

Radare Demystified



r2@33C3/2016

pancake@opcode.org

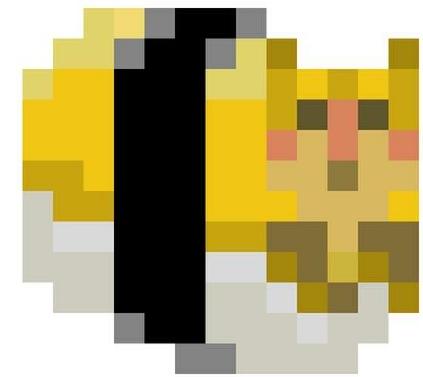
Introduction.



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

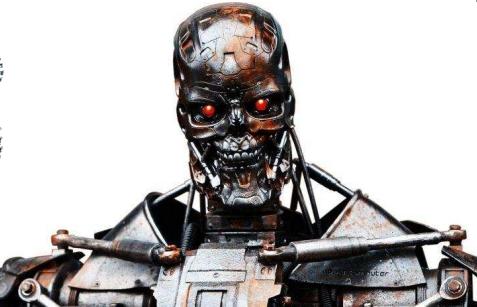
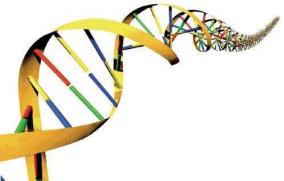
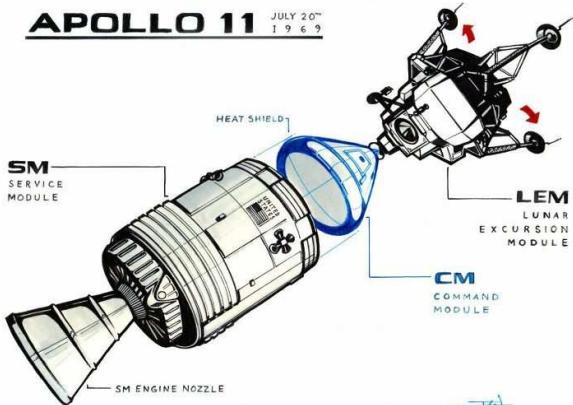


Stands For ‘Raw Data Recovery’

- 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?

APOLLO 11 JULY 20TH 1969



Wait..

Don't lose the rail

A dark, graffiti-covered tunnel with the word "Myths." overlaid.

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

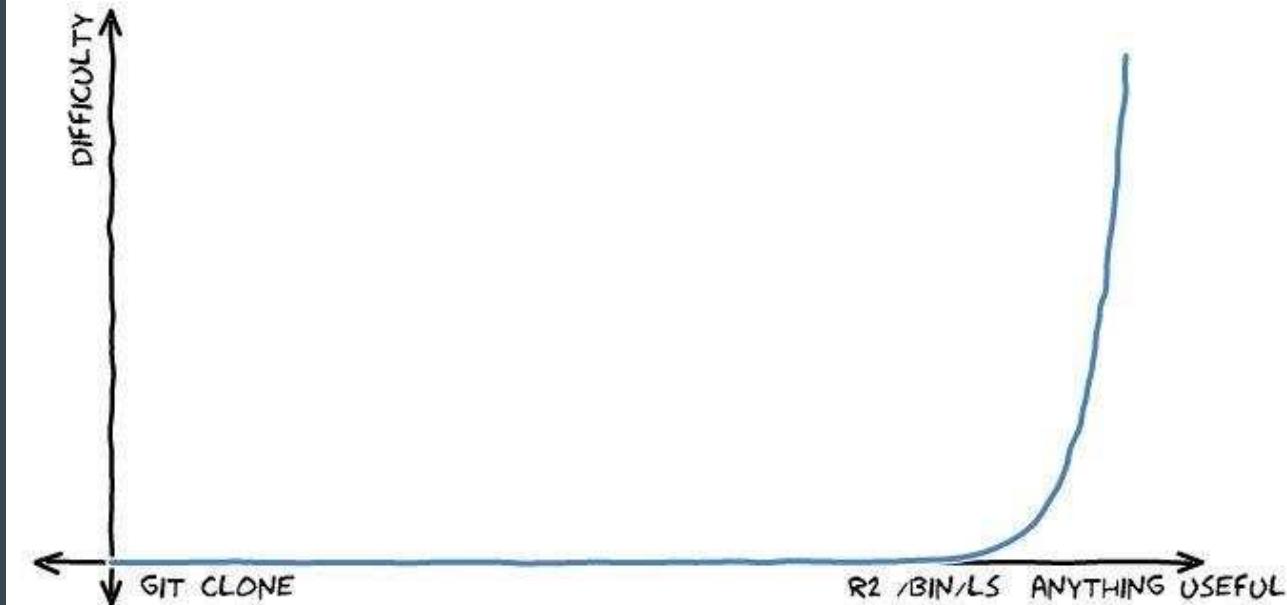
But First.. Let's Make a Poll

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

A black and white photograph of a person climbing a steep, rocky mountain slope. The climber is positioned in the center-left of the frame, leaning forward with an ice axe stuck into the rock. The mountain surface is rugged and textured. The sky is overcast and grey.

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

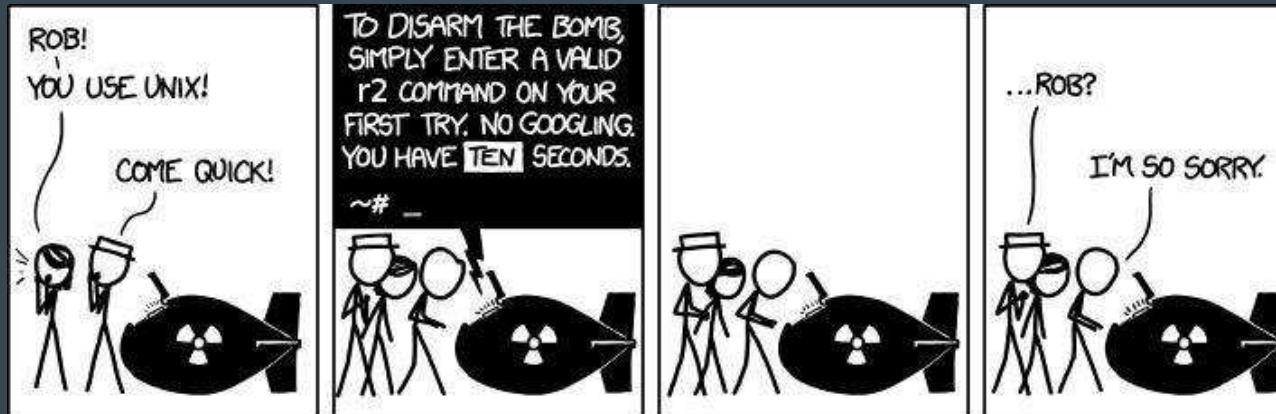
In the other hand...

