

Radare Demystified



r2@33C3/2016

pancake@nopcode.org

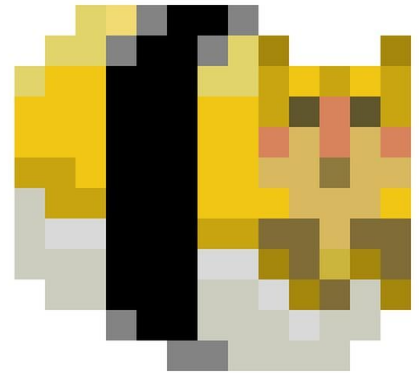
A black and white photograph of a space shuttle launching. The shuttle is positioned in the upper center, ascending vertically. It is surrounded by a massive, billowing cloud of white smoke and steam that fills most of the frame. In the foreground, the dark, flat roof of a large building is visible, with several small figures of people standing on it. A tall, thin tower with a cluster of lights at the top stands near the center. The overall scene is dramatic and captures the power of a rocket launch.

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

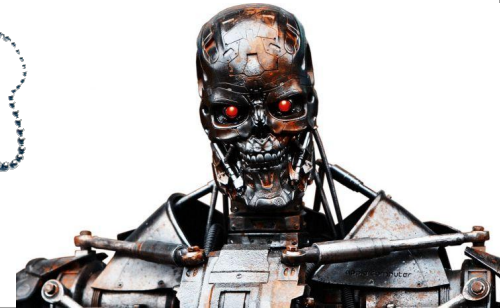
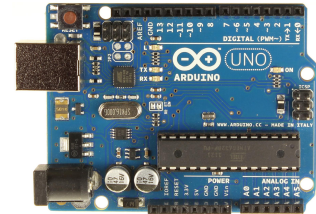
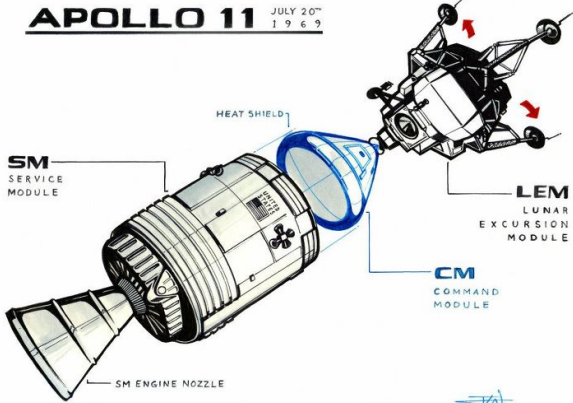
- Linux/Android
- macOS/iOS
- Windows
- QNX

```
$ rasm2 -L
```

```
$ rabin2 -L
```

```
$ r2pm -s
```

What can I inspect?



Wait..

Don't lose the rail

A perspective view of a long, narrow tunnel. The walls are made of rough, textured concrete and are heavily covered in black graffiti. The floor is a dark, gridded metal walkway. A series of square, grid-patterned lights are mounted on the ceiling, receding into the distance. The overall atmosphere is dark and mysterious.

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

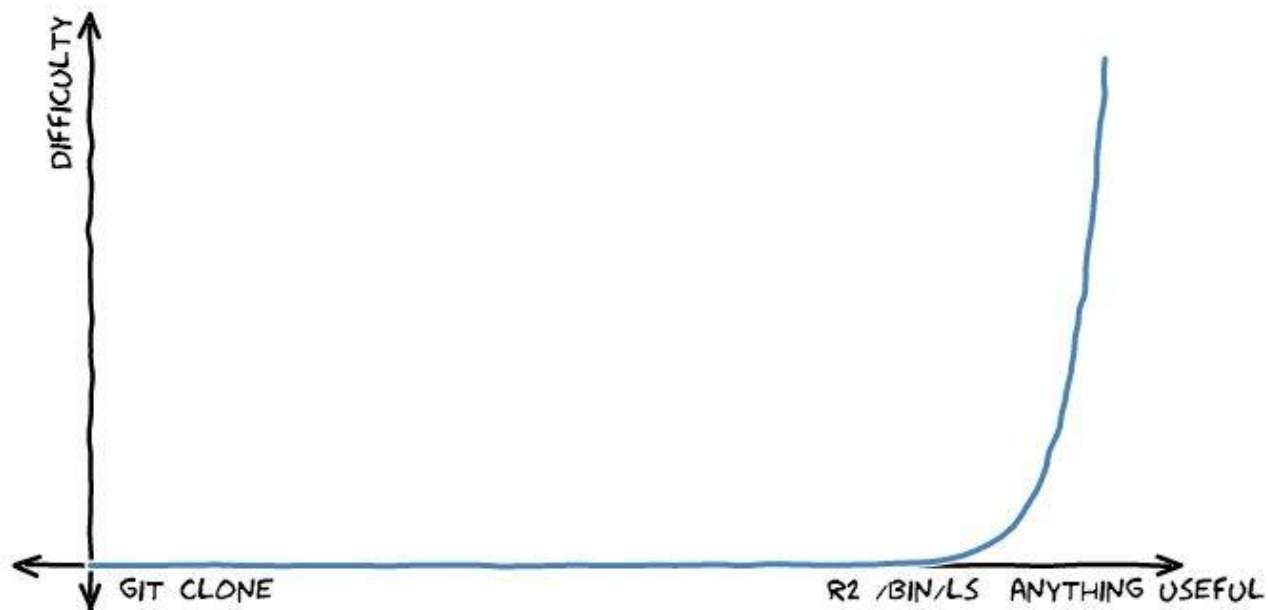
But First.. Let's Make a Poll

- How many of you know radare?
- How many of you use r2?

A black and white photograph of a person in winter clothing using a pickaxe on a steep, rocky slope. The person is positioned in the center-left of the frame, leaning forward and striking the rock. The background shows a vast, open landscape under a cloudy sky. The overall tone is somber and emphasizes the difficulty of the task.

It's Difficult.

R2 LEARNING CURVE



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

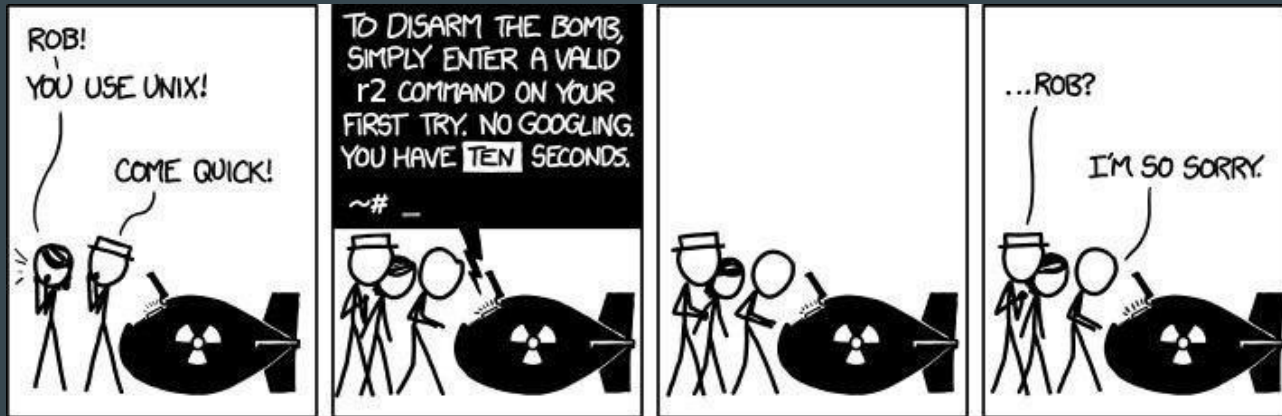
In the other hand..

