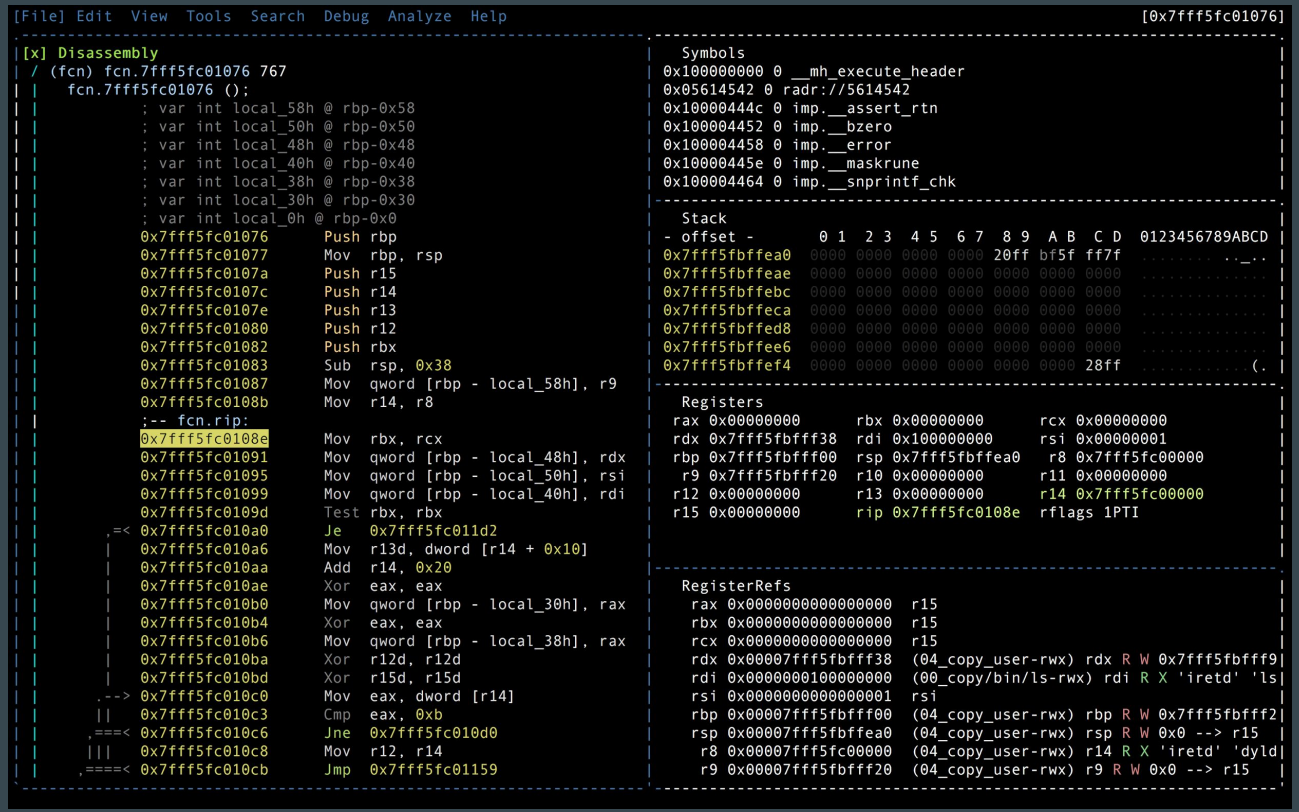
- The problem is not the lack of GUI, but the amount of them.

Visual Mode (V)

