r2 CON 2016

# radare2-1.0

## The curse of the milestone

--pancake

## Introduction

I’m the author of this thing called **radare** and this talk is gonna be about the state of the project, its weak points, their fixes and the future plans.

**DevRevs**: A concept that I invented to refer to the reverse engineers that are also developers (and vice versa). Main contributors have this profile.

If you enjoy this talk you are probably one of those.

## The culture of versioning

The version number of a program can tell you a lot of things about the software and the developers behind. There are several ways of understanding version numbers, and this may vary depending on the project or the language used.

**What do 1.0 means?**

- Initial release (others use 0.0.1 for this)
- First stable version
- Complete and finished version of the software

## The culture of versioning

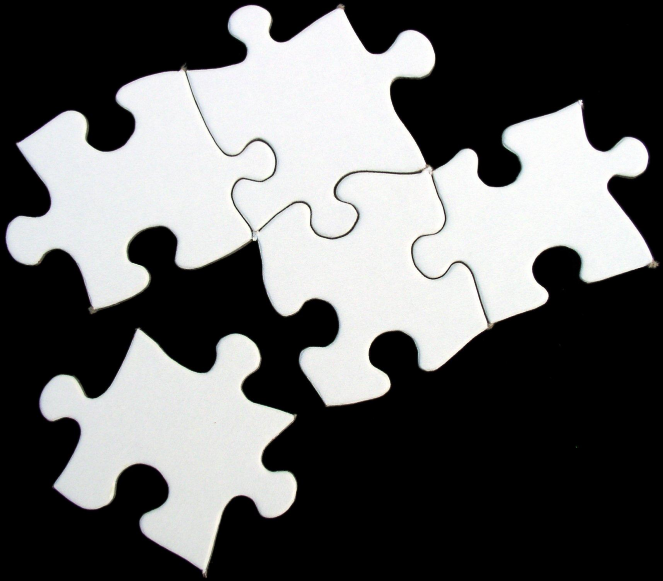
Version numbers can be just a number

- 43 (browsers like Mozilla Firefox or Google Chrome)
- 1.x (nmap, coreutils, jq, ...)
- 1.x.x (like r2, capstone, git, gnome, apache, ...)
  - Maybe it encodes abi compatibility level.
- 1.x.x.x (tor, sqlite, snort, …)
- 1a (like openssl.)
- 1-beta
- YYYYMMdd
- YYYY-MM
- Snapshot
- git
- ...

## ABI compatibility

According to the sematic versioning: (1)

- MAJOR version when you make incompatible API changes.
- MINOR version when you add functionality in a backwards-compatible manner.
- PATCH version when you make backwards-compatible bug fixes.



*1 http://semver.

## Timeline

Some understand versions as an evolution line of the project, where major number is used to represent the milestones that show big changes in the interface or the program functionalities.

Sometimes two big version numbers are developed in parallel. Version numbers increase every some specified amount of time.

GTK changed their versioning and release cycles recently, most projects are moving to more frequent releases.

6 week release cycle since 0.10 (like Rust). Frequent updates, aims stable git and fixed milestones handy for project management.

## Branches

Some projects use different branches for devel, which requires an extra effort to backport patches to every branch in order to construct proper LTS updates.

In r2land we use 1 branch:

- Evolve faster
- Always refactoring
- Less maintainance work
- Focus on features and stability
- Drop deprecated APIs and commands ASAP
- Using pull requests with jenkins+travis+appveyour integration

## Are we 1.0 yet?

**Many people thought that it would arrive after 0.9...**

But 0.9.1 arrived

**After 0.9.9 many people believed the next would be finally 1.0...**

But 0.10 appeared

**Last release is 0.10.5, guess what will be next...**

## r2-v1.0

Yes, 1.0 has arrived, and this doesn’t mean that it’s a stable or complete and finished piece of software.

What 1.0 means for radare2 is that it has reached 10 years!

## r2-v1.0

The first release of 2017 will be the r2-1.0

- += 0.1 every 6 weeks
- += 1.0 every year

This way the versioning represents different snapshots over time, rather than informing false abi compatibility information.

## Development Model

In r2land we used to not follow any strict development model, which turned into a continuous evolution of the practices that worked for us, development, the packagers and the users.

- Using ACR (configure+make)
- sys/install.sh to simplify updates
- Handmade makefiles
- Precompiled sdb databases.
- ...

I used to increase a 0.0.1 increment with about 1000 commits after some months, so version.

## Weak points

...

## Weak points = Complains

Those are the main complains from the users:

- Commands are complex and hard to remember
- Nobody knows all features of r2
- It have no GUI
- Doesn’t compiles
- Bindings are not idiomatic and hard to install/use
- There’s no documentation
- It’s slow when working with big files
- Not usable for large projects
- It is vulnerable because it’s in C

## Problem: Huge Codebase

It is true that r2 is quite big in size, this can be a problem for embedding into new targets with low resources, or when compiling multiple versions to find which commit broke a regression test.