- 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=

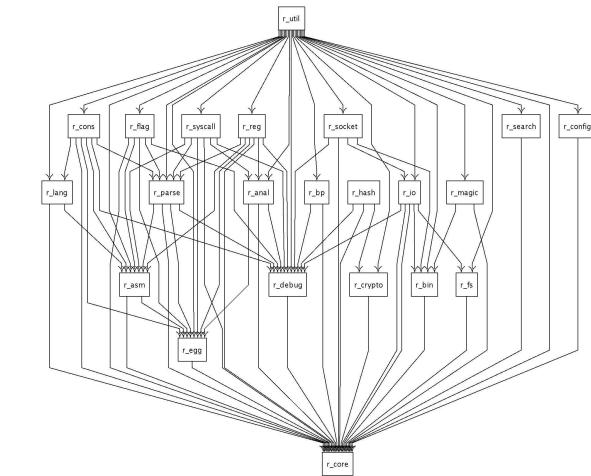
Hard To Remember

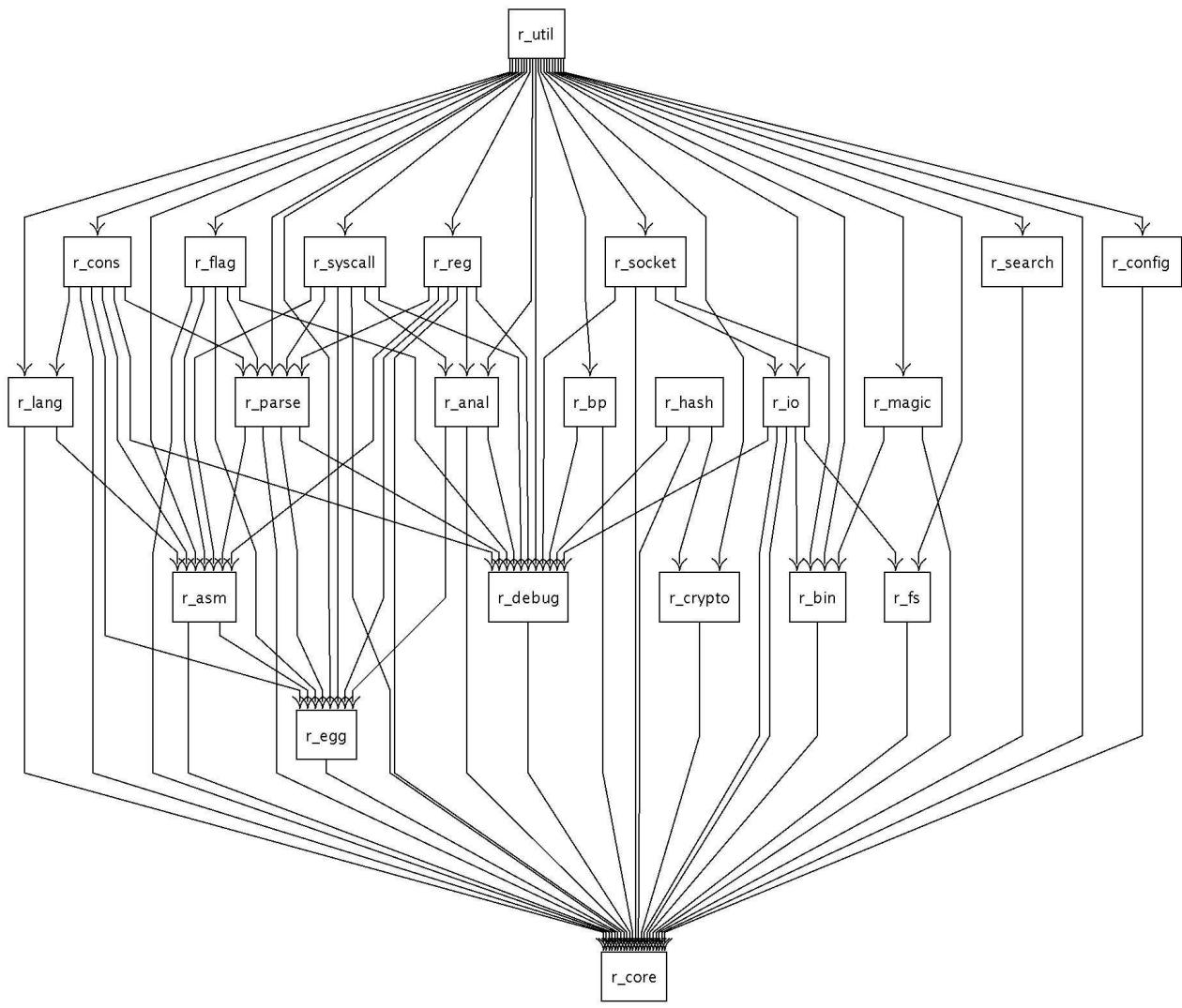
Commands follow simple mnemonic rules:

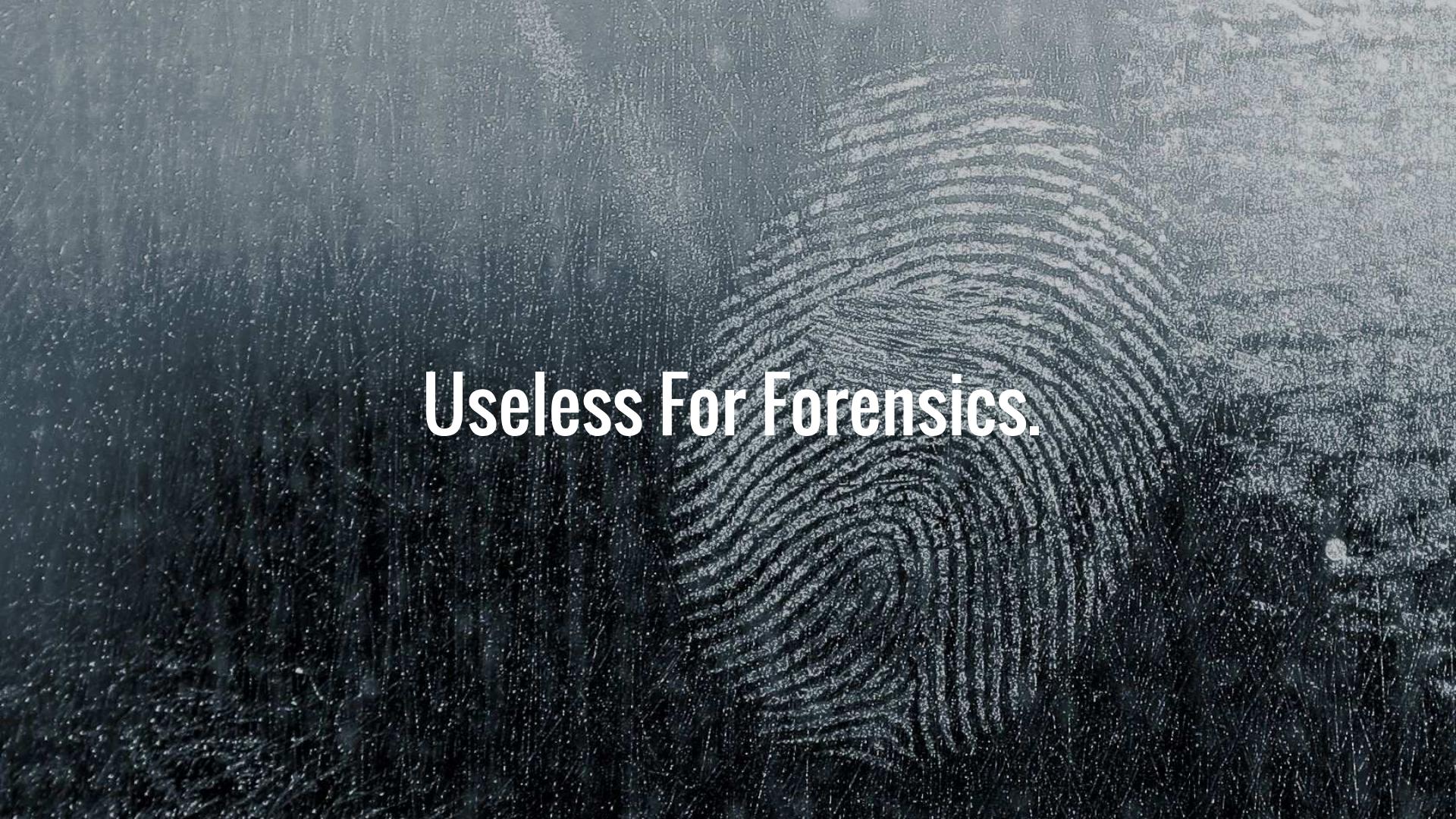
- Each char in the command is a subcommand of the previous one.
 - px -> print hexdump
 - pd -> print disasm
 - af -> analyze function
 - is -> info symbols
- Append ‘?’ to the command to get help about it
- Prefix with ‘.’ to interpret the output as commands
- Backticks are also supported x `?v 33`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







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

An aerial night photograph of a complex highway interchange. Multiple lanes of traffic are visible, with their headlights and尾灯 creating bright streaks against the dark asphalt. The interchange features several overpasses and ramps. In the background, there's a mix of urban development, including a gas station and some buildings. The overall scene conveys a sense of constant motion and activity.