- 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=

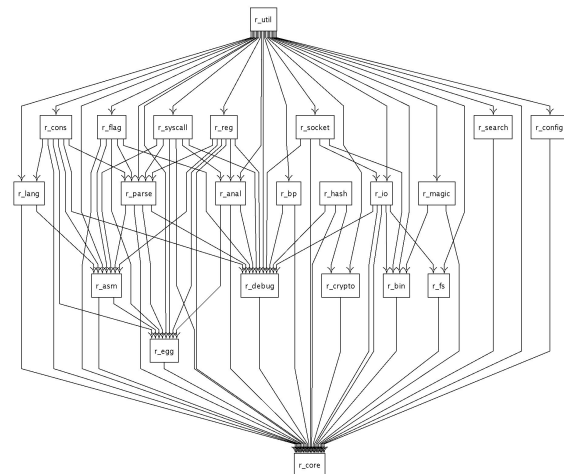
Hard To Remember

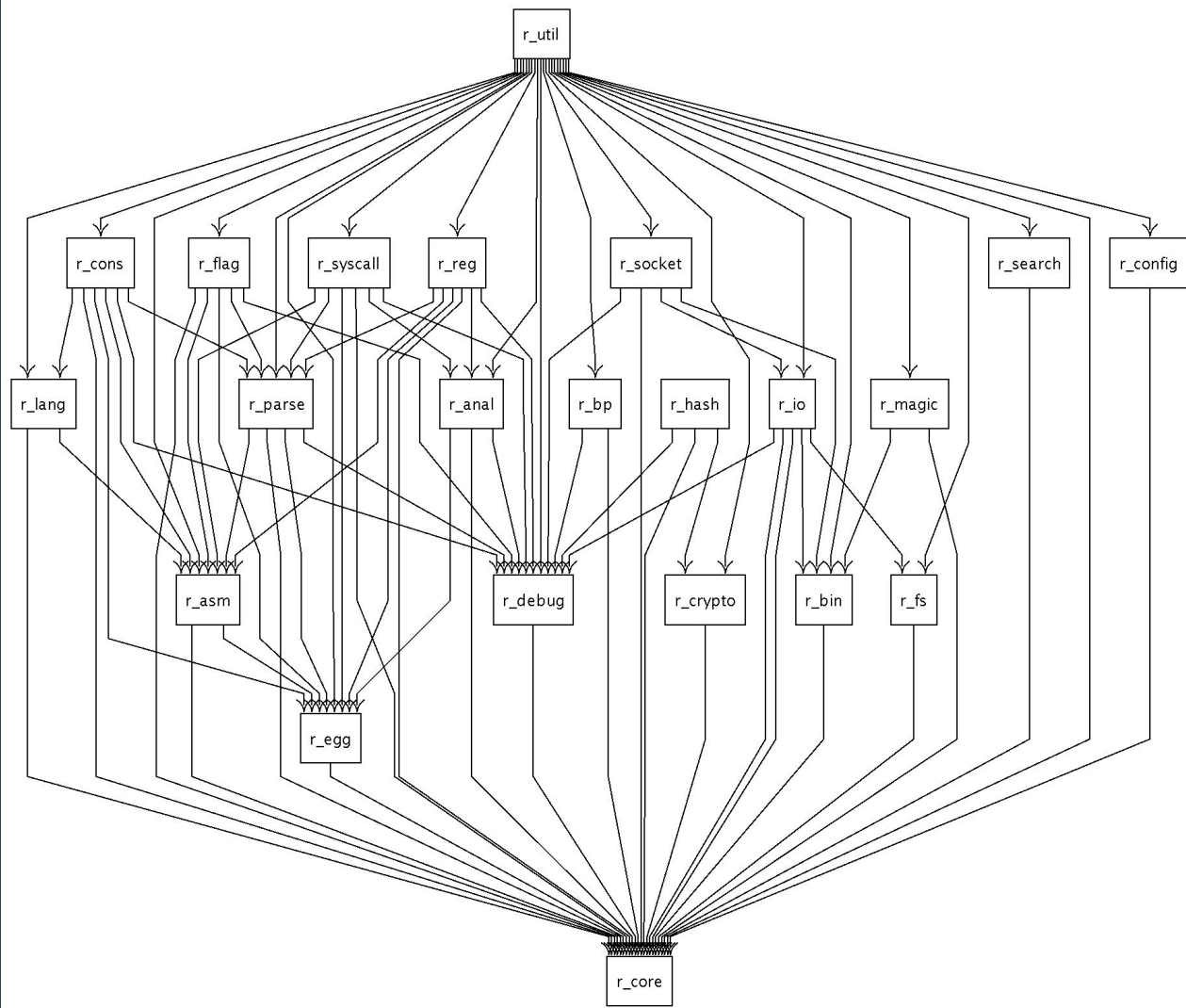
Commands follow simple mnemonic rules:

- 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





A close-up, high-contrast black and white photograph of a fingerprint on a dark, textured surface. The fingerprint is the central focus, showing clear ridge patterns. The background is a dark, grainy material with some vertical streaks. Overlaid on the fingerprint is the text "Useless For Forensics." in a bold, white, sans-serif font.

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.

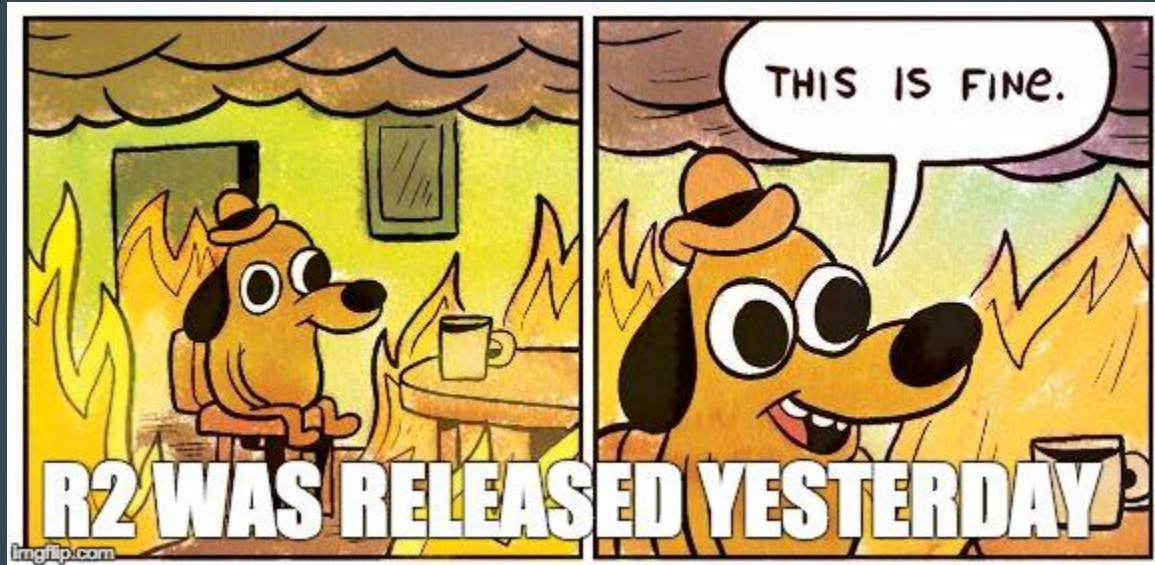
So...

- 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

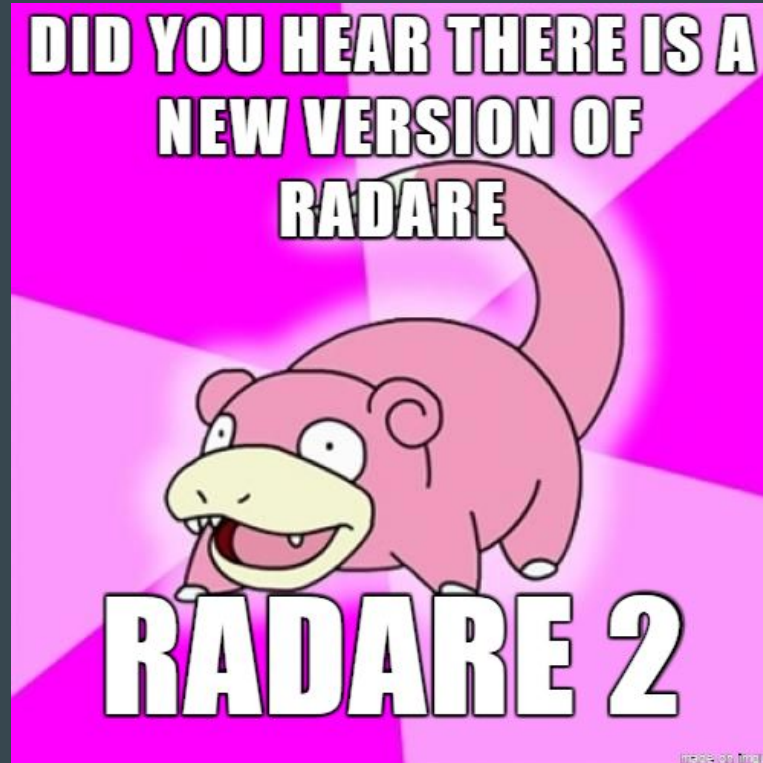
Feature X is Broken

- 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 MASM, 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.