**Solutions**

- Always refactor, removing dead or wrong code, optimizing
- Disable plugins with ./configure-plugins (sys/tiny.sh)
- Move code from r2 to r2-extras and make it available via r2pm
- Specify default plugins
- Use CI systems to delegate
- Do more review on new code
- Standarize syntax and indentation rules

## Dependencies

Aim to be dep-free, but that’s because it ships them inside, and some are optionally compilable against system libraries:

- **Binutils** stuff (GPL) and not available as a library
- **Capstone** (last release is old and it comes with some vulns)
- **LibMagic** (GNU one is vulnerable, so I took OpenBSD one (which source is crap, but at least smaller, and fixed a couple of bugs (reported upstream))
- **TCC** - Tiny C Compiler (stripped and modified to fit into r2)
- **GRUB** (GPL) - FileSystem and Partitions (full of vulns, fixed many of them + api)
- **SDB** - Done by me, and kept in sync into the r2 codebase
- **Udis86** - handy for small builds, but abandoned (capstone++)
- **(Libre|Open|Boring)SSL** - Used for BigNum+SSL (unused)

## Vulnerabilities and Patches

Sometimes external dependencies have vulnerabilities, found by us or just reported somewhere else. In r2land, we handle segfaults and vulns as a maximum priority issue. Which use to be fixed before a day of being noticed.

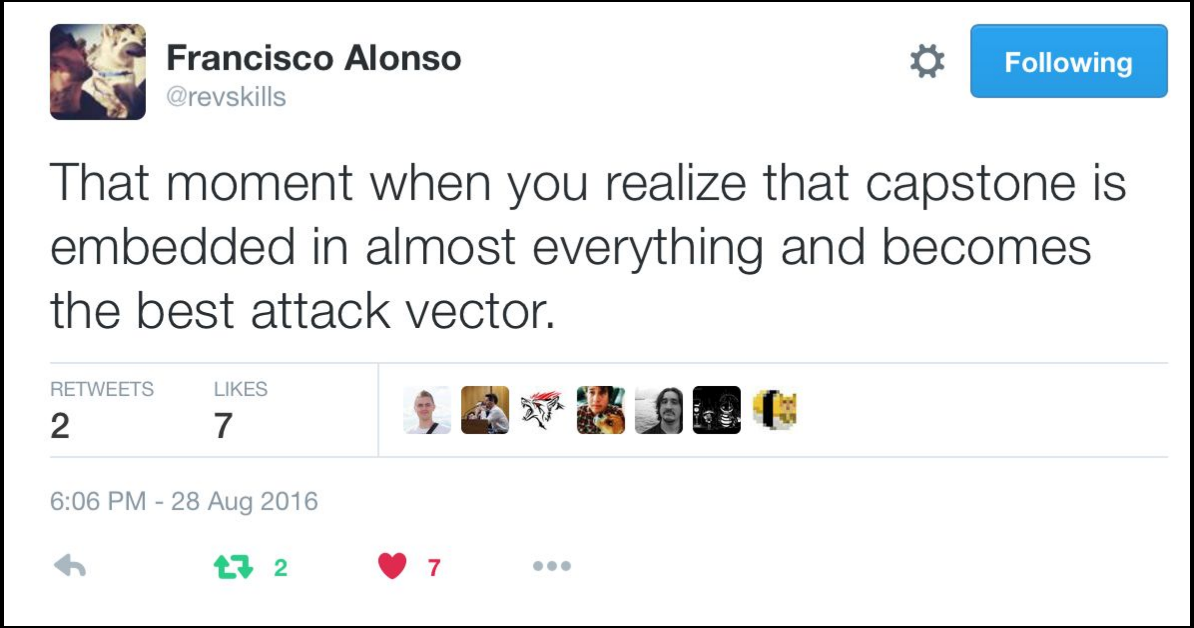
We ship also some patches to fix missbehaviours.

- aim to get pushed upstream
- use embedded libs + patches by default
- pick a specific branch + commit to detect regressions

## Vulnerabilities In Capstone

Revskills CVEs affecting git version of Capstone.

- **CVE-2016-7151** https://github.com/aquynh/capstone/pull/725
- **CVE-2016-4044** https://github.com/aquynh/capstone/issues/729
- **CVE-2016-3160** https://github.com/aquynh/capstone/issues/730


Francisco Alonso
@revskills

That moment when you realize that capstone is embedded in almost everything and becomes the best attack vector.