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>

A dark, atmospheric photograph of an abandoned classroom. The room is filled with rows of old wooden desks and chairs, all of which are empty. The lighting is very low, coming from several large windows on the right wall, which are covered in graffiti and have broken panes. The floor is dirty and shows signs of decay. In the foreground, there's a large, dark wooden desk or cabinet. The overall mood is somber and eerie.

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)

A dark, heavily damaged airplane fuselage lies on its side on a black sand beach under a cloudy sky. The aircraft is tilted at an angle, with its front section crushed and twisted. Large holes pockmark the metal skin of the fuselage, and the interior framework is visible through the broken windows and doors. The scene is somber and conveys a sense of collapse or instability.

Not Stable.

It's Not Stable

Stability depends on

- The amount of crashes
- Commands and APIs changes.

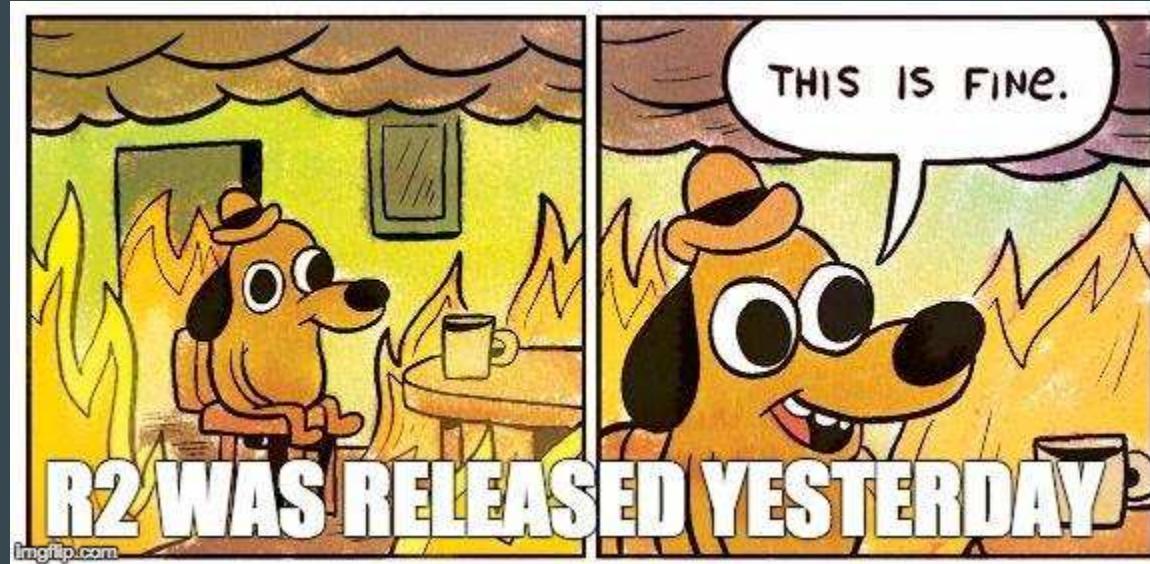
So...

- 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

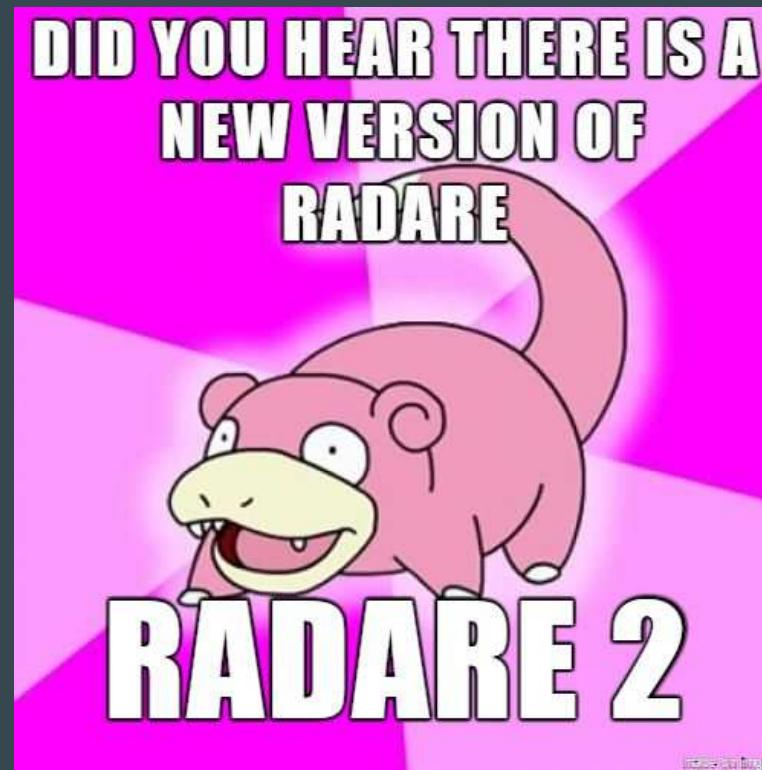
Feature X is Broken

- 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

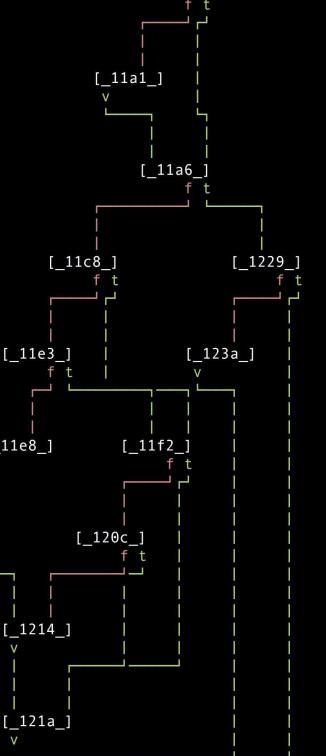
Writing UIs is boring and commandline tools are faster to develop and more flexible. But lazy minds usually like to wheel and click around instead of typing commands.

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)

```
[0x100001174]> VV @ sym.func.100001174 (nodes 79 edges 116 zoom 100%) BB-SMALL  
0x100001174:  
;-- main:  
;-- entry0:  
(fcn) sym.func.100001174 1392  
    sym.func.100001174 ()  
: var int local_660h @ rbp-0x660  
: var int local_658h @ rbp-0x658  
: var int local_654h @ rbp-0x654  
: var int local_650h @ rbp-0x650  
: var int local_64ch @ rbp-0x64c  
: var int local_648h @ rbp-0x648  
: var int local_640h @ rbp-0x640  
: var int local_440h @ rbp-0x440  
: var int local_34h @ rbp-0x34  
: var int local_30h @ rbp-0x30  
: var int local_2eh @ rbp-0x2e  
: var int local_0h @ rbp-0x0  
Push rbp  
Mov rbp, rsp  
Push r15  
Push r14  
Push r13  
Push r12  
Push rbx  
Sub rsp, 0x638  
Mov rbx, rsi  
Mov r14d, edi  
Lea rax, [rbp - local_640h]  
Mov qword [rbp - local_648h], rax  
Test r14d, r14d  
Jg 0x1000011a6 ;[a]  
  
f t  
v  
[ _11a1_ ]  
|  
v  
[ _11a6_ ]  
| f t  
| v  
[ _11c8_ ] [ _1229_ ]  
| f t | f t  
| v | v  
[ _11e3_ ] [ _123a_ ]  
| f t | f t  
| v | v  
[ _11e8_ ] [ _11f2_ ]  
| v | f t  
| [ _120c_ ]  
| f t  
| v  
[ _1214_ ]  
| v  
[ _121a_ ]  