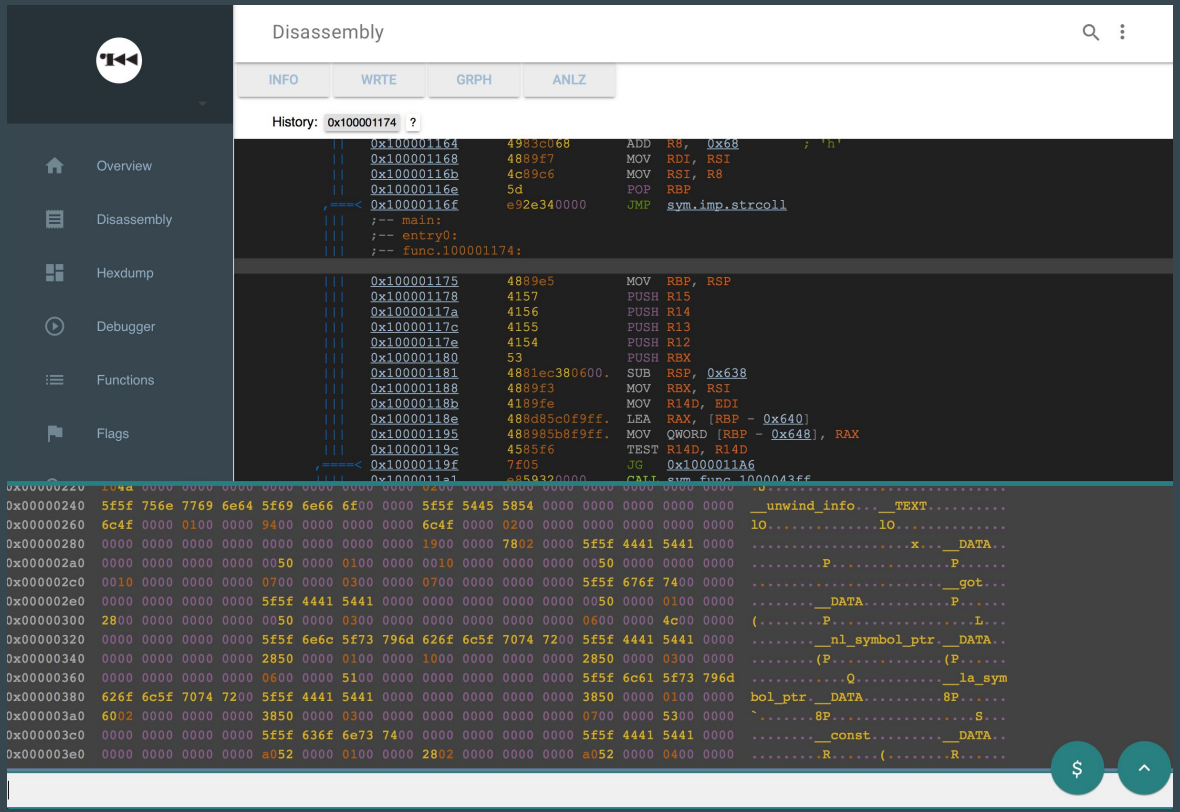




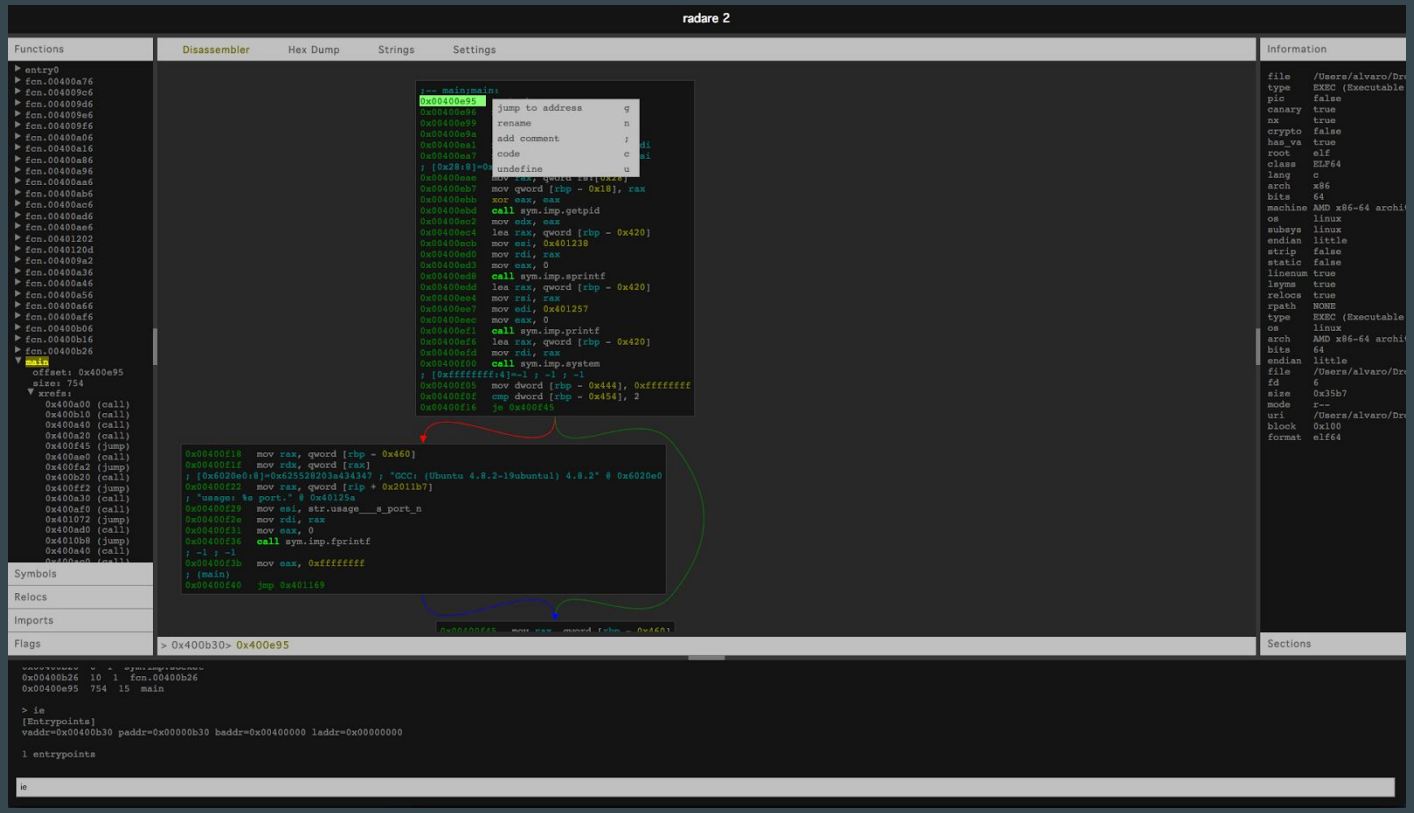
Tiled Visual (V!)



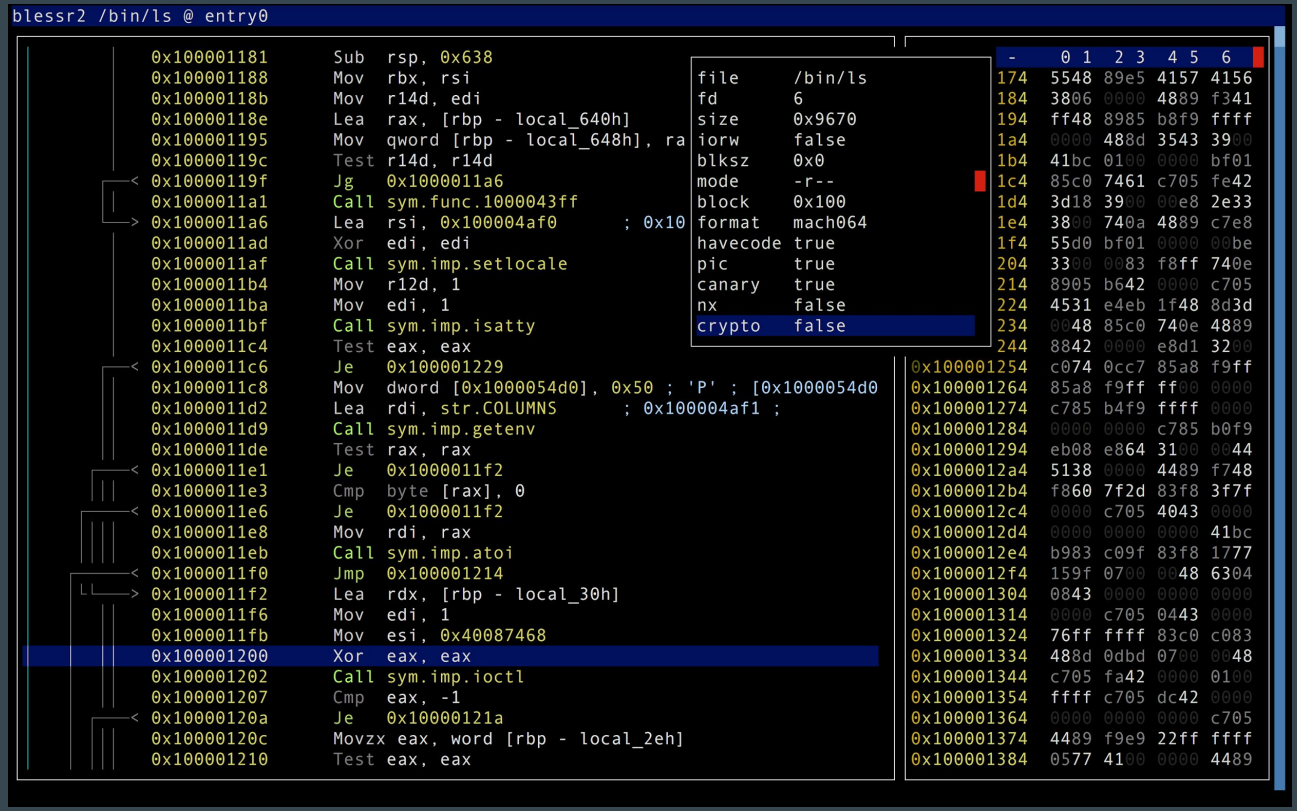
WebUI (=H,/m)