Tiled Visual (V!)

```
[File] Edit View Tools Search Debug Analyze Help [0x7fff5fc01076]
[x] Disassembly
/ (fcn) fcn.7fff5fc01076 767
  fcn.7fff5fc01076 ();
  ; var int local_58h @ rbp-0x58
  ; var int local_50h @ rbp-0x50
  ; var int local_48h @ rbp-0x48
  ; var int local_40h @ rbp-0x40
  ; var int local_38h @ rbp-0x38
  ; var int local_30h @ rbp-0x30
  ; var int local_0h @ rbp-0x0
0x7fff5fc01076  Push rbp
0x7fff5fc01077  Mov rbp, rsp
0x7fff5fc0107a  Push r15
0x7fff5fc0107c  Push r14
0x7fff5fc0107e  Push r13
0x7fff5fc01080  Push r12
0x7fff5fc01082  Push rbx
0x7fff5fc01083  Sub rsp, 0x38
0x7fff5fc01087  Mov qword [rbp - local_58h], r9
0x7fff5fc0108b  Mov r14, r8
;-- fcn.rip:
0x7fff5fc0108e  Mov rbx, rcx
0x7fff5fc01091  Mov qword [rbp - local_48h], rdx
0x7fff5fc01095  Mov qword [rbp - local_50h], rsi
0x7fff5fc01099  Mov qword [rbp - local_40h], rdi
0x7fff5fc0109d  Test rbx, rbx
,=< 0x7fff5fc010a0  Je 0x7fff5fc011d2
0x7fff5fc010a6  Mov r13d, dword [r14 + 0x10]
| 0x7fff5fc010aa  Add r14, 0x20
| 0x7fff5fc010ae  Xor eax, eax
| 0x7fff5fc010b0  Mov qword [rbp - local_30h], rax
| 0x7fff5fc010b4  Xor eax, eax
| 0x7fff5fc010b6  Mov qword [rbp - local_38h], rax
| 0x7fff5fc010ba  Xor r12d, r12d
| 0x7fff5fc010bd  Xor r15d, r15d
--> 0x7fff5fc010c0  Mov eax, dword [r14]
|| 0x7fff5fc010c3  Cmp eax, 0xb
,==< 0x7fff5fc010c6  Jne 0x7fff5fc010d0
||| 0x7fff5fc010c8  Mov r12, r14
,====< 0x7fff5fc010cb  Jmp 0x7fff5fc01159

Symbols
0x100000000 0 __mh_execute_header
0x05614542 0 radr://5614542
0x10000444c 0 imp.__assert_rtn
0x100004452 0 imp.__bzero
0x100004458 0 imp.__error
0x10000445e 0 imp.__maskrune
0x100004464 0 imp.__snprintf_chk

Stack
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D 0123456789ABCD
0x7fff5fbffea0 0000 0000 0000 0000 20ff bf5f ff7f 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffea4 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffebc 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffeca 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffed8 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffef6 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x7fff5fbffef4 0000 0000 0000 0000 0000 0000 0000 28ff 0000 0000 0000 0000 0000 0000 0000

Registers
rax 0x00000000 rbx 0x00000000 rcx 0x00000000
rdx 0x7fff5fbfff38 rdi 0x100000000 rsi 0x00000001
rbp 0x7fff5fbfff00 rsp 0x7fff5fbffea0 r8 0x7fff5fc00000
r9 0x7fff5fbfff20 r10 0x00000000 r11 0x00000000
r12 0x00000000 r13 0x00000000 r14 0x7fff5fc00000
r15 0x00000000 rip 0x7fff5fc0108e rflags 1PTI

RegisterRefs
rax 0x0000000000000000 r15
rbx 0x0000000000000000 r15
rcx 0x0000000000000000 r15
rdx 0x00007fff5fbfff38 (04_copy_user-rwx) rdx R W 0x7fff5fbfff9
rdi 0x0000000100000000 (00_copy/bin/ls-rwx) rdi R X 'iretd' 'ls'
rsi 0x0000000000000001 rsi
rbp 0x00007fff5fbfff00 (04_copy_user-rwx) rbp R W 0x7fff5fbfff2
rsp 0x00007fff5fbffea0 (04_copy_user-rwx) rsp R W 0x0 --> r15
r8 0x00007fff5fc00000 (04_copy_user-rwx) r14 R X 'iretd' 'dylld'
r9 0x00007fff5fbfff20 (04_copy_user-rwx) r9 R W 0x0 --> r15
```

WebUI (=H,/m)

The screenshot displays a debugger interface with a dark theme. On the left is a sidebar with navigation icons and labels: Overview, Disassembly, Hexdump, Debugger, Functions, and Flags. The main area is titled 'Disassembly' and features a search icon and a list of tabs: INFO (selected), WRTE, GRPH, and ANLZ. Below the tabs, the 'History' shows the current instruction address: `0x100001174`.

The assembly view shows the following instructions:

```
||| 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  
;--<< 0x1000011a1 889320000 CALL sym.func.1000013ff
```

Below the assembly view is a memory dump showing a hex dump and its ASCII representation. The memory dump starts at address `0x00000220` and includes various symbols like `_unwind_info...`, `LO...`, `...x...DATA...`, `...P...`, `...got...`, `...DATA...P...`, `...L...`, `...nl_symbol_ptr...DATA...`, `...P...`, `...Q...la_sym`, `bol_ptr...DATA...8P...`, `...8P...S...`, `...const...DATA...`, and `R...R...`.

At the bottom right of the debugger window, there are two circular buttons: a dollar sign (\$) and an up arrow (^).

WebUI (=H,/p)