```

```
[0x100001229 12% 215 /bin/ls]> ?0:f tmp:s..  
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 0123456789ABCDEF0123456789  
0x00000000 cffa edfe 0700 0001 0300 0080 0200 0000 1200 0000 0807 0000 8500 ..... H __PAGEZERO  
0x0000001a 2000 0000 0000 1900 0000 4810 0000 5f5f 5041 4745 5a45 524f 0000 .....  
0x00000034 0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 .....  
0x0000004e 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x00000068 1900 0000 2802 0000 5f5f 5445 5854 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x00000082 0000 0100 0000 0050 0000 0000 .....  
    rax 0x00000000    rbx 0x00000000    rcx 0x00000000    rdx 0x00000000  
    r8 0x00000000    r9 0x00000000    r10 0x00000000    r11 0x00000000  
    r14 0x00000000    r15 0x00000000    rip 0x00000000    rbp 0x00000000  
    0x100001229 488d3dc13800. Lea rdi, str.COLUMNS  
    0x100001230 e8d7320000 Call sym.imp.getenv  
    0x100001235 4885c0 Test rax, rax  
    < 0x100001238 740e Je 0x100001248  
    | 0x10000123a 4889c7 Mov rdi, rax  
    | 0x10000123d e86a320000 Call sym.imp.atoi  
    | 0x100001242 890588420000 Mov dword [0x1000054d0], eax  
    ; JMP XREF from 0x100001227 (sym.func.100001174)  
    > 0x100001248 e8d1320000 Call sym.imp.getuid  
    0x10000124d 41bd1000000 Mov r13d, 0x10  
    0x100001253 85c0 Test eax, eax  
    < 0x100001255 740c Je 0x100001263  
    | 0x100001257 c785a8f9ffff. Mov dword [rbp - local_658h], 0  
    | < 0x100001261 eb11 Jmp 0x100001274  
    | > 0x100001263 c785a8f9ffff. Mov dword [rbp - local_658h], 0  
    | 0x10000126d c6058c420000 Mov byte [0x100005500], 1  
    ; JMP XREF from 0x100001261 (sym.func.100001174)  
    > 0x100001274 c785b4f9ffff. Mov dword [rbp - local_64ch], 0  
    0x10000127e c785acf9ffff. Mov dword [rbp - local_654h], 0  
    0x100001288 c785b0f9ffff. Mov dword [rbp - local_650h], 0  
    0x100001292 31c9 Xor ecx, ecx  
    < 0x100001294 eb08 Jmp 0x10000129e  
    >> 0x100001296 e864310000 Call sym.func.1000043ff  
    || 0x10000129b 4489f9 Mov ecx, r15d  
    || ; JMP XREF from 0x100001294 (sym.func.100001174)  
    || ; JMP XREF from 0x1000012e3 (sym.func.100001174)  
    >> 0x10000129e 4189cf Mov r15d, ecx  
    || 0x1000012a1 488d151513800. Lea rdx, str._ABC...  
    || 0x1000012a8 4489f7 Mov edi, r14d  
    || 0x1000012ab 4889de Mov rsi, rbx  
    || 0x1000012ae e85f320000 Call sym.imp getopt  
    || 0x1000012b3 83f860 Cmp eax, 0x60 .....  
    ; `'; ``'; ``'
```

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
| | -- fcн.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

```
0x1000000000 0 __mh_execute_header
0x05614542 0 radr://5614542
0x10000444c 0 imp._assert_rtn
0x100004452 0 imp._bzero
0x100004458 0 imp._error
0x10000445e 0 imp._maskrun
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
0x7fff5fbffea6	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
0x7fff5fbffee6	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0x7fff5fbffef4	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	28ff	(.

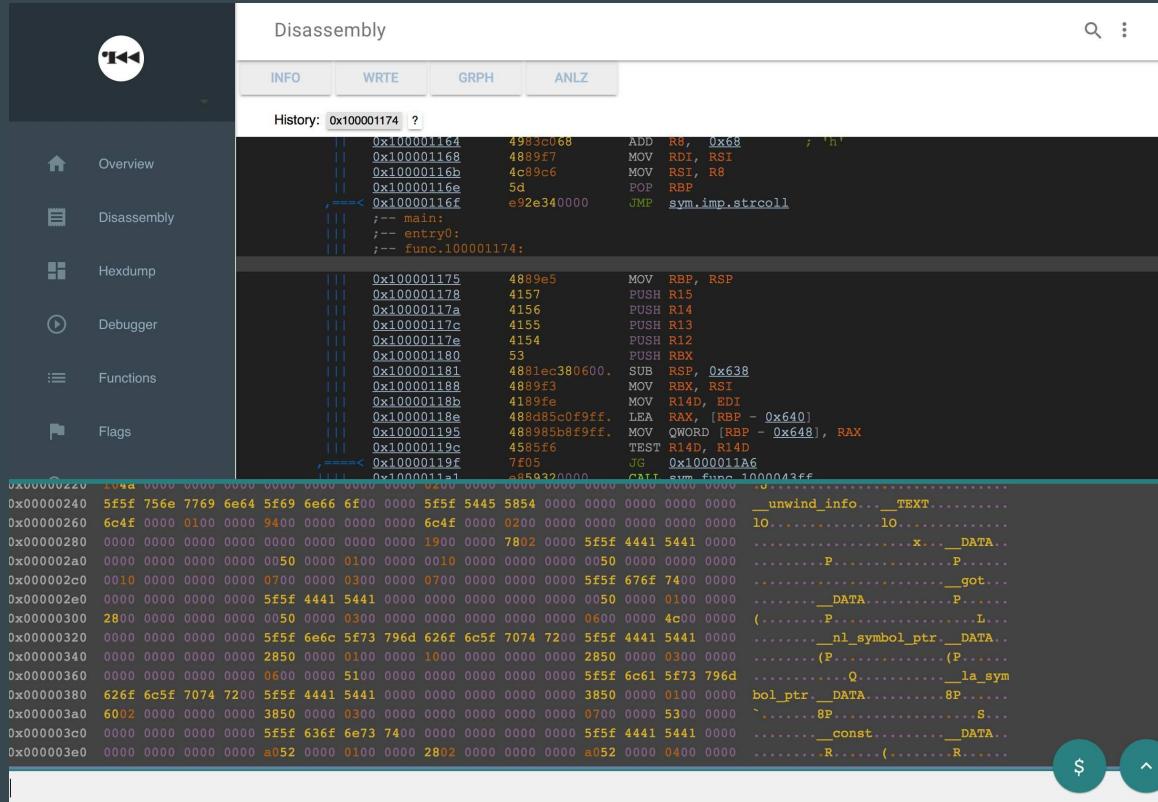
| Registers