Disassembly
INFO
WRTE
GRPH
ANLZ
History: 0x100001174 ?
Overview
Disassembly
Hexdump
Debugger
Functions
Flags
0x100001164 4983c068 ADD R8, 0x68 ; 'h'
0x100001168 4889f7 MOV RDI, RSI
0x10000116b 4c89c6 MOV RSI, R8
0x10000116e 5d POP RBP
0x10000116f e92e340000 JMP sym.imp.strcoll
;-- main:
;-- entry0:
;-- func.100001174:
0x100001175 4889e5 MOV RBP, RSP
0x100001178 4157 PUSH R15
0x10000117a 4156 PUSH R14
0x10000117c 4155 PUSH R13
0x10000117e 4154 PUSH R12
0x100001180 53 PUSH RBX
0x100001181 4881ec380600. SUB RSP, 0x638
0x100001188 4889f3 MOV RBX, RSI
0x10000118b 4189fe MOV R14D, EDI
0x10000118e 488d85c0f9ff. LEA RAX, [RBP - 0x640]
0x100001195 488985b8f9ff. MOV QWORD [RBP - 0x648], RAX
0x10000119c 4585f6 TEST R14D, R14D
0x10000119f 7f05 JG 0x1000011A6
0x00000240 5f5f 756e 7769 6e64 5f69 6e66 6f00 0000 5f5f 5445 5854 0000 0000 0000 0000 0000 __unwind_info..__TEXT..........
0x00000260 6c4f 0000 0100 0000 9400 0000 0000 0000 6c4f 0000 0200 0000 0000 0000 0000 0000 lO..............lO..............
0x00000280 0000 0000 0000 0000 0000 0000 0000 0000 1900 0000 7802 0000 5f5f 4441 5441 0000 ....x...__DATA..
0x000002a0 0000 0000 0000 0000 0050 0000 0100 0000 0010 0000 0000 0000 0000 0050 0000 0000 .....P.............P.....
0x000002c0 0010 0000 0000 0000 0700 0000 0300 0000 0700 0000 0000 0000 5f5f 676f 7400 0000 ....__got...
0x000002e0 0000 0000 0000 0000 5f5f 4441 5441 0000 0000 0000 0000 0000 0050 0000 0100 0000 ....__DATA..........P.....
0x00000300 2800 0000 0000 0000 0050 0000 0300 0000 0000 0000 0000 0000 0600 0000 4c00 0000 (.........P..............L...
0x00000320 0000 0000 0000 0000 5f5f 6e6c 5f73 796d 626f 6c5f 7074 7200 5f5f 4441 5441 0000 ....__nl_symbol_ptr.__DATA..
0x00000340 0000 0000 0000 0000 2850 0000 0100 0000 1000 0000 0000 0000 2850 0000 0300 0000 ....(P..............(P......
0x00000360 0000 0000 0000 0000 0600 0000 5100 0000 0000 0000 0000 0000 5f5f 6c61 5f73 796d ....Q...........__la_sym
0x00000380 626f 6c5f 7074 7200 5f5f 4441 5441 0000 0000 0000 0000 0000 3850 0000 0100 0000 bol_ptr.__DATA..........8P......
0x000003a0 6002 0000 0000 0000 3850 0000 0300 0000 0000 0000 0000 0000 0700 0000 5300 0000 `.......8P..................S...
0x000003c0 0000 0000 0000 0000 5f5f 636f 6e73 7400 0000 0000 0000 0000 5f5f 4441 5441 0000 ....__const..........__DATA..
0x000003e0 0000 0000 0000 0000 a052 0000 0100 0000 2802 0000 0000 0000 a052 0000 0400 0000 ....R......(.........R.....
$

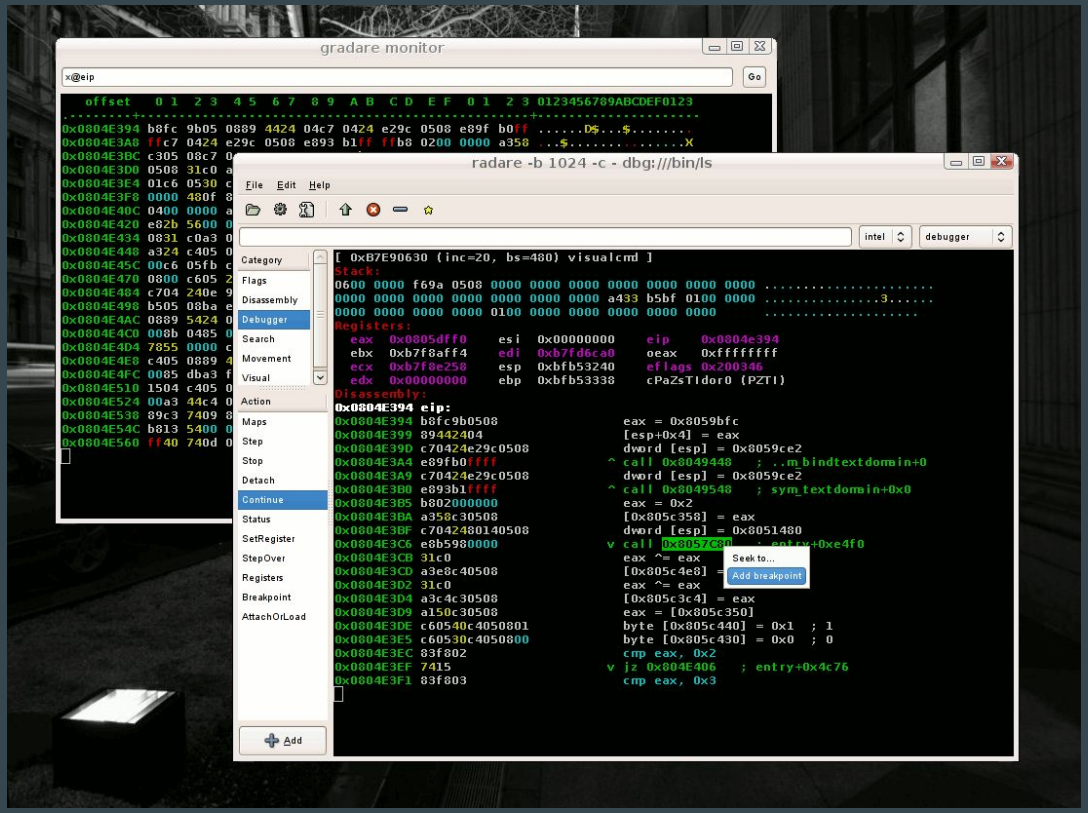
WebUI (=H,/p)


radare 2
Functions
Disassembler
Hex Dump
Strings
Settings
Information
Symbols
Relocs
Imports
Flags
Sections
jump to address g
rename n
add comment ;
code c
undefine u
> 0x400b30> 0x400e95
> ie
[Entrypoints]
vaddr=0x00400b30 paddr=0x00000b30 baddr=0x00400000 laddr=0x00000000
1 entrypoints
ie