RETWEETS 2 LIKES 7

6:06 PM - 28 Aug 2016

## Dependency Problems

- We need to keep in sync with master
  - Not always easy if patches are applied
  - Patches on top of those projects must be simple
- We must push all fixes upstream, as well as report issues
- Distros dislike static linking, so we need to provide a way to link against system ones
  - Can make r2 vulnerable, or behave different than using the shipped versions
  - Simpler bug fixing for distros
  - Smaller Builds

## Solution: Packaging

Since r2-0.10.0 (2015-09-24) there's a program called r2pm that allows to install and manage external dependencies and plugins, installing everything in the home or system (-g)

**(DEMO)**

## C Language

C is the most portable language out there, there are compilers that can generate code for microcontrollers, or even to javascript to rule the web.

The good side

- It is pretty low level. Mostly WYWIWYG
- Many analysis and helper tools available
- It’s easy to debug

The bad side

- Requires skilled developers
- More time invested in code reviews
- Some time of the dev must be reinvested in fixing bugs

## Solution: Code Review

C is a weak language in the sense that provides little abstraction from the generated code and requires the developer to handle memory, resources, buffer sizes, endians swaps and other caveats that can make spend more time fixing bugs than developing.

- Multiple reviews before merging helps
- Defining good code practices
- Using Scan Coverity and Clang Analyzer
- Fuzzing and

## Coding Style

We use a custom coding style inspired by GNOME and Perl standards.

No tool is able to indent C code properly.

- Avoids control flow bugs (goto fail)
- Makes the code more readable
- Slowly adopting the coding style
- Not reindenting 3rd party code

## RDD - Regression Driven Development

Some projects use TDD, which means that you write tests before code.

In r2land I invented the RDD, which is basically the same, but in reversed order. The reason for this is because the way r2 is developed and its culture.

- Most parts of r2 has been done for a specific need or fun.
- radare->radare2 transition showed that having a perfect design from the beginning is impossible for projects big like this (and even small like dwm)
- Moving fast is a key piece of the project, we want things now.
- Writing tests is boring
- Testing everything is impossible (code coverage)
- Integration with Travis is important to avoid regressions, slowdowns the devel (wait for results), but avoids mayhem.

## RDD - Regression Driven Development

DEMO: Write a test

```
$ git clone https://github.com/radare/radare2-regressions

$ cd radare2-regressions

$ vim t/my_test

$ sh run_tests.sh t/my_test
```

## Infrastructure

Having a testsuite is not enough for catching the bugs in time, it is very useful to have travis building every single commit and running the testsuite on different system setups.

- Testsuite only running on Linux-x86-64 and OSX-x86-64 (travis+jenkins)
- Automated Windows, Android builds
- Manual iOS and Android updates
- Releasing requires some time (6 week helps)

## Endianness

R2 is the process of being fully big-endian friendly. (thx damo22)

The host endianness is now ignored completely, and just uses binary operations to grab the values in the specified endian (big or little).

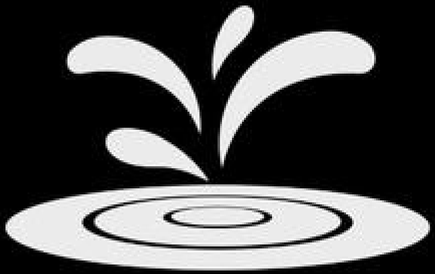
Configurable via cfg.bigendian.

- Host Endian (local cpu)
- Target Endian (target asm.arch for asm/disasm and pxw/pointers)
- Implicit Endian (bin files with fixed endianness)

## Memory leaks

Currently it is a problem. But not really dramatic.

- Use valgrind to profile the heap usage
- Commands with loops shouldn’t leak (pd, …)
- To leak or to double free
- Ownership concept problems
- No reference counting



## Other implicit bugs from C

- Null dereferences
- Buffer Overflows
- Heap overflows / underflows
- Format strings
- Use After Free
- Pointer alignment issues
- Double Frees
- Corrupted pointers
- Memory leaks
- OOB reads
- …

## Alternative languages

**The only language that fit in the field of C is probably Rust.**

- No big or mandatory runtime library
- No garbage collector
- Low level
- Easily to talk

**Bad points:**

- Young, but stable.
- Hard to find experienced programmers
- Mind shift.

## Other languages

As long as r2 is modular and pluggable it is possible to rewrite every single module in a different language being able to reuse the other parts and plugins written in C and just replacing the specific library.

- Swift
- Vala
- Nim/Ooc

Extending or replacing parts of r2 in dynamic or scripting languages like Python, NodeJS, Ruby or C# is possible, but for Core, it’s preferably to choose a language that:

- No garbage collection
- Low level (no mangling for public symbols, no runtime)
- On top of POSIX (slightly blocking for kernel or embedded land)

## Performance

Many people think that r2 must be fast because it is written in C. This is not a true statement because the way memory is managed and the algorithms implemented it can result in slower than other languages. But anyway, r2 is reasonably fast for most use cases, except for:

- Big files
  - Having a large number of flags or functions results in heavily slow r2
  - IO is complex
  - Many memory copies
  - RBin parsing duplicates memory
  - Repeated processes (analysis in pd, etc)
- Remote debugger
  - Non-cached requests

## Performance: IO

The libr/io is responsible of abstracting the access to files, memory, processes. It exposes several layers to define sections, maps, specify permissions, caching temporal patches, non-memoized functions, etc. All this complexity results in slow read operations.