```
rax 0x00000000      rbx 0x00000000      rcx 0x00000000
rdx 0x7fff5fbff38      rdi 0x10000000      rsi 0x00000001
rbp 0x7fff5fbff00      rsp 0x7fff5fbffea0      r8 0x7fff5fc00000
r9 0x7fff5fbff20      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 0x00007ffff5fbff38 (04_copy_user-rwx) rdx R W 0x7fff5fbff38
rdi 0x0000000100000000 rdi 0x0000000100000000 (00_copy/bin/ls-rwx) rdi R X 'iretd' 'ls'
rsi 0x0000000000000001 rsi
rbp 0x00007ffff5fbff00 (04_copy_user-rwx) rbp R W 0x7fff5fbff00
rsp 0x00007ffff5fbffea0 (04_copy_user-rwx) rsp R W 0x0 --> r15
r8 0x00007ffff5fc00000 (04_copy_user-rwx) r14 R X 'iretd' 'dyld'
r9 0x00007ffff5fbff20 (04_copy_user-rwx) r9 R W 0x0 --> r15
```

WebUI (=H,/m)



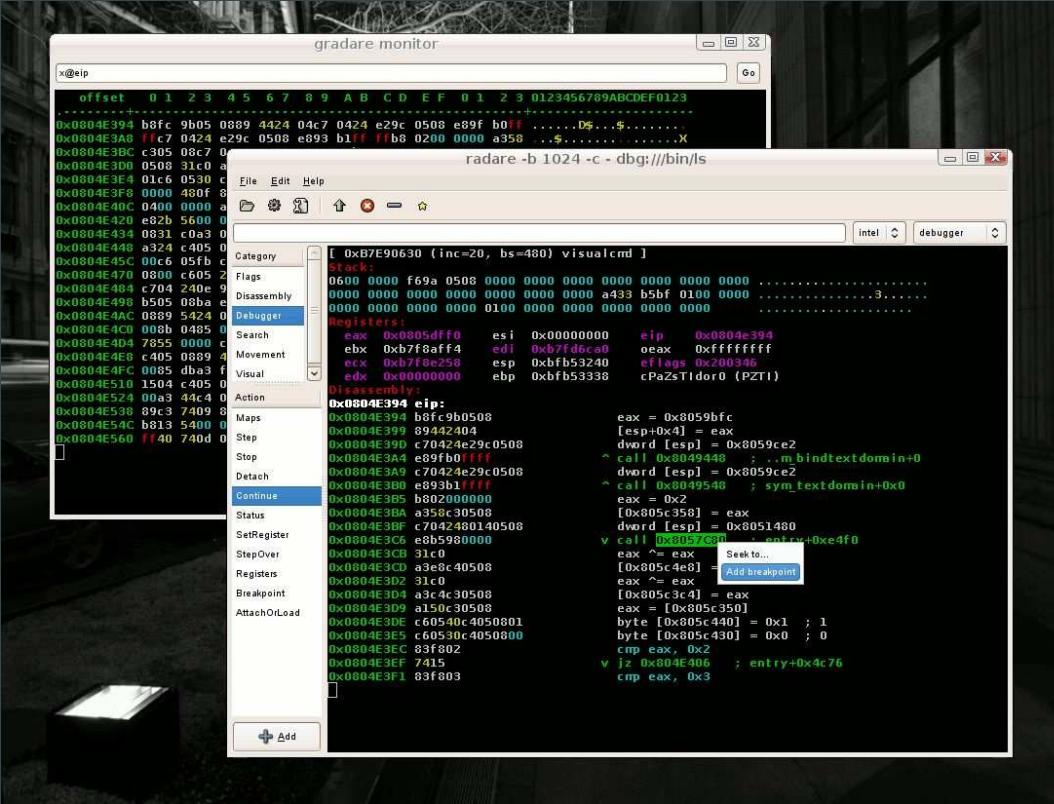
WebUI (=H,/p)

BlessR2 (Node+Blessed)

blessr2 /bin/ls @ entry@

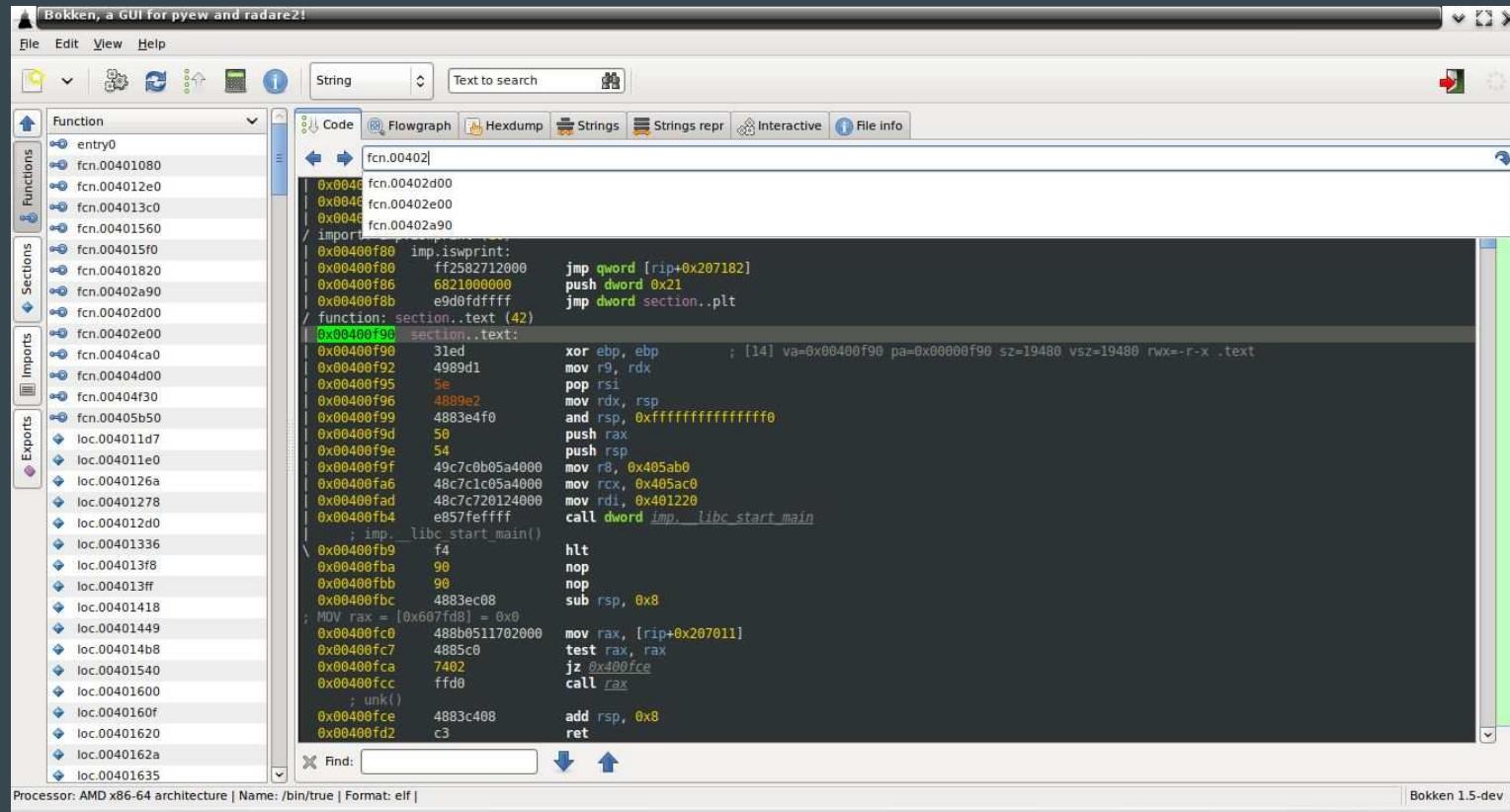
0x100001181	Sub	rsp, 0x638	file	/bin/ls	-	0	1	2
0x100001188	Mov	rbx, rsi	fd	6	174	5548	89e5	4157
0x10000118b	Mov	r14d, edi	size	0x9670	184	3806	0000	4889
0x10000118e	Lea	rax, [rbp - local_640h]	iowr	false	194	ff48	8985	b8f9
0x100001195	Mov	qword [rbp - local_648h], ra	blkSz	0x0	1a4	0000	48bd	3543