radare 2

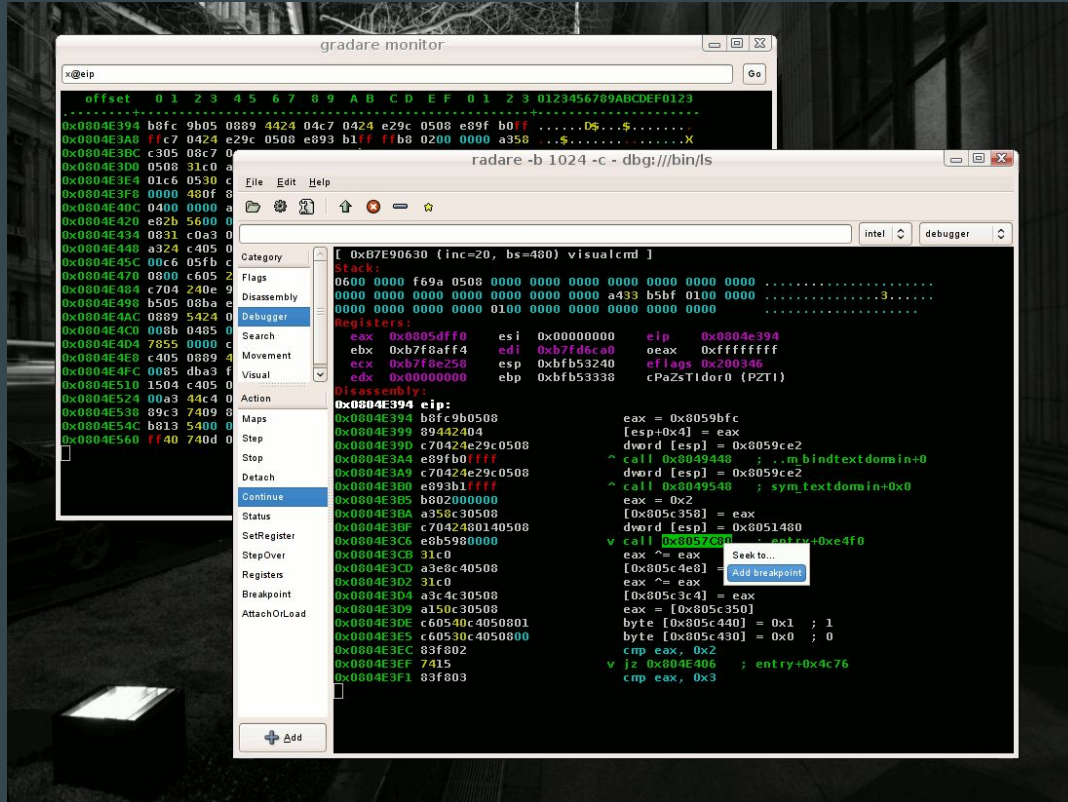
Functions	Disassembler	Hex Dump	Strings	Settings	Information
▶ entry0 ▶ fcn.00400a76 ▶ fcn.004009c6 ▶ fcn.004009d6 ▶ fcn.00400996 ▶ fcn.004009f6 ▶ fcn.00400a06 ▶ fcn.00400a16 ▶ fcn.00400a86 ▶ fcn.00400a96 ▶ fcn.00400aa6 ▶ fcn.00400ab6 ▶ fcn.00400ac6 ▶ fcn.00400ad6 ▶ fcn.00400ae6 ▶ fcn.00401202 ▶ fcn.0040120d ▶ fcn.004009a2 ▶ fcn.00400a36 ▶ fcn.00400a46 ▶ fcn.00400a56 ▶ fcn.00400a66 ▶ fcn.00400a76 ▶ fcn.00400b06 ▶ fcn.00400b16 ▶ fcn.00400b26 ▶ main offset: 0x400e95 ↗ lines 754 ↘ xrefs: 0x400a00 (call) 0x400b10 (call) 0x400a40 (call) 0x400a20 (call) 0x400f45 (jump) 0x400a00 (call) 0x400f42 (jump) 0x400b20 (call) 0x400f42 (jump) 0x400a30 (call) 0x400a09 (call) 0x401072 (jump) 0x400a00 (call) 0x401098 (jump) 0x400a40 (call) 0x400a00 (call)	<pre> -- main:main 0x00400e95 jump to address q 0x00400e96 rename n 0x00400e99 add comment ; dl 0x00400ea1 nops c nl j [0x28f8]; 0x00400e9e mov rax, qword [rip+0x28f8] 0x00400eb7 mov qword [rbp-0x418], rax 0x00400ebb xor eax, eax 0x00400ebd call sym.imp.getpid 0x00400ec2 mov rdx, eax 0x00400ec4 lea rax, qword [rbp-0x420] 0x00400ecb mov esi, 0x401238 0x00400ed0 mov rdi, rax 0x00400ed3 mov eax, 0 0x00400ed6 call sym.imp.printf 0x00400ed8 lea rax, qword [rbp-0x420] 0x00400ee4 mov rsi, rax 0x00400ee7 mov edi, 0x401257 0x00400eeb mov eax, 0 0x00400ef1 call sym.imp.printf 0x00400ef6 lea rax, qword [rbp-0x420] 0x00400fdd mov rdi, rax 0x00400f00 call sym.imp.system ; [0x28ffffef4]-1; -1; -1 0x00400f05 mov dword [rbp-0x444], 0xffffffff 0x00400f0f cmp dword [rbp-0x454], 2 0x00400f16 je 0x400f45 </pre>				file /Users/alvaro/Dr... type EXEC (Executable) pic false canary true nx true crypto false has_va true root elf class ELF64 lang c arch x86 bits 64 machine AMD x86-64 archi... os linux subsys linux endian little strip false static false linenam true lysms true relocs true rpath NONE type EXEC (Executable) os linux arch AMD x86-64 archi... bits 64 endian little file /Users/alvaro/Dr... fd 6 size 0x35b7 mode r-- uri /Users/alvaro/Dr... block 0x100 format elf64
Symbols Relocs Imports Flags	<pre> 0x00400e18 mov rax, qword [rbp-0x460] 0x00400e21 mov rdx, qword [rsi] ; [0x4020e08]-0x62559393a434347; "GCC (Ubuntu 4.8.2-19ubuntu1) 4.8.2" # 0x6020e0 0x00400e22 mov rax, qword [rip+0x2011b7] ; usage: %s port: % # 0x40125a 0x00400e29 mov esi, str.usage__s_port_n 0x00400e2a mov rdi, rax 0x00400e31 mov eax, 0 0x00400e36 call sym.imp.printf ; -1; -1 0x00400f3b mov eax, 0xffffffff ; (main) 0x00400f40 jmp 0x401169 </pre>				
Functions Disassembler Hex Dump Strings Settings Information	<pre> 0x00400b30-0x400e95 </pre>				Sections
<pre> vaddr=0x00400b30 paddr=0x00000b30 baddr=0x00400000 laddr=0x00000000 > ie [rehyppoints] vaddr=0x00400b30 paddr=0x00000b30 baddr=0x00400000 laddr=0x00000000 1 entrypoints ie </pre>					

BlessR2 (Node+Blessed)