- Do single reads in a block before a processing loop
- Don’t do small reads on every iteration
- Having many sections

## SDB

The disk/memory string key=value database is used in many places of r2 for:

- Easy to use hashtables
- Can be mmaped from disk, so it takes no memory
- Inspectable by the user (k command)
- Leak-free and clean api

**The bad side**

- A hashtable is not always
- Strings are not always the more optimal way to represent stuff
- Used for storing data of projects

## Analysis

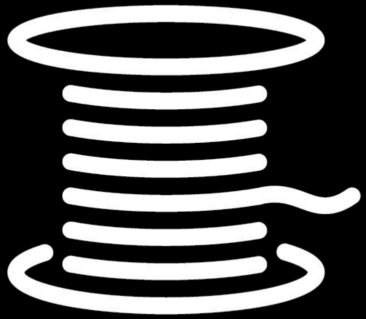
Statically analyzing code is not an easy problem to solve.

- Same loop for all archs
- Each anal plugin provides basic backend logic
- Merge with anal + asm?
- Reuse information from disasm and analysis to avoid having to disasm every instruction twice
- Recursive analysis is weak
    - Break analysis with wrong code
    - Code coverage is hard
- Requires emulation sometimes
- Storage problem

## Parallelism/Threading

R2 is monotask, it uses no threads at all. This decision is on purpose.

- Simple design
- Avoid unnecessary mutexes
  - Probably deadlocks
- Co-routines and async are hard
- Still lot to optimize before going into this
- Queues and snapshotting sounds better to me
- Basic support for threads (background webserver, ..)



## Debuggers

R2 is monotask, it uses no threads at all. This decision is on purpose

Weak points

- Unstable and poorly tested
- Early threading support
- All are sync operations
- More support for kernel
- Port dmh to more platforms

## Bindings

Bindings are automatically generated by valabind. A tool I wrote that reads vapi descriptions and generates bindings for swig or other targets.

Those bindings are not idiomatic and are pretty much 1:1 with the C API, so the same API documentation can be reused across all the languages.

Imposes some restrictions in the C code base of r2 to integrate with Vala:

- Don’t expose unions
- RList everywhere
- Implicit ownership rules
- Naming conventions for functions to be OO-like

## r2pipe

Simplest interface possible with r2: the shell. It provides a way to run a command and get the output in return.

This simple concept can be used thru sockets, file descriptors, native apis via dlopen(), http, rap, etc..

Deserializing JSON objects is faster than FFI.

Easy to port to new languages.