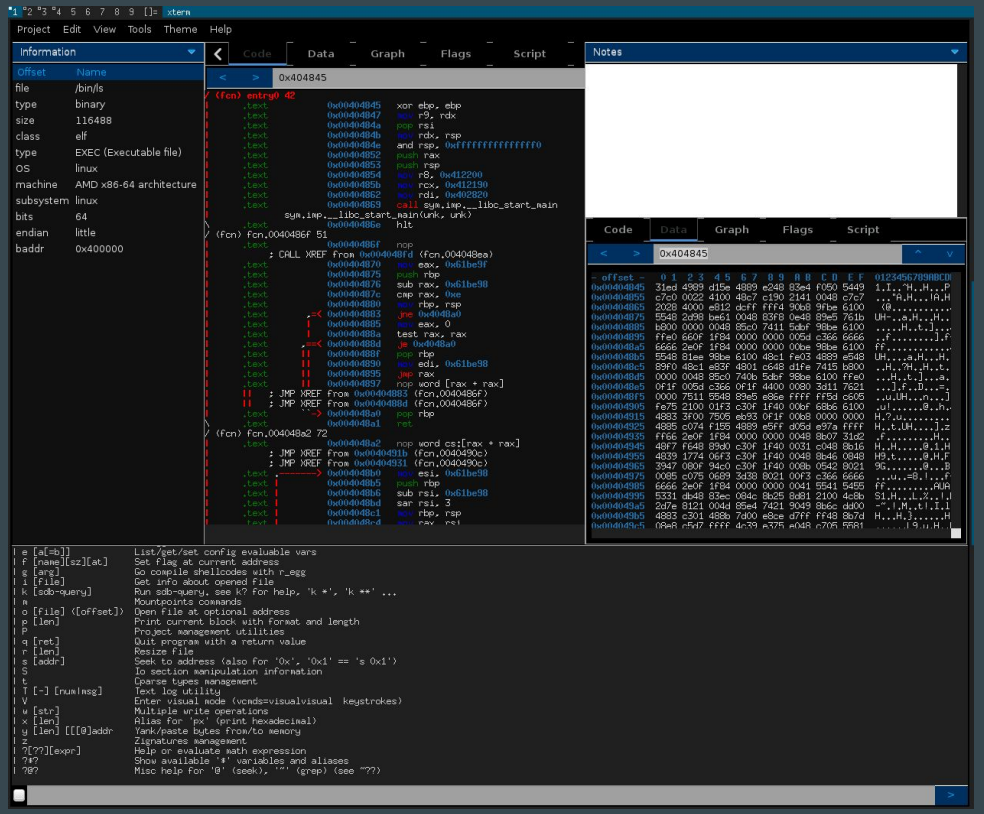
BlessR2 (Node+Blessed)

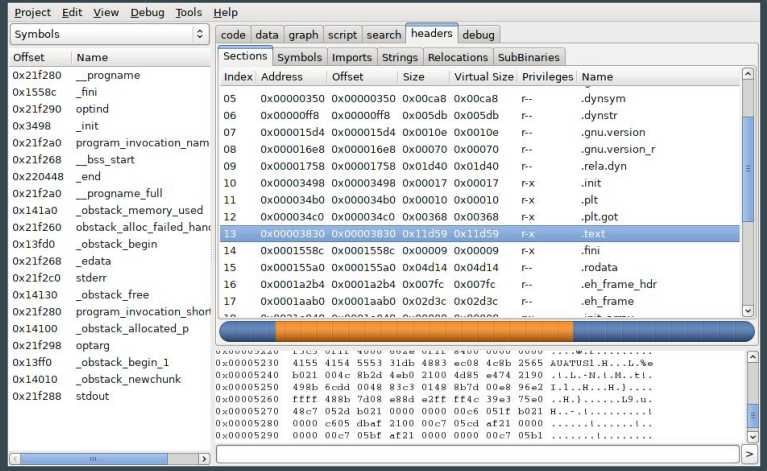


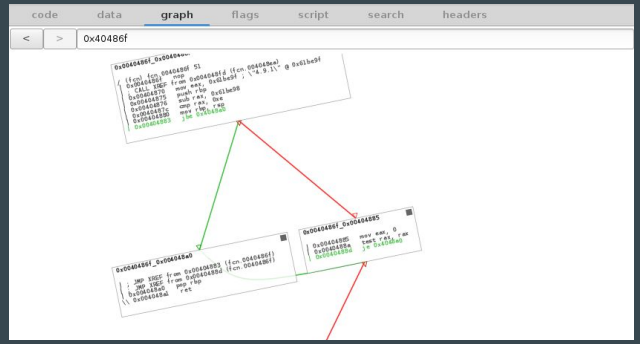
Gradare (Gtk2+Vte)



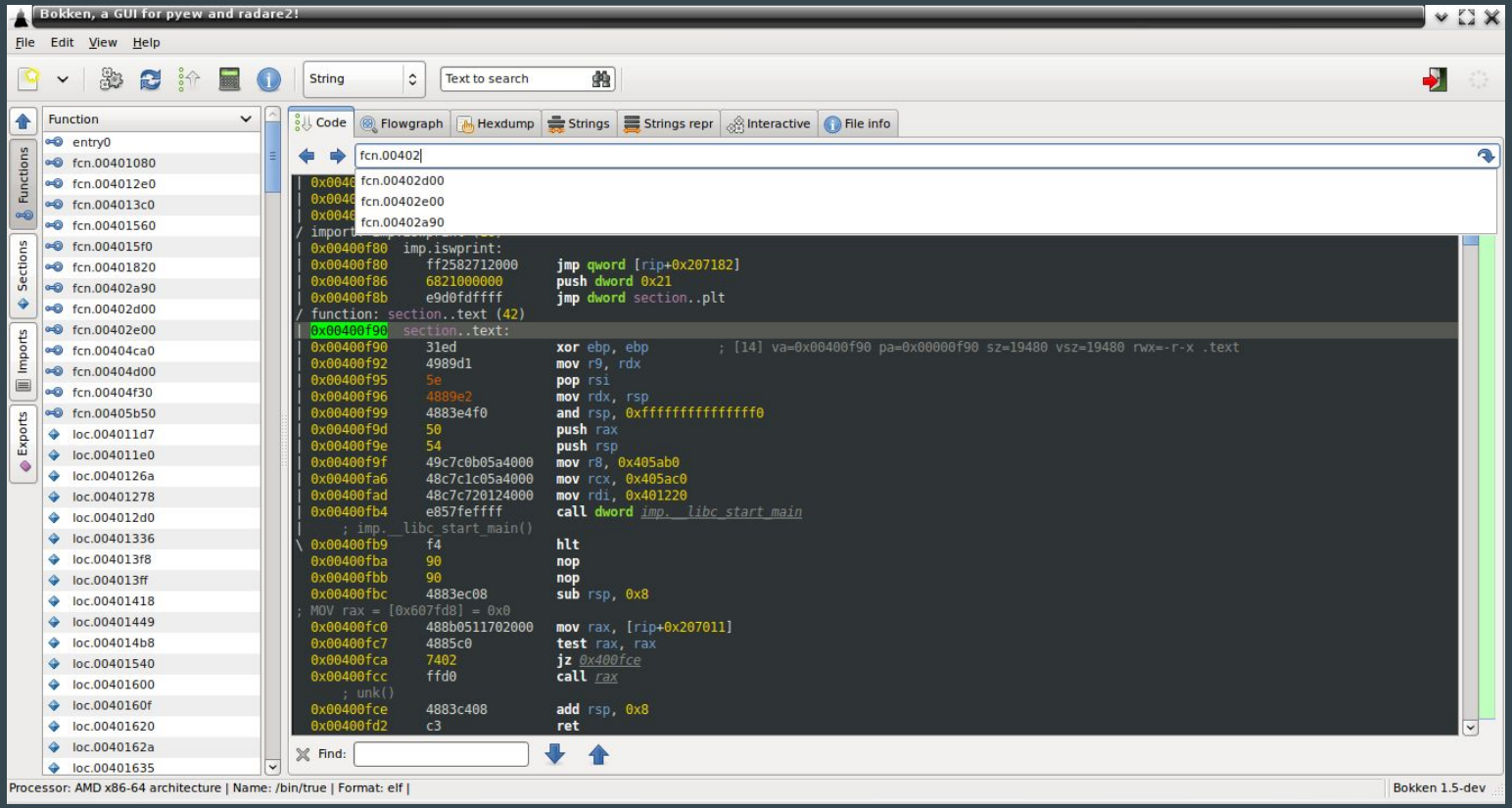
Ragui (abandoned/unreleased)