blessr2 /bin/ls @ entry0

<pre>0x100001181 Sub rsp, 0x638 0x100001188 Mov rbx, rsi 0x10000118b Mov r14d, edi 0x10000118e Lea rax, [rbp - local_640h] 0x100001195 Mov qword [rbp - local_648h], rax 0x10000119c Test r14d, r14d 0x10000119f Jg 0x1000011a6 0x1000011a1 Call sym.func.1000043ff 0x1000011a6 Lea rsi, 0x100004af0 ; 0x10 0x1000011ad Xor edi, edi 0x1000011af Call sym.imp.setlocale 0x1000011b4 Mov r12d, 1 0x1000011ba Mov edi, 1 0x1000011bf Call sym.imp.isatty 0x1000011c4 Test eax, eax 0x1000011c6 Je 0x100001229 0x1000011c8 Mov dword [0x1000054d0], 0x50 ; 'P' ; [0x1000054d0] 0x1000011d2 Lea rdi, str.COLUMN\$; 0x100004af1 ; 0x1000011d9 Call sym.imp.getenv 0x1000011de Test rax, rax 0x1000011e1 Je 0x1000011f2 0x1000011e3 Cmp byte [rax], 0 0x1000011e6 Je 0x1000011f2 0x1000011e8 Mov rdi, rax 0x1000011eb Call sym.imp.atoi 0x1000011f0 Jmp 0x100001214 0x1000011f2 Lea rdx, [rbp - local_30h] 0x1000011f6 Mov edi, 1 0x1000011fb Mov esi, 0x40087468 0x100001200 Xor eax, eax 0x100001202 Call sym.imp.ioctl 0x100001207 Cmp eax, -1 0x10000120a Je 0x10000121a 0x10000120c Movzx eax, word [rbp - local_2eh] 0x100001210 Test eax, eax</pre>	<pre>file /bin/ls fd 6 size 0x9670 iorw false blksz 0x0 mode -r-- block 0x100 format mach064 havecode true pic true canary true nx false crypto false</pre>	<table border="1"><thead><tr><th>-</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr></thead><tbody><tr><td>174</td><td>5548</td><td>89e5</td><td>4157</td><td>4156</td><td></td><td></td><td></td></tr><tr><td>184</td><td>3806</td><td>0000</td><td>4889</td><td>f341</td><td></td><td></td><td></td></tr><tr><td>194</td><td>ff48</td><td>8985</td><td>b8f9</td><td>ffff</td><td></td><td></td><td></td></tr><tr><td>1a4</td><td>0000</td><td>488d</td><td>3543</td><td>3900</td><td></td><td></td><td></td></tr><tr><td>1b4</td><td>41bc</td><td>0100</td><td>0000</td><td>bf01</td><td></td><td></td><td></td></tr><tr><td>1c4</td><td>85c0</td><td>7461</td><td>c705</td><td>fe42</td><td></td><td></td><td></td></tr><tr><td>1d4</td><td>3d18</td><td>3900</td><td>00e8</td><td>2e33</td><td></td><td></td><td></td></tr><tr><td>1e4</td><td>3800</td><td>740a</td><td>4889</td><td>c7e8</td><td></td><td></td><td></td></tr><tr><td>1f4</td><td>55d0</td><td>bf01</td><td>0000</td><td>00be</td><td></td><td></td><td></td></tr><tr><td>204</td><td>3300</td><td>0083</td><td>f8ff</td><td>740e</td><td></td><td></td><td></td></tr><tr><td>214</td><td>8905</td><td>b642</td><td>0000</td><td>c705</td><td></td><td></td><td></td></tr><tr><td>224</td><td>4531</td><td>e4eb</td><td>1f48</td><td>8d3d</td><td></td><td></td><td></td></tr><tr><td>234</td><td>0048</td><td>85c0</td><td>740e</td><td>4889</td><td></td><td></td><td></td></tr><tr><td>244</td><td>8842</td><td>0000</td><td>e8d1</td><td>3200</td><td></td><td></td><td></td></tr><tr><td>0x100001254</td><td>c074</td><td>0cc7</td><td>85a8</td><td>f9ff</td><td></td><td></td><td></td></tr><tr><td>0x100001264</td><td>85a8</td><td>f9ff</td><td>fff0</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x100001274</td><td>c785</td><td>b4f9</td><td>ffff</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x100001284</td><td>0000</td><td>0000</td><td>c785</td><td>b0f9</td><td></td><td></td><td></td></tr><tr><td>0x100001294</td><td>eb08</td><td>e864</td><td>3100</td><td>0044</td><td></td><td></td><td></td></tr><tr><td>0x1000012a4</td><td>5138</td><td>0000</td><td>4489</td><td>f748</td><td></td><td></td><td></td></tr><tr><td>0x1000012b4</td><td>f860</td><td>7fd2</td><td>83f8</td><td>3f7f</td><td></td><td></td><td></td></tr><tr><td>0x1000012c4</td><td>0000</td><td>c705</td><td>4043</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x1000012d4</td><td>0000</td><td>0000</td><td>0000</td><td>41bc</td><td></td><td></td><td></td></tr><tr><td>0x1000012e4</td><td>b983</td><td>c09f</td><td>83f8</td><td>1777</td><td></td><td></td><td></td></tr><tr><td>0x1000012f4</td><td>159f</td><td>0700</td><td>0048</td><td>6304</td><td></td><td></td><td></td></tr><tr><td>0x100001304</td><td>0843</td><td>0000</td><td>0000</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x100001314</td><td>0000</td><td>c705</td><td>0443</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x100001324</td><td>76ff</td><td>ffff</td><td>83c0</td><td>c083</td><td></td><td></td><td></td></tr><tr><td>0x100001334</td><td>488d</td><td>0dbd</td><td>0700</td><td>0048</td><td></td><td></td><td></td></tr><tr><td>0x100001344</td><td>c705</td><td>fa42</td><td>0000</td><td>0100</td><td></td><td></td><td></td></tr><tr><td>0x100001354</td><td>ffff</td><td>c705</td><td>dc42</td><td>0000</td><td></td><td></td><td></td></tr><tr><td>0x100001364</td><td>0000</td><td>0000</td><td>0000</td><td>c705</td><td></td><td></td><td></td></tr><tr><td>0x100001374</td><td>4489</td><td>f9e9</td><td>22ff</td><td>ffff</td><td></td><td></td><td></td></tr><tr><td>0x100001384</td><td>0577</td><td>4100</td><td>0000</td><td>4489</td><td></td><td></td><td></td></tr></tbody></table>	-	0	1	2	3	4	5	6	174	5548	89e5	4157	4156				184	3806	0000	4889	f341				194	ff48	8985	b8f9	ffff				1a4	0000	488d	3543	3900				1b4	41bc	0100	0000	bf01				1c4	85c0	7461	c705	fe42				1d4	3d18	3900	00e8	2e33				1e4	3800	740a	4889	c7e8				1f4	55d0	bf01	0000	00be				204	3300	0083	f8ff	740e				214	8905	b642	0000	c705				224	4531	e4eb	1f48	8d3d				234	0048	85c0	740e	4889				244	8842	0000	e8d1	3200				0x100001254	c074	0cc7	85a8	f9ff				0x100001264	85a8	f9ff	fff0	0000				0x100001274	c785	b4f9	ffff	0000				0x100001284	0000	0000	c785	b0f9				0x100001294	eb08	e864	3100	0044				0x1000012a4	5138	0000	4489	f748				0x1000012b4	f860	7fd2	83f8	3f7f				0x1000012c4	0000	c705	4043	0000				0x1000012d4	0000	0000	0000	41bc				0x1000012e4	b983	c09f	83f8	1777				0x1000012f4	159f	0700	0048	6304				0x100001304	0843	0000	0000	0000				0x100001314	0000	c705	0443	0000				0x100001324	76ff	ffff	83c0	c083				0x100001334	488d	0dbd	0700	0048				0x100001344	c705	fa42	0000	0100				0x100001354	ffff	c705	dc42	0000				0x100001364	0000	0000	0000	c705				0x100001374	4489	f9e9	22ff	ffff				0x100001384	0577	4100	0000	4489			
-	0	1	2	3	4	5	6																																																																																																																																																																																																																																																																																			
174	5548	89e5	4157	4156																																																																																																																																																																																																																																																																																						
184	3806	0000	4889	f341																																																																																																																																																																																																																																																																																						
194	ff48	8985	b8f9	ffff																																																																																																																																																																																																																																																																																						
1a4	0000	488d	3543	3900																																																																																																																																																																																																																																																																																						
1b4	41bc	0100	0000	bf01																																																																																																																																																																																																																																																																																						
1c4	85c0	7461	c705	fe42																																																																																																																																																																																																																																																																																						
1d4	3d18	3900	00e8	2e33																																																																																																																																																																																																																																																																																						
1e4	3800	740a	4889	c7e8																																																																																																																																																																																																																																																																																						
1f4	55d0	bf01	0000	00be																																																																																																																																																																																																																																																																																						
204	3300	0083	f8ff	740e																																																																																																																																																																																																																																																																																						
214	8905	b642	0000	c705																																																																																																																																																																																																																																																																																						
224	4531	e4eb	1f48	8d3d																																																																																																																																																																																																																																																																																						
234	0048	85c0	740e	4889																																																																																																																																																																																																																																																																																						
244	8842	0000	e8d1	3200																																																																																																																																																																																																																																																																																						
0x100001254	c074	0cc7	85a8	f9ff																																																																																																																																																																																																																																																																																						
0x100001264	85a8	f9ff	fff0	0000																																																																																																																																																																																																																																																																																						
0x100001274	c785	b4f9	ffff	0000																																																																																																																																																																																																																																																																																						
0x100001284	0000	0000	c785	b0f9																																																																																																																																																																																																																																																																																						
0x100001294	eb08	e864	3100	0044																																																																																																																																																																																																																																																																																						
0x1000012a4	5138	0000	4489	f748																																																																																																																																																																																																																																																																																						
0x1000012b4	f860	7fd2	83f8	3f7f																																																																																																																																																																																																																																																																																						
0x1000012c4	0000	c705	4043	0000																																																																																																																																																																																																																																																																																						
0x1000012d4	0000	0000	0000	41bc																																																																																																																																																																																																																																																																																						
0x1000012e4	b983	c09f	83f8	1777																																																																																																																																																																																																																																																																																						
0x1000012f4	159f	0700	0048	6304																																																																																																																																																																																																																																																																																						
0x100001304	0843	0000	0000	0000																																																																																																																																																																																																																																																																																						
0x100001314	0000	c705	0443	0000																																																																																																																																																																																																																																																																																						
0x100001324	76ff	ffff	83c0	c083																																																																																																																																																																																																																																																																																						
0x100001334	488d	0dbd	0700	0048																																																																																																																																																																																																																																																																																						
0x100001344	c705	fa42	0000	0100																																																																																																																																																																																																																																																																																						
0x100001354	ffff	c705	dc42	0000																																																																																																																																																																																																																																																																																						
0x100001364	0000	0000	0000	c705																																																																																																																																																																																																																																																																																						
0x100001374	4489	f9e9	22ff	ffff																																																																																																																																																																																																																																																																																						
0x100001384	0577	4100	0000	4489																																																																																																																																																																																																																																																																																						

Gradare (Gtk2+Vte)



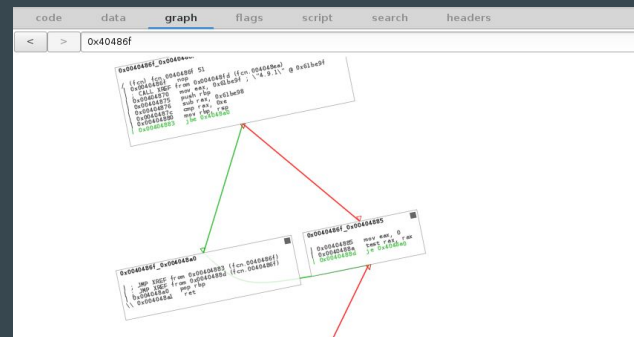
Ragui (abandoned/unreleased)

The screenshot shows the Ragui debugger interface. The top menu includes Project, Edit, View, Tools, Theme, and Help. The main window is divided into several panes:

- Information:** Displays file details for 'file' (./bin/file), including type (binary), size (116488), class (elf), type (EXEC (Executable file)), OS (linux), machine (AMD x86-64 architecture), subsystem (linux), bits (64), endian (little), and baddr (0x400000).
- Code:** Shows assembly code for function 'Fcn_00040485'. It includes instructions like 'xor ebp, ebp', 'mov r9, rdx', 'push rax', 'and rsp, 0xffffffff', 'push rsp', 'mov r2, 0x12290', 'rcx, 0x412190', 'mov rdi, 0x402820', 'call qword ptr [lib_start_main@unlkn.unk]', and 'hlt'.
- Notes:** A section for notes, currently empty.
- Code (Hex):** A hex dump of the code at address 0x404845, showing instructions like 'xor rax, rax', 'sub rax, 0x611e38', 'cap rax, 0x0', 'rtp, rsp', 'jmp 0x40484a', 'test rax, rax', 'jnz 0x40484a', 'push rbp', 'sub rax, 0x611e38', 'jmp rax', 'nop word cs:[rax + rax]', 'jmp rax', 'push rbp', 'sub rax, 0x611e38', 'sar rax, 3', 'rtp, rsp', and 'rav, ret'.
- Command List:** A list of commands such as '[a][b]', '[name][sz][at]', '[new]', '[file]', '[sdb-query]', '[n]', '[Offaet]', '[len]', '[ret]', '[len]', '[addr]', '[S]', '[[-] [runmsg]', '[ret]', '[len]', '[len][[0]addr]', '[?][exp]', and '??'.