// demo

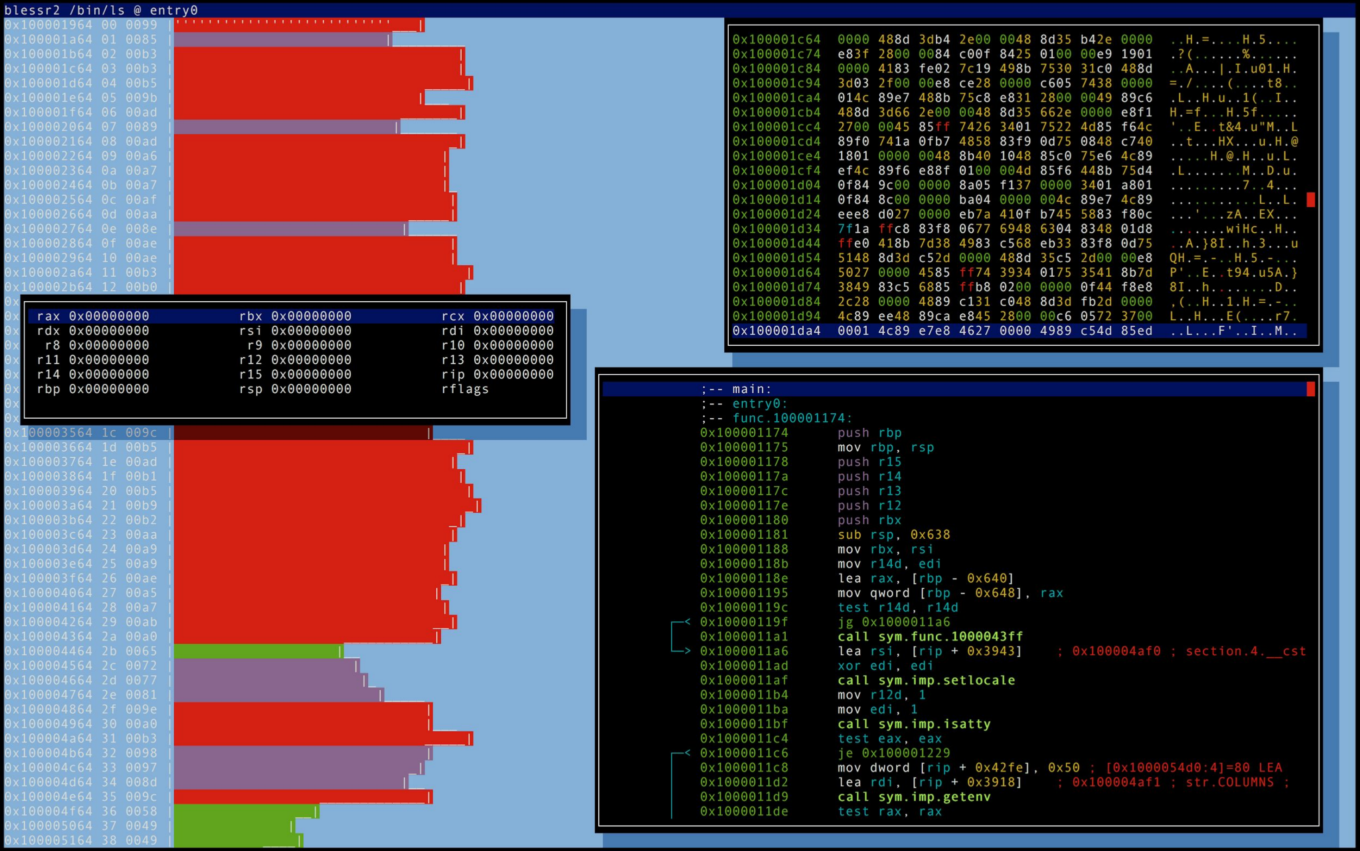
## User Interface

The main complain from new users is the lack of a user interface.

- Writing a UI is boring
- R2 api evolves and requires constant updates in both sides
- The shell is more powerful and expressive
- Need ways for remoting

# Solution 1: TUI

```
$ npm install blessr2
```



```
blessr2 /bin/ls @ entry0
0x100001964 00 0099 |
0x100001a64 01 0085 |
0x100001b64 02 00b3 |
0x100001c64 03 00b3 |
0x100001d64 04 00b5 |
0x100001e64 05 009b |
0x100001f64 06 00ad |
0x100002064 07 0089 |
0x100002164 08 00ad |
0x100002264 09 00a6 |
0x100002364 0a 00a7 |
0x100002464 0b 00a7 |
0x100002564 0c 00af |
0x100002664 0d 00aa |
0x100002764 0e 008e |
0x100002864 0f 00ae |
0x100002964 10 00ae |
0x100002a64 11 00b3 |
0x100002b64 12 00b0 |
0x
0x
0x
0x
0x
0x
0x
0x
0x
0x100003564 1c 009c |
0x100003664 1d 00b5 |
0x100003764 1e 00ad |
0x100003864 1f 00b1 |
0x100003964 20 00b5 |
0x100003a64 21 00b9 |
0x100003b64 22 00b2 |
0x100003c64 23 00aa |
0x100003d64 24 00a9 |
0x100003e64 25 00a9 |
0x100003f64 26 00ae |
0x100004064 27 00a5 |
0x100004164 28 00a7 |
0x100004264 29 00ab |
0x100004364 2a 00a0 |
0x100004464 2b 0065 |
0x100004564 2c 0072 |
0x100004664 2d 0077 |
0x100004764 2e 0081 |
0x100004864 2f 009e |
0x100004964 30 00a0 |
0x100004a64 31 00b3 |
0x100004b64 32 0098 |
0x100004c64 33 0097 |
0x100004d64 34 008d |
0x100004e64 35 009c |
0x100004f64 36 0058 |
0x100005064 37 0049 |
0x100005164 38 0049 |
```