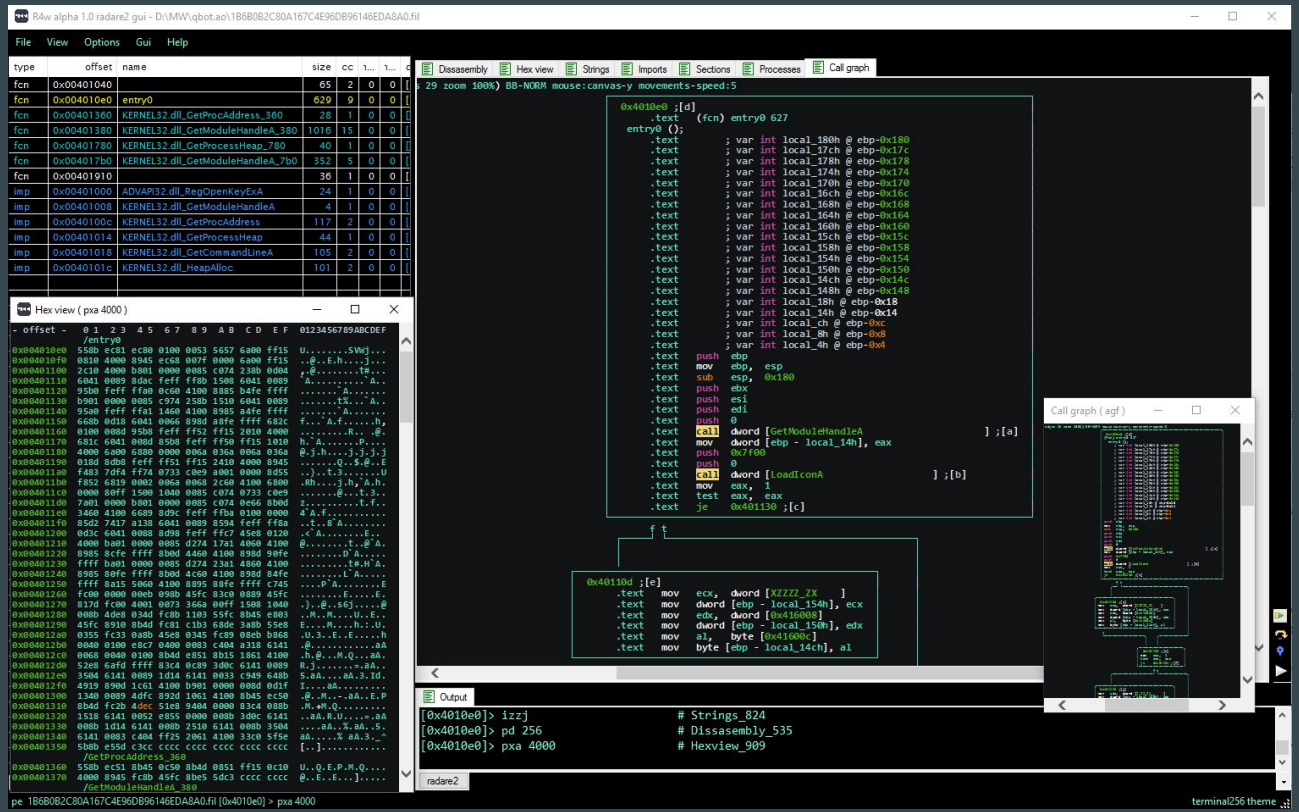




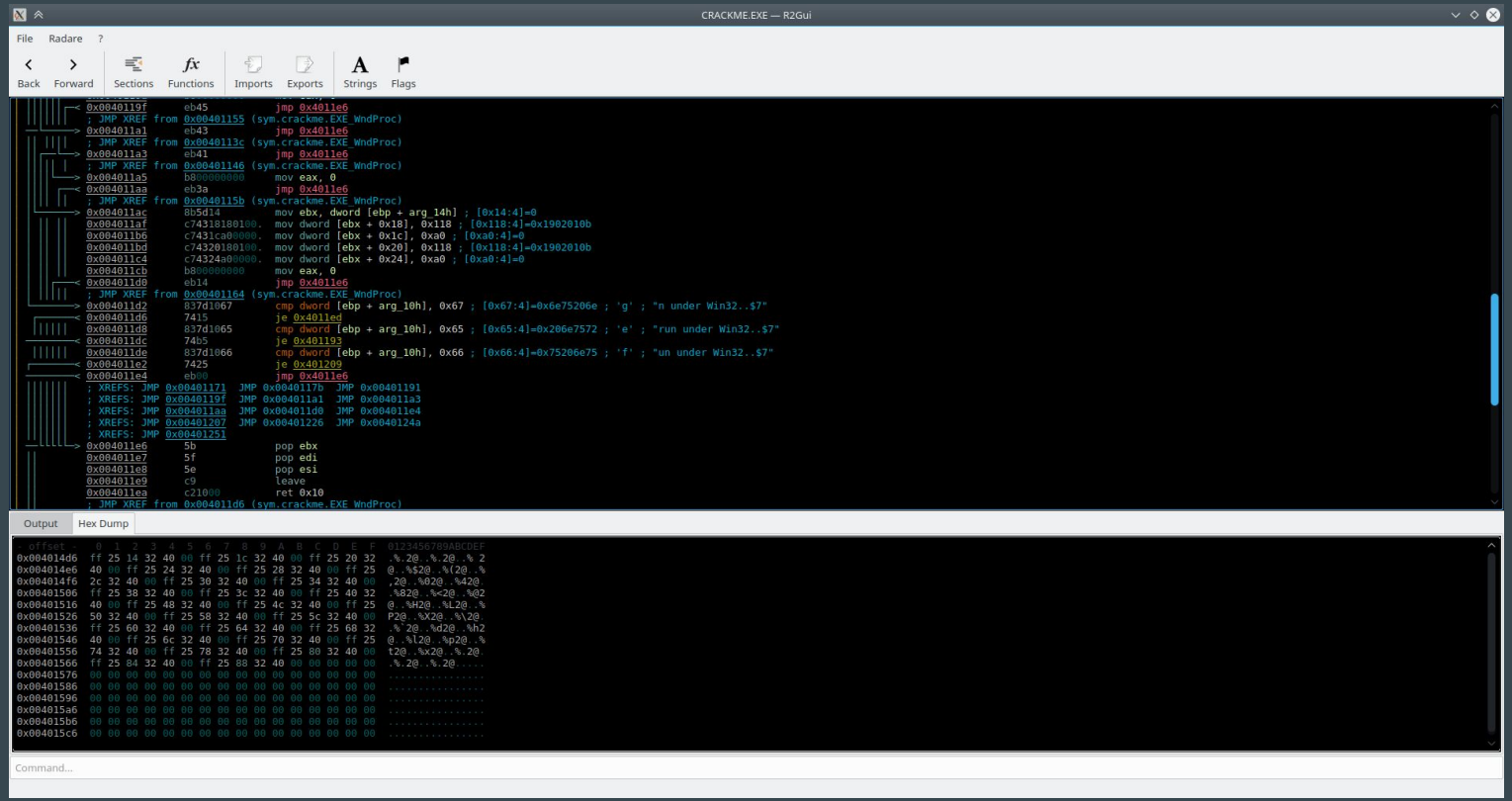
Bokken (Py/Gtk2)