The screenshot shows the Symbols window in the Ragui debugger. The window title is 'Project Edit View Debug Tools Help'. The Symbols window has tabs for 'code', 'data', 'graph', 'script', 'search', 'headers', and 'debug'. The main area displays a list of symbols with columns for Index, Address, Offset, Size, Relocations, Virtual Size, Privileges, and Name. The symbols include:

- 0x21f280 _programe
- 0x1558c _fini
- 0x21f290 optind
- 0x3498 _init
- 0x21f2a0 program_invocation_name
- 0x21f268 __bss_start
- 0x220448 _end
- 0x21f2a0 __programe_full
- 0x141a0 __obstack_memory_used
- 0x21f260 __obstack_alloc_failed_handler
- 0x13fd0 __obstack_begin
- 0x21f268 _edata
- 0x21f2c0 stder
- 0x21f2c0 __obstack_free
- 0x21f280 program_invocation_short_name
- 0x214100 __obstack_allocated_p
- 0x21f298 optarg
- 0x13ff0 __obstack_begin_1
- 0x14010 __obstack_newchunk
- 0x21f288 stduot



Bokken (Py/Gtk2)

The screenshot displays the Bokken application window, titled "Bokken, a GUI for pyew and radare2!". The interface includes a menu bar (File, Edit, View, Help), a toolbar with icons for search and navigation, and a search box containing "String".

The left sidebar contains several panels: "Function" (listing functions like entry0, fcn.00401080, etc.), "Sections" (listing sections like section..text), "Imports" (listing imported symbols), and "Exports" (listing exported symbols).

The main window displays assembly code for function "fcn.00402" at address "0x00400f90". The code includes instructions such as `xor ebp, ebp`, `mov r9, rdx`, `pop rsi`, `mov rdx, rsp`, `and rsp, 0xffffffffffff0`, `push rax`, `push rsp`, `mov r8, 0x405ab0`, `mov rcx, 0x405ac0`, `mov rdi, 0x401220`, `call dword imp._libc.start.main`, `hlt`, `nop`, `sub rsp, 0x8`, `MOV rax = [0x607fd8] = 0x0`, `mov rax, [rip+0x207011]`, `test rax, rax`, `jz @x400fce`, `call rax`, `unk()`, `add rsp, 0x8`, and `ret`.

The status bar at the bottom indicates the processor is "AMD x86-64 architecture", the name is "/bin/true", the format is "elf", and the version is "Bokken 1.5-dev".

R2G4W (.NET/MFC)

The screenshot displays the Immunity Debugger interface with the following components:

- Assembly View:** Shows the disassembled code for function `entry0`. The code includes variable declarations for `local_180h` through `local_5h`, a call to `GetCurrentProcessId`, and a loop structure with `push esi`, `push ecx`, `push ebx`, `push esp`, and `sub esp, 0x180`. The loop contains a call to `GetCurrentProcessId` and a `je` instruction at address `0x001130`.
- Hex View (pxa 4000):** Shows the raw memory bytes corresponding to the assembly code, with offsets from `000010e0` to `00001130`.
- Call Graph (agf):** A window showing a graph of the function's calls, with a call to `GetCurrentProcessId` highlighted.
- Command Window:** Contains the following commands:

```
[0x4010e0] > izzz # Strings 824
[0x4010e0] > pd 256 # Disassembly 535
[0x4010e0] > pxa 4000 # Hexview 909
```

At the bottom of the debugger window, the file path is shown as `pe 1B6B082C0A167C4E96DB96146EDA8A0.FIL [0x4010e0] > pxa 4000`.

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

The screenshot shows a debugger window titled "CRACKME.EXE - R2GUI". The main pane displays assembly code with various instructions and control flow statements. The code includes jumps, moves, and comparisons, with some instructions being cross-referenced to other parts of the program. The assembly code is as follows:

```
0x0040119f eb45 jmp 0x4011e6
; JMP XREF from 0x00401155 (sym.crackme.EXE.WndProc)
0x004011a1 eb43 jmp 0x4011e6
; JMP XREF from 0x0040113c (sym.crackme.EXE.WndProc)
0x004011a3 eb41 jmp 0x4011e6
; JMP XREF from 0x00401146 (sym.crackme.EXE.WndProc)
0x004011a5 b8000000 mov eax, 0
0x004011a8 eb3a jmp 0x4011e6
; JMP XREF from 0x0040115b (sym.crackme.EXE.WndProc)
0x004011ac 8b5d14 mov ebx, dword [ebp + arg_14h]; [0x14:4]=0
0x004011af c74319180100 mov dword [ebx + 0x81], 0x118; [0x118:4]=0x1902010b
0x004011b6 c7431ca0e000 mov dword [ebx + 0x1c], 0xa0; [0xa0:4]=0
0x004011bd c74320180100 mov dword [ebx + 0x20], 0x118; [0x118:4]=0x1902010b
0x004011c4 c74324a0e000 mov dword [ebx + 0x24], 0xa0; [0xa0:4]=0
0x004011cb b8000000 mov eax, 0
0x004011d0 eb14 jmp 0x4011e6
; JMP XREF from 0x00401164 (sym.crackme.EXE.WndProc)
0x004011d2 837d1067 cmp dword [ebp + arg_10h], 0x67; [0x67:4]=0x6e75206e; 'g'; "n under Win32..$7"
0x004011d6 7415 je 0x4011ed
0x004011d8 837d1065 cmp dword [ebp + arg_10h], 0x65; [0x65:4]=0x706e7572; 'e'; "run under Win32..$7"
0x004011dc 74b5 je 0x401193
0x004011de 837d1066 cmp dword [ebp + arg_10h], 0x66; [0x66:4]=0x75206e75; 'f'; "un under Win32..$7"
0x004011e2 7425 je 0x4011e9
0x004011e4 eb00 jmp 0x4011e6
; XREFS: JMP 0x00401171 JMP 0x0040117b JMP 0x00401191
; XREFS: JMP 0x00401197 JMP 0x004011a1 JMP 0x004011a3
; XREFS: JMP 0x004011a8 JMP 0x004011b6 JMP 0x004011bd
; XREFS: JMP 0x00401207 JMP 0x00401226 JMP 0x0040124a
; XREFS: JMP 0x00401251
0x004011e6 5b pop ebx
0x004011e7 5f pop edi
0x004011e8 5e pop esi
0x004011e9 c9 leave
0x004011ea c21000 ret 0x10
; JMP XREF from 0x004011d6 (sym.crackme.EXE.WndProc)
```

The bottom pane shows a hex dump of memory. The first few lines of the hex dump are as follows:

```
Offset: 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x004014d6 ff 25 14 32 40 00 ff 25 1c 32 40 00 ff 25 20 32 %20 %20 %20 %2
0x004014e0 40 00 ff 25 24 32 40 00 ff 25 28 32 40 00 ff 25 2c 32 40 00 ff 25 30 32 40 00 ff 25 34 32 40 00
0x004014f6 2c 32 40 00 ff 25 30 32 40 00 ff 25 34 32 40 00 %20 %020 %420
0x00401500 ff 25 38 32 40 00 ff 25 3c 32 40 00 ff 25 40 32 %820 %c20 %02
0x00401510 40 00 ff 25 48 32 40 00 ff 25 4c 32 40 00 ff 25 50 %h20 %l20 %
0x00401520 50 32 40 00 ff 25 58 32 40 00 ff 25 5c 32 40 00 P20 %X20 %X20
0x00401530 ff 25 60 32 40 00 ff 25 64 32 40 00 ff 25 68 32 %20 %d20 %h2
0x00401540 40 00 ff 25 6c 32 40 00 ff 25 70 32 40 00 ff 25 74 %l20 %p20 %
0x00401550 74 32 40 00 ff 25 78 32 40 00 ff 25 80 32 40 00 t20 %x20 %20
0x00401560 ff 25 84 32 40 00 ff 25 88 32 40 00 00 %20 %20 %20
0x00401570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00401580 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00401590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x004015a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x004015b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x004015c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

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

laito - /Users/hteso/Pocs/true32