0x10000119c	Test	r14d, r14d	mode	-r--	1b4	41bc	0100	0000
0x10000119f	Jg	0x1000011a6	block	0x100	1c4	85c0	7461	c705
0x1000011a1	Call	sym.func.1000043ff	format	mach064	1d4	3d18	3900	e8e2
0x1000011a6	Lea	rsi, 0x100004af0 ; 0x10	havecode	true	1e4	3800	7460	4889
0x1000011ad	Xor	edi, edi	pic	true	1f4	55d0	bf01	00be
0x1000011af	Call	sym.imp.setlocale	canary	true	204	3300	0088	f8ff
0x1000011b4	Mov	r12d, 1	nx	false	214	8905	b642	0000
0x1000011ba	Mov	edi, 1	crypto	false	224	4531	e4eb	1f48
0x1000011bf	Call	sym.imp.isatty			234	0048	85c0	740e
0x1000011c4	Test	eax, eax			244	8842	0000	e8d1
0x1000011c6	Je	0x100001229				0x100001254	c074	0cc7
0x1000011c8	Mov	dword [0x1000054d0], 0x50 ; 'P' : [0x1000054d0				0x100001264	85ab	f9ff
0x1000011d2	Lea	rdi, str.COLUMNS ; 0x100004af1				0x100001274	c785	b4f9
0x1000011d9	Call	sym.imp.getenv				0x100001284	0000	0000
0x1000011de	Test	rax, rx				0x100001294	eb08	e864
0x1000011e1	Je	0x1000011f2				0x1000012a4	5138	0000
0x1000011e3	Cmp	byte [rax], 0				0x1000012b4	f860	7f2d
0x1000011e6	Je	0x1000011f2				0x1000012c4	0000	c705
0x1000011e8	Mov	rdi, rx				0x1000012d4	4043	0000