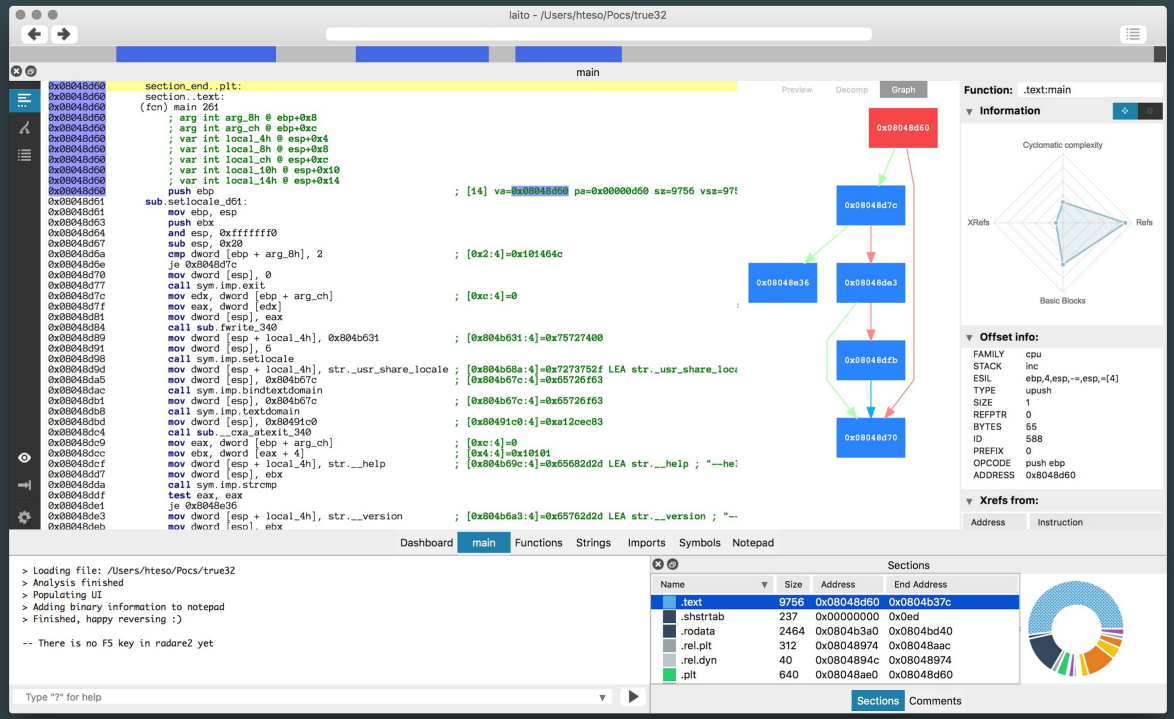
R2G4W (.NET/MFC)



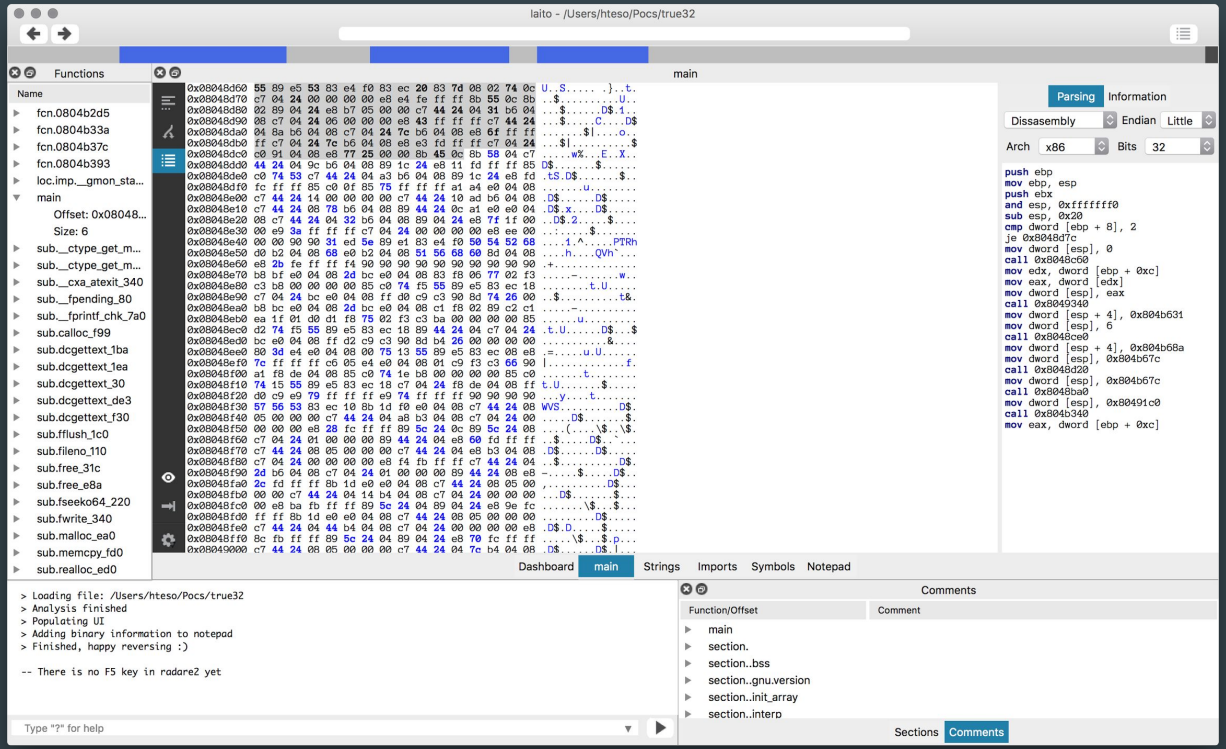
R2GUI (QT5/C++) (3 days ago)



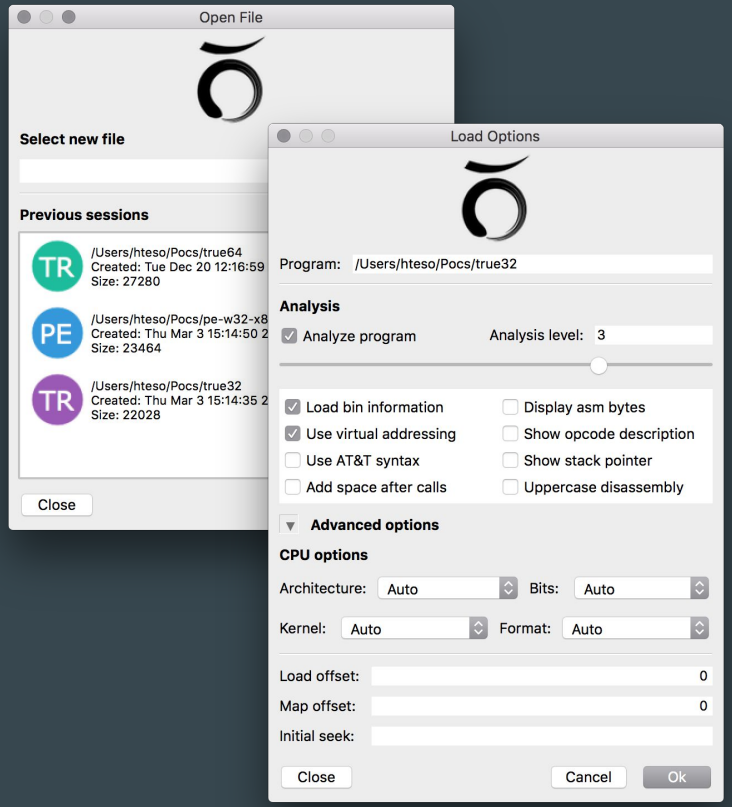
laito (Qt/C++) (alpha release on early 2017)