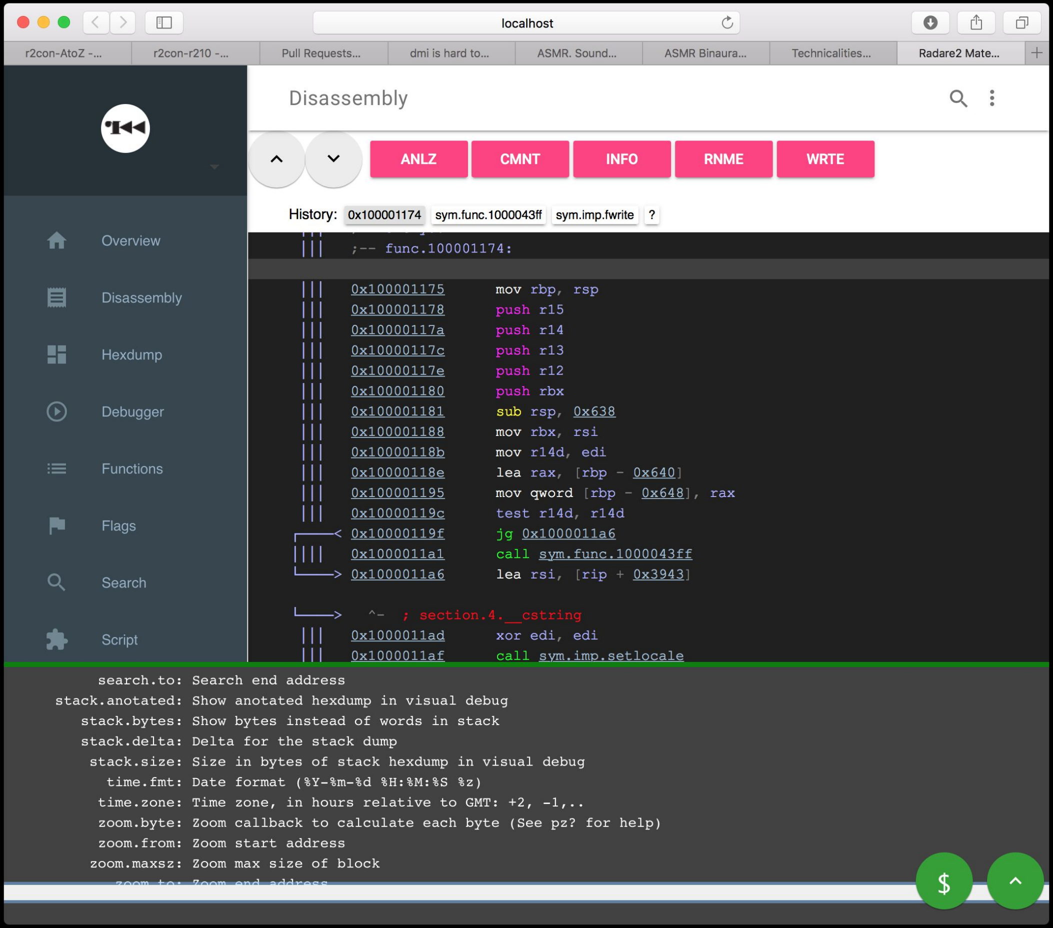
```
 rax 0x00000000            rbx 0x00000000            rcx 0x00000000
 rdx 0x00000000            rsi 0x00000000            rdi 0x00000000
  r8 0x00000000             r9 0x00000000            r10 0x00000000
 r11 0x00000000            r12 0x00000000            r13 0x00000000
 r14 0x00000000            r15 0x00000000            rip 0x00000000
 rbp 0x00000000            rsp 0x00000000            rflags
```

```
0x100001c64  0000 488d 3db4 2e00 0048 8d35 b42e 0000  ..H.=....H.5....
0x100001c74  e83f 2800 0084 c00f 8425 0100 00e9 1901  .?(......%......
0x100001c84  0000 4183 fe02 7c19 498b 7530 31c0 488d  ..A...|.I.u01.H.
0x100001c94  3d03 2f00 00e8 ce28 0000 c605 7438 0000  =./....(....t8..
0x100001ca4  014c 89e7 488b 75c8 e831 2800 0049 89c6  .L..H.u..1(..I..
0x100001cb4  488d 3d66 2e00 0048 8d35 662e 0000 e8f1  H.=f...H.5f.....
0x100001cc4  2700 0045 85ff 7426 3401 7522 4d85 f64c  '..E..t&4.u"M..L
0x100001cd4  89f0 741a 0fb7 4858 83f9 0d75 0848 c740  ..t...HX...u.H.@
0x100001ce4  1801 0000 0048 8b40 1048 85c0 75e6 4c89  .....H.@.H..u.L.
0x100001cf4  ef4c 89f6 e88f 0100 004d 85f6 448b 75d4  .L.......M..D.u.
0x100001d04  0f84 9c00 0000 8a05 f137 0000 3401 a801  .........7..4...
0x100001d14  0f84 8c00 0000 ba04 0000 004c 89e7 4c89  ...........L..L.
0x100001d24  eee8 d027 0000 eb7a 410f b745 5883 f80c  ...'...zA..EX...
0x100001d34  7f1a ffc8 83f8 0677 6948 6304 8348 01d8  .......wiHc..H..
0x100001d44  ffe0 418b 7d38 4983 c568 eb33 83f8 0d75  ..A.}8I..h.3...u
0x100001d54  5148 8d3d c52d 0000 488d 35c5 2d00 00e8  QH.=.-..H.5.-...
0x100001d64  5027 0000 4585 ff74 3934 0175 3541 8b7d  P'..E..t94.u5A.}
0x100001d74  3849 83c5 6885 ffb8 0200 0000 0f44 f8e8  8I..h........D..
0x100001d84  2c28 0000 4889 c131 c048 8d3d fb2d 0000  ,(..H..1.H.=.-..
0x100001d94  4c89 ee48 89ca e845 2800 00c6 0572 3700  L..H...E(....r7.
0x100001da4  0001 4c89 e7e8 4627 0000 4989 c54d 85ed  ..L...F'..I..M..
```

```
                 ;-- main:
                 ;-- entry0:
                 ;-- func.100001174:
                 0x100001174      push rbp
                 0x100001175      mov rbp, rsp
                 0x100001178      push r15
                 0x10000117a      push r14
                 0x10000117c      push r13
                 0x10000117e      push r12
                 0x100001180      push rbx
                 0x100001181      sub rsp, 0x638
                 0x100001188      mov rbx, rsi
                 0x10000118b      mov r14d, edi
                 0x10000118e      lea rax, [rbp - 0x640]
                 0x100001195      mov qword [rbp - 0x648], rax
                 0x10000119c      test r14d, r14d
             ┌─< 0x10000119f      jg 0x1000011a6
             │   0x1000011a1      call sym.func.1000043ff
             └─> 0x1000011a6      lea rsi, [rip + 0x3943]     ; 0x100004af0 ; section.4.__cst
                 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 [rip + 0x42fe], 0x50 ; [0x1000054d0:4]=80 LEA
             │   0x1000011d2      lea rdi, [rip + 0x3918]     ; 0x100004af1 ; str.COLUMNS ;
             │   0x1000011d9      call sym.imp.getenv
             │   0x1000011de      test rax, rax
```