0x1000011eb	Call	sym.imp.atoi				0x1000012e4	0000	0000
0x1000011f0	Jmp	0x100001214				0x1000012f4	b983	83f8
0x1000011f2	Lea	rdx, [rbp - local_30h]				0x100001304	159f	0700
0x1000011f6	Mov	edi, 1				0x100001314	0843	0000
0x1000011fb	Mov	esi, 0x40087468				0x100001324	0000	c705
0x100001200	Xor	eax, eax				0x100001334	488d	0dbd
0x100001202	Call	sym.imp.ioctl				0x100001344	c705	f424
0x100001207	Cmp	eax, -1				0x100001354	ffff	c705
0x10000120a	Je	0x10000121a				0x100001364	0000	dc42
0x10000120c	Movzx	eax, word [rbp - local_2eh]				0x100001374	4489	f9e9
0x100001210	Test	eax, eax				0x100001384	0577	4100

Gradare (Gtk2+Vte)

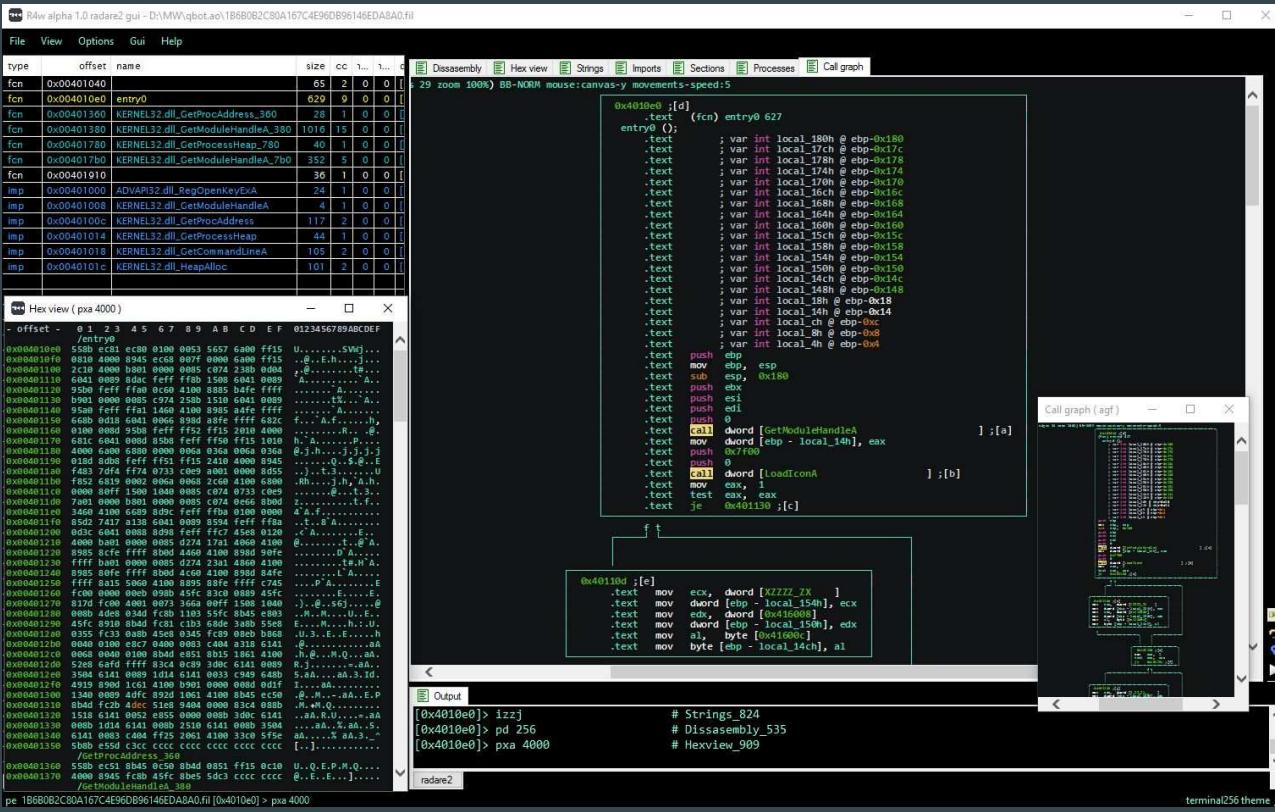


Ragui (abandoned/unreleased)

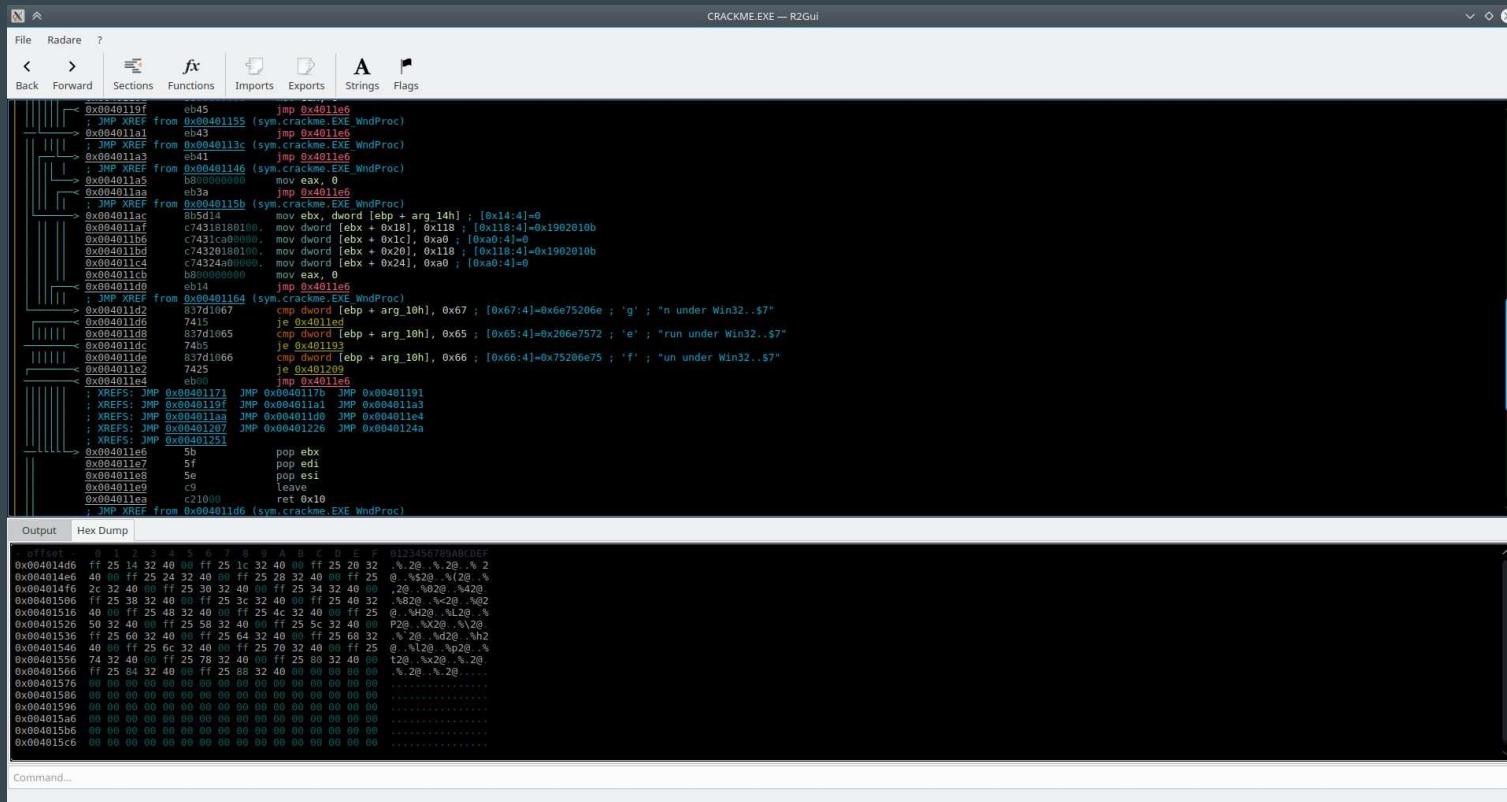
Bokken (Py/Gtk2)



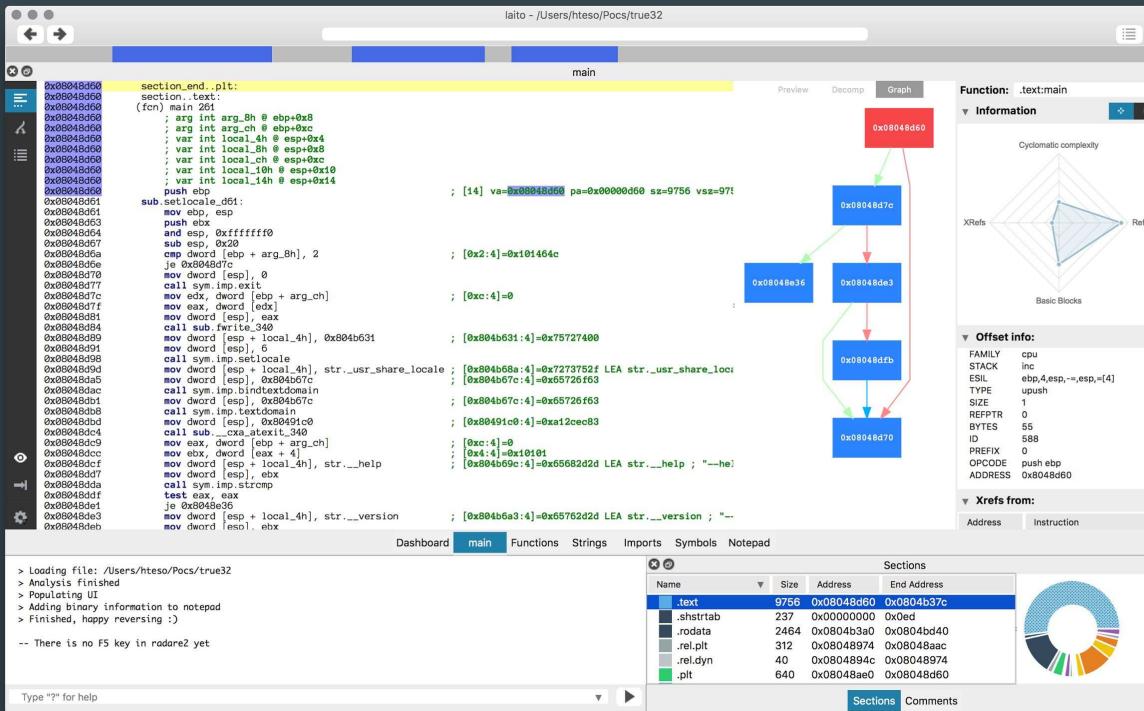
R2G4W (.NET/MFC)



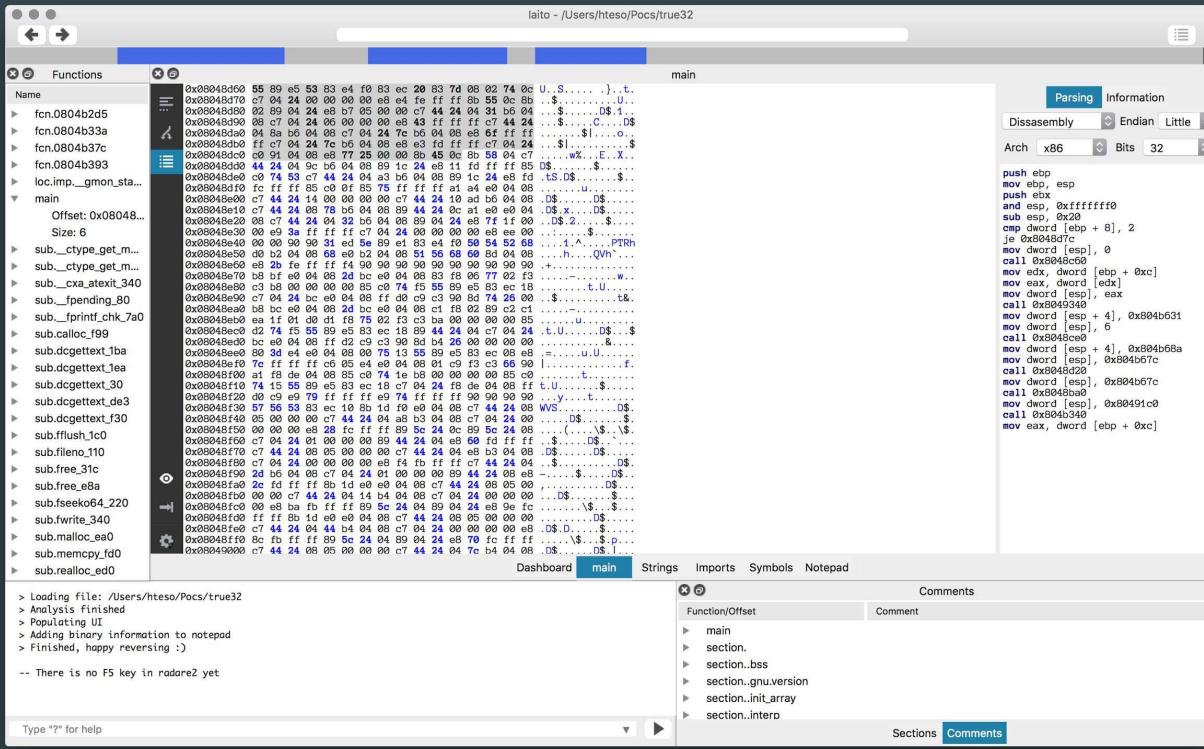
R2GUI (QT5/C++) (3 days ago)



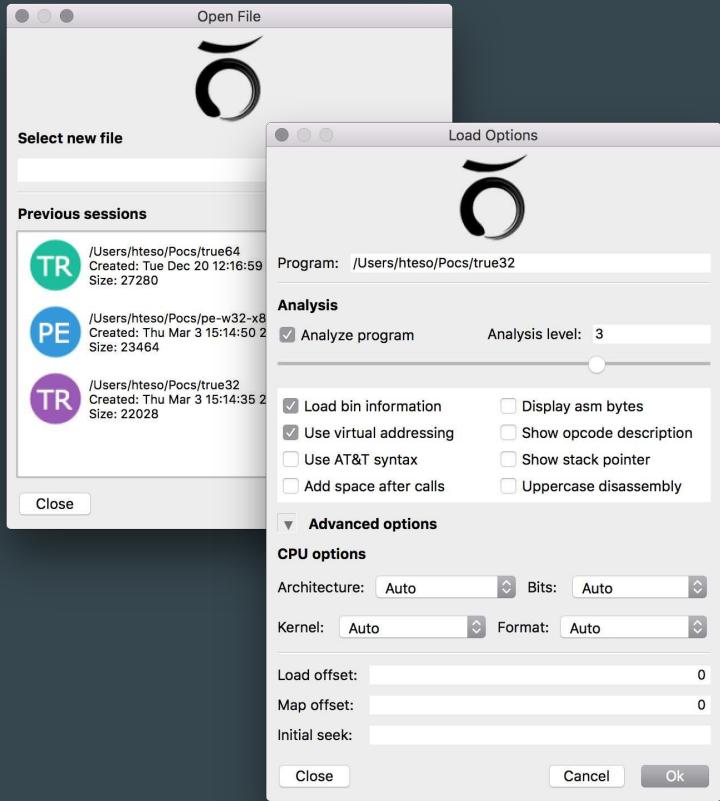
laito (Qt/C++) (alpha release on early 2017)



laito (Qt/C++)



laito (Qt/C++)



The image shows the main interface of laito with the title bar 'laito - /Users/hteso/Pocs/true32'. The assembly view displays the following code:

```
    v0      true
    v1      /lib/ld-linux.so.2
    bintype elf
    class  ELF32
    long   c
    arch   x86
    bits   32
    machine Intel 80386
    os     Linux
    minopsz 1
    H1     maxopsz 16
    pcalign 0
    H2     subsys linux
    entry  _start
    stripped true
    static  false
    lineum false
    lsyms  false
    relocs false
    rpath  NONE
    binsz  21052

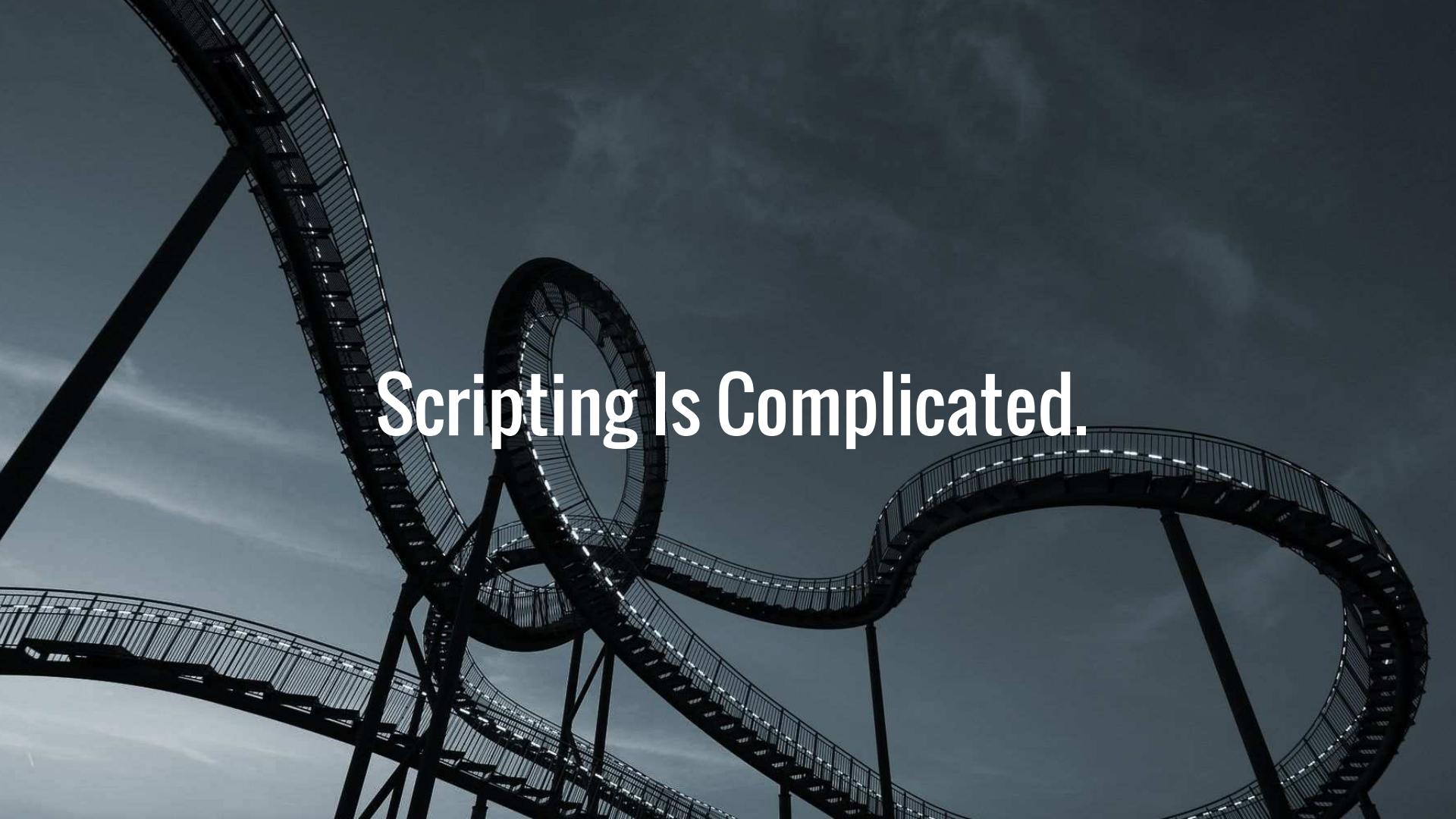
[Entrypoints]
vaddr=0x08048e44 paddr=0x000000e44 baddr=0x08048000 laddr=0x00000000 haddr=0x00000018 type=program
1 entrypoints

[Main]
vaddr=0x08048d60 paddr=0x000000d60
```

The notes pane at the bottom right contains the following text:

```
> 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
```

The notes pane has tabs for 'Dashboard', 'main', 'Functions', 'Strings', 'Imports', 'Symbols', and 'Notepad'. The 'Notepad' tab is selected. It includes sections for 'Function/Offset', 'Comment', 'Sections', and 'Comments'.

A dark, high-angle photograph of a roller coaster track against a cloudy sky. The track is composed of multiple dark, winding steel beams with railings, forming several loops and turns. The perspective is from below, looking up at the complex structure of the track.

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

A large mining excavator is silhouetted against a dark sky, its bright lights illuminating the surrounding area. The machine is positioned on the right side of the frame, facing towards the left. Its arm is extended downwards, and its bucket is partially visible. The background shows a dark landscape with some distant lights.

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/mm

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)

A dark, high-contrast photograph showing several silhouetted oil derrick structures against a bright, cloudy sky. The scene is framed by the dark shapes of the structures and the bright highlights on the clouds.

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 dark, moody photograph of a man standing on a rocky outcrop, looking out over a large body of water and a range of mountains shrouded in mist.

Questions.