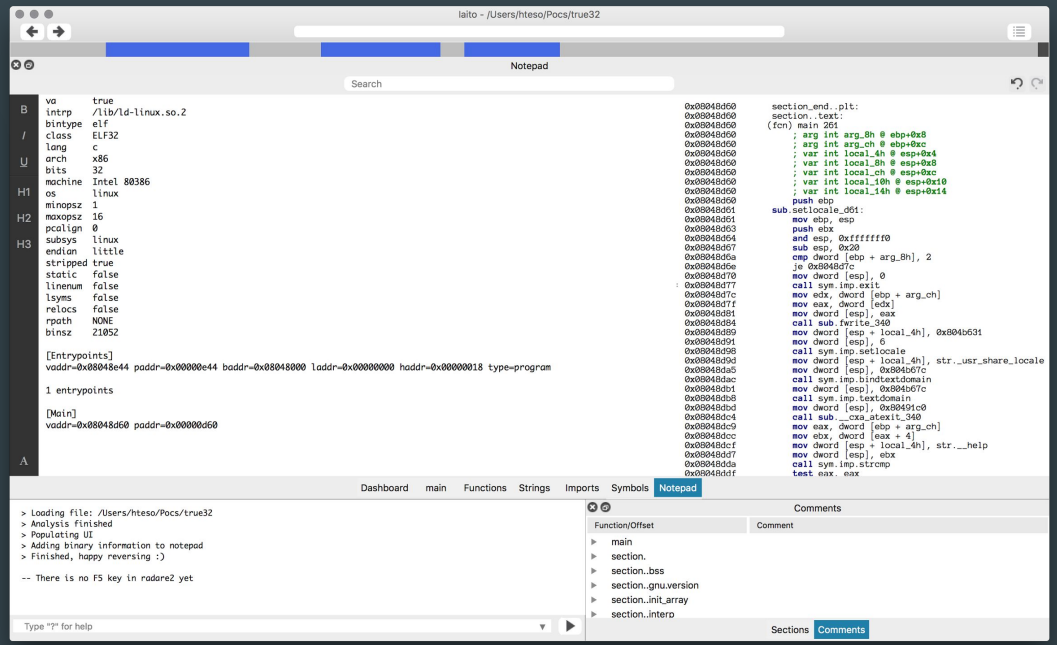


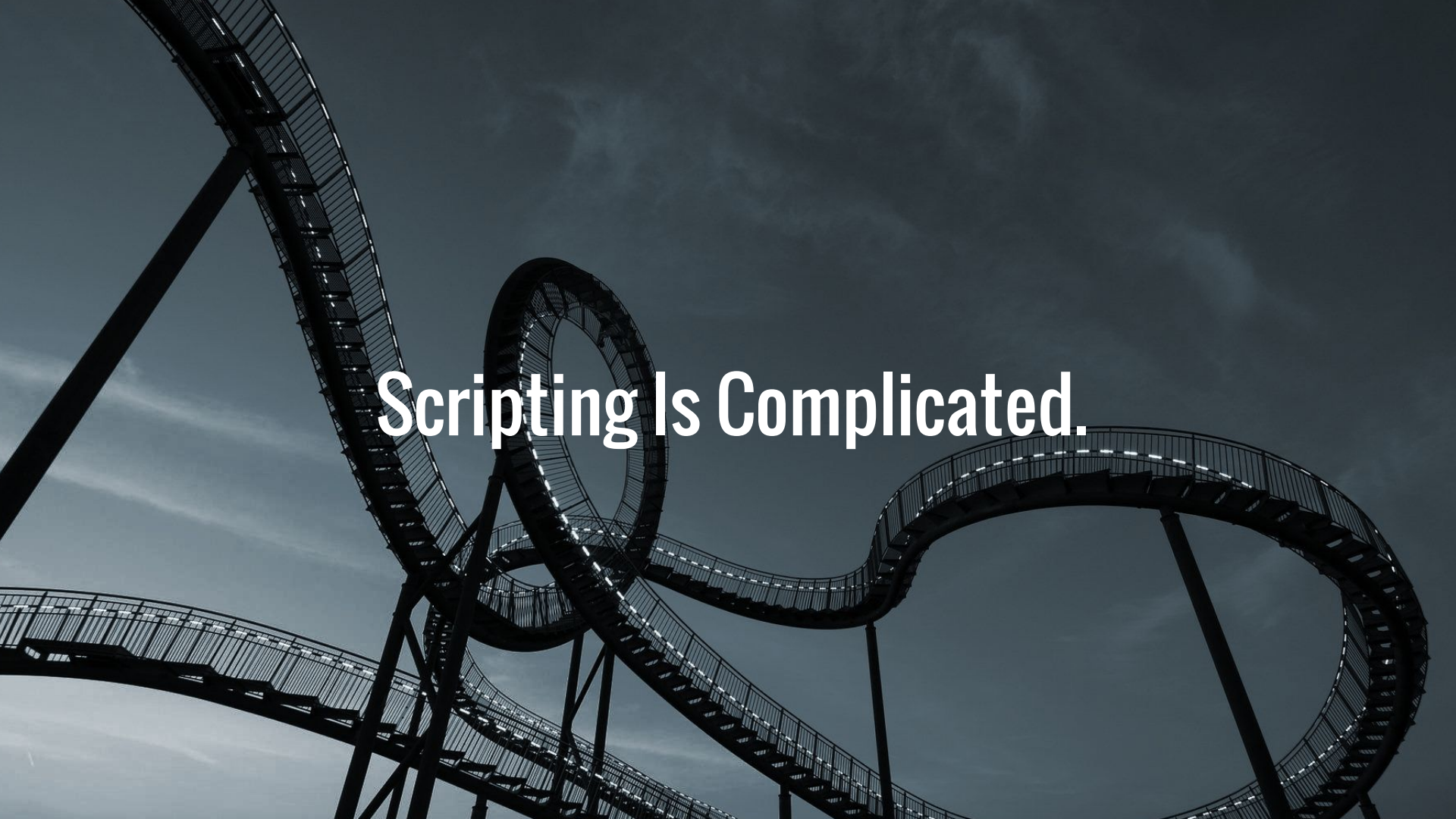
laito (Qt/C++)