# Solution 2: WebUI

```
$ r2 -c=H /bin/ls
```



```
;-- func.100001174:

0x100001175      mov rbp, rsp
0x100001178      push r15
0x10000117a      push r14
0x10000117c      push r13
0x10000117e      push r12
0x100001180      push rbx
0x100001181      sub rsp, 0x638
0x100001188      mov rbx, rsi
0x10000118b      mov r14d, edi
0x10000118e      lea rax, [rbp - 0x640]
0x100001195      mov qword [rbp - 0x648], rax
0x10000119c      test r14d, r14d
0x10000119f      jg 0x1000011a6
0x1000011a1      call sym.func.1000043ff
0x1000011a6      lea rsi, [rip + 0x3943]

   ^-  ; section.4.__cstring
0x1000011ad      xor edi, edi
0x1000011af      call sym.imp.setlocale
```

```
     search.to: Search end address
stack.anotated: Show anotated hexdump in visual debug
   stack.bytes: Show bytes instead of words in stack
   stack.delta: Delta for the stack dump
    stack.size: Size in bytes of stack hexdump in visual debug
      time.fmt: Date format (%Y-%m-%d %H:%M:%S %z)
     time.zone: Time zone, in hours relative to GMT: +2, -1,..
     zoom.byte: Zoom callback to calculate each byte (See pz? for help)
     zoom.from: Zoom start address
    zoom.maxsz: Zoom max size of block
```