```

0x08048d60 section_end_pit:
0x08048d60 section_text:
0x08048d60 (fcn) main 261
0x08048d60 ; arg int arg_8h @ ebp+0x8
0x08048d60 ; arg int arg_ch @ ebp+0xc
0x08048d60 ; var int local_4h @ esp+0x4
0x08048d60 ; var int local_8h @ esp+0x8
0x08048d60 ; var int local_ch @ esp+0xc
0x08048d60 ; var int local_10h @ esp+0x10
0x08048d60 ; var int local_14h @ esp+0x14
0x08048d60 push ebp
0x08048d61 sub.setlocale_d61:
0x08048d61 mov ebp, esp
0x08048d63 push ebx
0x08048d64 and esp, 0xffffffff
0x08048d67 sub esp, 0x20
0x08048d6a cmp dword [ebp + arg_8h], 2
0x08048d70 je 0x08048d7c
0x08048d77 call sym.imp.exit
0x08048d7c mov edx, dword [ebp + arg_ch]
0x08048d7f mov eax, dword [edx]
0x08048d81 mov dword [esp], eax
0x08048d84 call sub.fwrite_340
0x08048d89 mov dword [esp + local_4h], 0x0804b63f
0x08048d91 mov dword [esp], 0
0x08048d96 call sym.imp.setlocale
0x08048d9d mov dword [esp + local_4h], str._usr_share_locale
0x08048da5 mov dword [esp], 0x0804b67c
0x08048dac call sym.imp.bindtextdomain
0x08048db1 mov dword [esp], 0x0804b67e
0x08048db8 call sym.imp.textdomain
0x08048dbd mov dword [esp], 0x080491c0
0x08048dc4 call sub._calloc_text_340
0x08048dc9 mov eax, dword [ebp + arg_ch]
0x08048dcc mov ebx, dword [eax + 4]
0x08048dcf mov dword [esp + local_4h], str.__help
0x08048d7f mov dword [esp], ebx
0x08048da7 call sym.imp.strcmp
0x08048daa test eax, eax
0x08048d41 je 0x08048e36
0x08048d63 mov dword [esp + local_4h], str._version
0x08048d6b mov dword [esp], eax
                
```

```

; [14] va=0x08048d60 pa=0x00000d50 sz=9756 vsz=97f
; [0x2:4]=0x101464c
; [0xc:4]=0
; [0x0804b63f:4]=0x75727400
; [0x0804b68a:4]=0x7273752f LEA str._usr_share_locc
; [0x0804b67c:4]=0x65726f53
; [0x0804b67e:4]=0x65726f53
; [0x080491c0:4]=0x120ec83
; [0xc:4]=0
; [0x4:4]=0x10101
; [0x04b69c:4]=0x65682d2d LEA str.__help ; "--hel
; [0x0804b6a3:4]=0x65762d2d LEA str._version ; "--
                
```

main

```

[14] va=0x08048d60 pa=0x00000d50 sz=9756 vsz=97f
; [0x2:4]=0x101464c
; [0xc:4]=0
; [0x0804b63f:4]=0x75727400
; [0x0804b68a:4]=0x7273752f LEA str._usr_share_locc
; [0x0804b67c:4]=0x65726f53
; [0x0804b67e:4]=0x65726f53
; [0x080491c0:4]=0x120ec83
; [0xc:4]=0
; [0x4:4]=0x10101
; [0x04b69c:4]=0x65682d2d LEA str.__help ; "--hel
; [0x0804b6a3:4]=0x65762d2d LEA str._version ; "--
                
```

Function: .text:main

Information

Cyclomatic complexity

XRefs

Basic Blocks

Rfs

Offset info:

FAMILY cpu

STACK inc

ESIL ebp,4,esp,-,esp,[4]

TYPE upush

SIZE 1

REFPTR 0

BYTES 55

ID 588

PREFIX 0

OPCODE push ebp

ADDRESS 0x08048d60

Xrefs from:

Address Instruction

Dashboard

main

Functions
Strings
Imports
Symbols
Notepad

Sections

Name	Size	Address	End Address
.text	9756	0x08048d60	0x0804b37c
.shstrtab	237	0x00000000	0x0ed
.rodata	2464	0x0804b3a0	0x0804bd40
.rel.plt	312	0x08048974	0x08048aac
.rel.dyn	40	0x0804894c	0x08048974
.plt	640	0x08048ae0	0x08048d60

Sections
Comments

> Loading file: /Users/hteso/Pocs/true32

> Analysis finished

> Populating UI

> Adding binary information to notepad

> Finished, happy reversing :)

-- There is no FS key in radare2 yet

Type "?* for help

laito (Qt/C++)

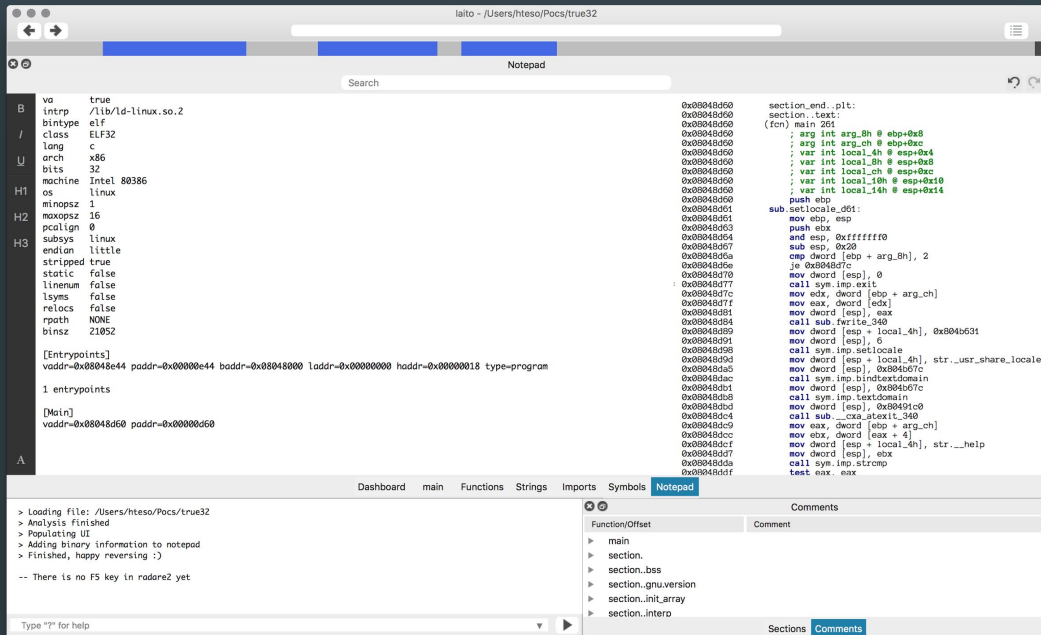
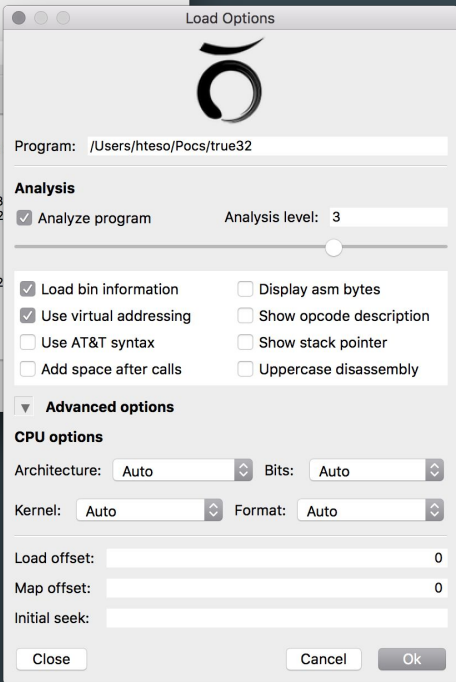
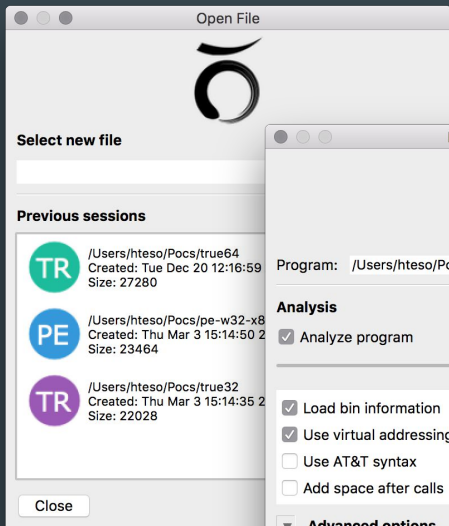
The screenshot displays the laito application window, titled "laito - /Users/hteso/Pocs/true32". The interface is divided into several panes:

- Functions:** A list of functions including `fcn.0804b2d5`, `fcn.0804b33a`, `fcn.0804b37c`, `fcn.0804b393`, `loc.imp_gmon_sta...`, and `main`.
- Disassembly:** A detailed view of the `main` function, showing assembly instructions with their addresses and hex values. For example, `0x0804bd60 55 89 e5 53 83 e4 f0 83 ec 20 83 7d 08 02 74 0c U.S.].t.`
- Comments:** A pane showing comments for the disassembled instructions, such as `> Loading file: /Users/hteso/Pocs/true32`, `> Analysis finished`, `> Populating UI`, `> Adding binary information to notepad`, `> Finished, happy reversing :)`, and `-- There is no F5 key in radore2 yet`.
- Bottom Panel:** A navigation pane with tabs for `Dashboard`, `main`, `Strings`, `Imports`, `Symbols`, and `Notepad`. Below the tabs are sections for `Function/Offset` and `Comment`.

The disassembly pane shows the following instructions (addresses and hex values are truncated for brevity):