laito - /Users/hteso/Pocs/true32
Functions
Name
fcn.0804b2d5
fcn.0804b33a
fcn.0804b37c
fcn.0804b393
loc.imp.__gmon_sta...
main
Offset: 0x08048...
Size: 6
sub.__ctype_get_m...
sub.__ctype_get_m...
sub.__cxa_atexit_340
sub.__fpending_80
sub.__fprintf_chk_7a0
sub.calloc_f99
sub.dcgettext_1ba
sub.dcgettext_1ea
sub.dcgettext_30
sub.dcgettext_de3
sub.dcgettext_f30
sub.fflush_1c0
sub.fileno_110
sub.free_31c
sub.free_e8a
sub.fseeko64_220
sub.fwrite_340
sub.malloc_ea0
sub.memcpy_fd0
sub.realloc_ed0
main
Parsing
Information
Dissasembly
Endian
Little
Arch
x86
Bits
32
push ebp
mov ebp, esp
push ebx
and esp, 0xfffffff0
sub esp, 0x20
cmp dword [ebp + 8], 2
je 0x8048d7c
mov dword [esp], 0
call 0x8048c50
mov edx, dword [ebp + 0xc]
mov eax, dword [edx]
mov dword [esp], eax
call 0x8049340
mov dword [esp + 4], 0x804b631
mov dword [esp], 6
call 0x8048ce0
mov dword [esp + 4], 0x804b68a
mov dword [esp], 0x804b67c
call 0x8048d20
mov dword [esp], 0x804b67c
call 0x8048ba0
mov dword [esp], 0x80491c0
call 0x804b340
mov eax, dword [ebp + 0xc]
Dashboard
main
Strings
Imports
Symbols
Notepad
> Loading file: /Users/hteso/Pocs/true32
> Analysis finished
> Populating UI
> Adding binary information to notepad
> Finished, happy reversing :)
-- There is no F5 key in radare2 yet
Type "?" for help
Comments
Function/Offset
Comment
main
section.
section..bss
section..gnu.version
section..init_array
section..interp
Sections
Comments

laito (Qt/C++)


Open File
Select new file
Previous sessions
/Users/hteso/Pocs/true64
Created: Tue Dec 20 12:16:59
Size: 27280
/Users/hteso/Pocs/pe-w32-x8
Created: Thu Mar 3 15:14:50 2
Size: 23464
/Users/hteso/Pocs/true32
Created: Thu Mar 3 15:14:35 2
Size: 22028
Close
Load Options
Program: /Users/hteso/Pocs/true32
Analysis
Analyze program
Analysis level: 3
Load bin information
Use virtual addressing
Use AT&T syntax
Add space after calls
Display asm bytes
Show opcode description
Show stack pointer
Uppercase disassembly
Advanced options
CPU options
Architecture: Auto
Bits: Auto
Kernel: Auto
Format: Auto
Load offset: 0
Map offset: 0
Initial seek:
Close
Cancel
Ok


laito - /Users/hteso/Pocs/true32
Notepad
Search
Dashboard
main
Functions
Strings
Imports
Symbols
Notepad
Comments
Function/Offset
Comment
Sections
Comments


Scripting Is Complicated.

Scripting

Automate actions, create plugins, add new commands or extending functionality can be done in C or in any other programming language using:

- R2 commands, macros, modifiers, repeaters, ...
- RLang internal evaluation of $lang expressions into r2 (libr/lang)
- Native Swig/Valabind Bindings (radare2-bindings)
- R2Pipe (string and json api for RCore.cmd())

r2pipe

APIs around r_core_cmd_str()

- open()
- cmd()
- cmdj()
- quit()

- Write Plugins for (io/asm/bin)
- JSON deserialization
- Sync / Async

- Support A LOT of languages
  - r2pm cd radare2-r2pipe
- Many connection methods
  - Native/RAP/HTTP/PIPE/..

List of Supported Languages

- C / C++
- Vala
- C# / F#
- Nim
- DLang
- Swift
- Java
- Go
- Haskell
- Python
- NodeJS
- Ruby
- Perl
- PHP
- Erlang
- OCaml
- Lisp / NewLisp
- Clojure

r2pipe

(demo)

Mirai Malware Config Decryption


Debugger Is Confusing.

Debugger Is Confusing

- Starts debugging at dyld (not the program entrypoint)
- Not aiming to replace a source debugger (but supports dwarf/pdb/..)
- Programs can have multiple slices or entrypoints (rabin2 -x)
- Changes in memory doesn’t apply to disk
- Rarun2 profiles needed sometimes

Debugger Basics

- Spawn or Attach
- Pluggable for local and remote
  - native/gdb/windbg/bochs/...
- Subcommands of ‘d’
- Telescoping
  - dr= / drr
  - pxr @ rsp
- Remoting via rap:// and =!=
- Inject code with dx
- Dump/Restore reg/mem states
- Memory
  - read/write/pages/perms
- Registers
  - families/get/set/flags
- Processes
  - children/tls
- Descriptors
  - sockets/files/windows
- Breakpoints
  - sw/hw/mmu

Debugger Backends.

As long as everything in r2land is pluggable, debuggers are also considered modular parts and there are many implementations for them, you can write your own!

In Core:

- Bochs
- WinDBG
- GDB
- QNX
- ESIL

Via r2pm:

- R2frida
- R2lldb

GDB://

Gdb client stub implemented from scratch, to be used with QEMU, VMWare, gdbserver, ...

- GDB protocol is crap
- Mixes binary, plaintext and XML with ascii checksums \o/
- Each platform (arch/os pair) requires changes
- X86/X64 support is there

- WIP to properly support MIPS, ARM, ARM64 and AVR

R2LLDB

Available via r2pm, uses the LLDB python API to talk to r2 via r2pipe with RAP.

- Allows to use a running LLDB session from r2
- Works on all Apple things (watchOS, iOS, ...) without jb
- Also works for XNU kernel debugging

Easily portable to GDB-Python (not yet done)

R2Frida

Use Frida as a backend for memory access and in-process code injection.

There are other plugins like r2lldb, bochs, gdb.. that are also interesting..

- Attach to local or remote process
- Supports macOS, iOS, Linux, Android, QNX, Windows
- Javascript code injection and hooking
- Apis and commands to resolve classes, methods, etc

R2Frida

(demo)


WTH IS ESIL.

ESIL

- Stands for ‘Evaluable Strings Intermediate Language’
- Standard intermediate language in r2
- Reuses text-based register profiles from analysis or debugger
- Forth-like Language (2 stacks)
- Each instruction is translated to a single string

Mov Eax, 33 => 33,eax,=

- Used for emulation, assisted debugging
- Search expressions, Predict jumps, Find references

ESIL

- ae subcommands used to manipulate the virtual machine of ESIL
  - aeim - initialize host stack
  - aer - registers
  - aesu - step until
- /E search offsets that match an ESIL expression
- e asm.emu / likely branches
- aae - emulate code to find computed references to strings

- Unicorn support available in r2pm, but not as complete as ESIL

ESIL

(demo)


Exploiting.

Exploiting

Provides all the tools needed for researching vulns and developing exploits.

- Hexadecimal Editor, Assembler, Disassembler
- Analyzer, Bindiffer, Search Code/String/Data
- Debugger, Emulator, Stack Analysis (pxr)
- Other Facilities for Exploiting
  - ROP Gadget Search / Classification (rarop WUI)
  - DeBruijn Patterns Generate / Find Offset (wop)
  - Register/Stack Telescoping (drr)
  - Heap Analysis (dmh)

DirtyCow

The exploit for CVE-2016-5195 can be easily integrated into r2 as an IO plugin.

This vulnerability can be used to modify the contents of system files without root privileges (Linux 2007 (>= 2.6.22) until 2016 (< 4.8.3)).

(demo)

https://dirtycow.ninja/

https://www.nowsecure.com/blog/2016/12/08/android-dirty-cow-patch/


Questions.