```
0x0804bd60 55 89 e5 53 83 e4 f0 83 ec 20 83 7d 08 02 74 0c U.S. ....].t.
0x0804bd70 c7 04 24 00 00 00 e8 e4 fe ff ff 8b 55 0c 8b $. ....U.
0x0804bd80 02 89 04 24 e8 b7 05 00 00 c7 44 24 04 31 b6 04 $. ....DS.1.
0x0804bd90 08 c7 04 24 06 00 00 e8 43 ff ff c7 44 24 $. ....C.
0x0804bda0 04 8a b6 04 08 c7 04 24 7c b6 04 08 e8 6f ff ff $. ....$.
0x0804bdb0 ff c7 04 24 7c b6 04 08 e8 e3 fd ff c7 04 24 $. ....$.
0x0804bdc0 c0 91 04 08 e8 77 28 00 00 8b 45 0c 8b 59 04 c7 $. ....w..E.X..
0x0804bd00 44 24 04 0c b6 04 08 89 1c 24 e8 11 fd ff ff 85 $. ....$.
0x0804bde0 c0 74 53 c7 44 24 04 a3 b6 04 08 89 1c 24 e8 fd $. ....$.
0x0804bf00 fc ff ff 05 00 01 85 78 ff ff ff a1 04 e0 04 08 $. ....U.
0x0804be00 c7 44 24 14 00 00 00 c7 44 24 10 ad b6 04 08 $. ....$.
0x0804be10 c7 44 24 08 78 b6 04 08 89 44 24 0c a1 e0 e0 04 $. ....$.
0x0804be20 08 c7 44 24 04 32 b6 04 08 89 04 24 e8 7f ff 00 $. ....$.
0x0804be30 00 e9 3a ff ff ff c7 04 24 00 00 00 e8 e0 00 $. ....$.
0x0804be40 00 00 00 31 ed 5e 89 e1 83 e4 f0 50 54 52 60 $. ....PTH
0x0804be50 d0 b2 04 08 e8 b2 04 08 51 56 68 6d 04 08 $. ....$.
0x0804be60 e8 2b fe ff ff f4 90 90 90 90 90 90 90 90 90 $. ....QVN
0x0804be70 b8 bf e0 04 08 2d bc e0 04 08 83 f8 06 77 02 f3 $. ....w.
0x0804be80 c3 b8 00 00 00 85 e0 74 15 85 69 e5 83 ec 18 $. ....U.
0x0804be90 c7 04 24 bc e0 04 08 ff d0 c9 c3 90 8d 74 26 00 $. ....$.
0x0804bea0 b8 bc e0 04 08 2d bc e0 04 08 c1 f8 02 89 c2 c1 $. ....
0x0804beb0 e8 ff 01 d0 d1 f8 78 02 f3 c3 ba 00 00 00 85 $. ....
0x0804bec0 d2 74 15 85 69 e5 83 ec 18 89 44 24 04 c7 04 24 $. ....U.
0x0804bed0 bc e0 04 08 ff d2 c9 c3 90 8d b4 26 00 00 00 00 $. ....&
0x0804bee0 80 3d e4 e0 04 08 00 78 13 55 89 e5 83 ec 08 e8 $. ....U.
0x0804bf00 7c ff ff ff 05 e4 e0 04 08 91 ed f3 05 90 $. ....$.
0x0804bf00 a1 f8 de 04 08 85 c0 74 1e b8 00 00 00 85 c0 $. ....
0x0804bf10 74 15 85 69 e5 83 ec 18 c7 04 24 f8 de 04 08 ff f $. ....U.
0x0804bf20 80 c9 e9 79 ff ff ff ff ff ff ff ff ff ff ff ff $. ....U.
0x0804bf30 57 55 83 8c 10 8b 1d f0 e0 04 08 c7 44 24 08 W $. ....$.
0x0804bf40 05 00 00 c7 44 24 04 a8 b3 08 c7 04 24 00 00 $. ....$.
0x0804bf50 00 00 e8 28 fc ff ff 89 5c 24 0c 89 5c 24 00 $. ....$.
0x0804bf60 c7 04 24 01 00 00 89 44 24 04 e8 08 ff ff $. ....$.
0x0804bf70 c7 44 24 08 05 00 00 c7 44 24 04 e8 b3 04 08 $. ....$.
0x0804bf80 c7 04 24 00 00 00 e8 f4 fb ff ff c7 44 24 04 $. ....$.
0x0804bf90 2d b6 04 08 c7 04 24 91 08 00 00 44 24 e8 08 $. ....$.
0x0804ffa0 2c fd ff ff 8b 1d e0 e0 04 08 c7 44 24 08 05 00 $. ....$.
0x0804ffb0 00 c0 c7 44 24 04 14 b4 04 08 c7 04 24 00 00 00 $. ....$.
0x0804ffc0 00 8a fb ff ff 89 5c 24 04 89 04 24 e8 5e ff $. ....$.
0x0804fd00 ff ff 8b 1d e0 e0 04 08 c7 44 24 08 05 00 00 $. ....$.
0x0804fe00 c7 44 24 04 44 b4 04 08 c7 04 24 00 00 00 e8 $. ....$.
0x0804ff00 5c fb ff ff 89 5c 24 04 89 04 24 e8 78 fc ff $. ....$.
0x0804ff00 c7 44 24 08 05 00 00 c7 44 24 04 7c b4 04 08 $. ....$.
0x0804ff00 c7 44 24 08 05 00 00 c7 44 24 04 7c b4 04 08 $. ....$.
```

Iaito (Qt/C++)



A black and white photograph of a roller coaster track. The track is dark and silhouetted against a light, cloudy sky. It features several loops and drops, with a prominent vertical loop in the center. The track is supported by a network of dark metal beams. The overall mood is dramatic and complex.

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)



WHAT 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)

A high-contrast, black and white photograph of an oil field. The image is dominated by the silhouettes of several pumpjacks (oil pumps) against a bright, sunlit sky. The sun is positioned behind one of the pumpjacks, creating a strong lens flare and casting the entire scene into deep shadow. The sky is filled with scattered clouds, and the overall mood is industrial and stark. The word "Exploiting." is overlaid in the center in a clean, white, sans-serif font.

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 ($\geq 2.6.22$) until 2016 ($< 4.8.3$)).

(demo)

<https://dirtycow.ninja/>

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

A black and white photograph of a person standing on a rocky outcrop, looking out over a large, calm lake. The lake is surrounded by steep, forested mountains. The sky is overcast and misty, with some snow visible on the mountain slopes. The overall mood is contemplative and serene.

Questions.