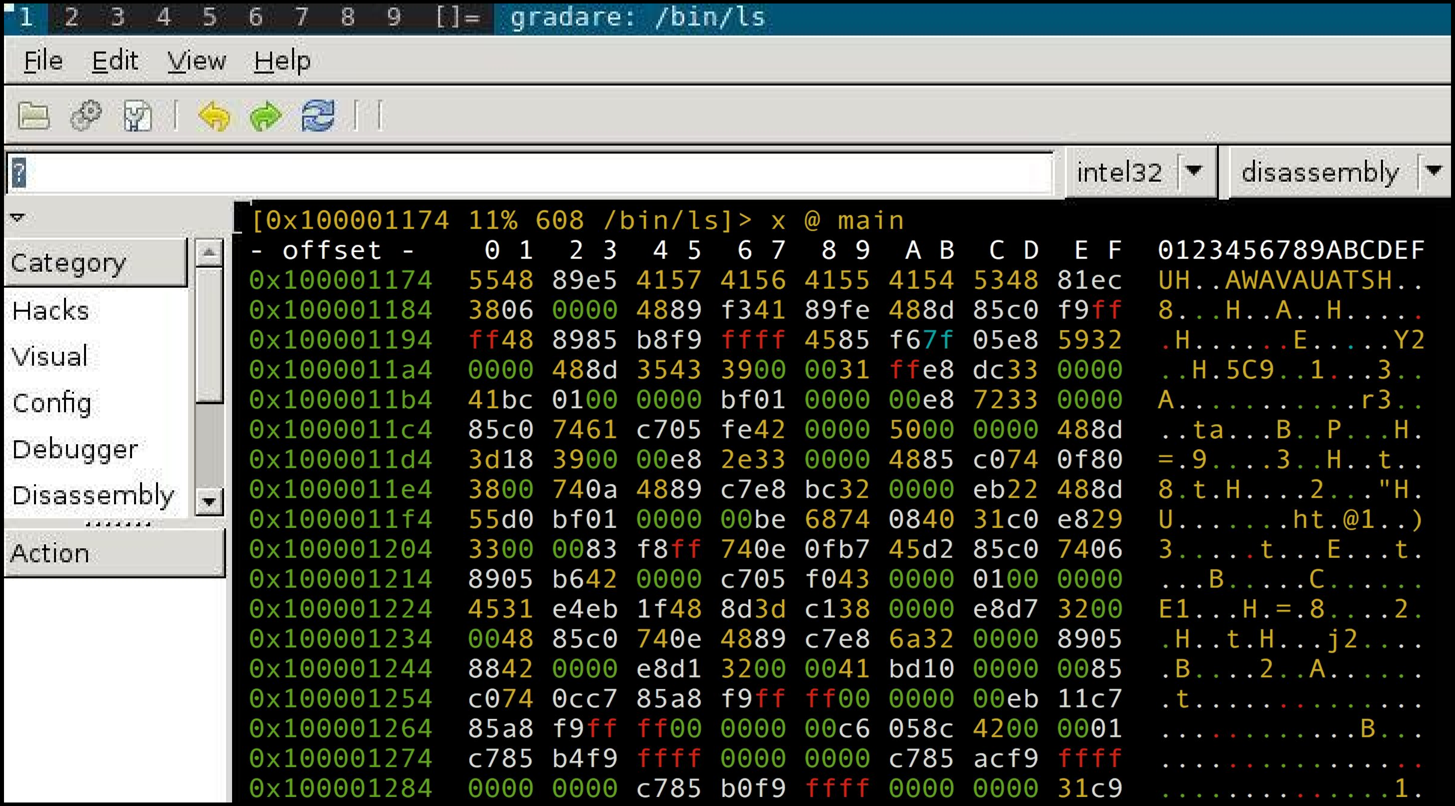
## Solution 3: Native App

I have running versions of r2 on iWatch, unjailbroken iPhones, and wrote a couple of GUIs in GTK and QT. Hey! But there’s also Bokken and Gradare2!

- Boring to develop, lack of widgets and task queue
- Jumping up and down from core to gui is tiring
- Better user experience

# Solution 3: Native App: Gradare2

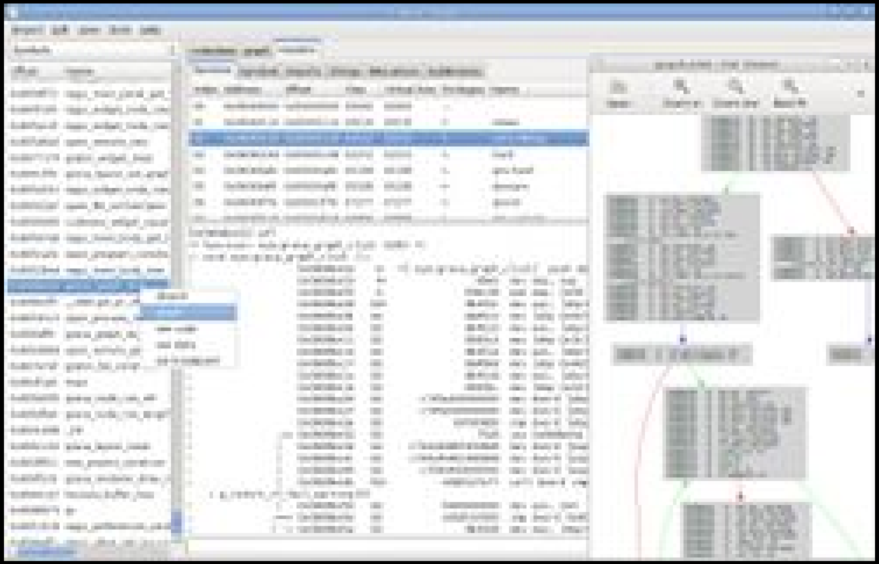
Rewrite of gradare, wip gtk3, vte-based UI



```
[0x100001174 11% 608 /bin/ls]> x @ main
- offset -    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x100001174  5548 89e5 4157 4156 4155 4154 5348 81ec  UH..AWAVAUATSH..
0x100001184  3806 0000 4889 f341 89fe 488d 85c0 f9ff  8...H..A..H.....
0x100001194  ff48 8985 b8f9 ffff 4585 f67f 05e8 5932  .H......E.....Y2
0x1000011a4  0000 488d 3543 3900 0031 ffe8 dc33 0000  ..H.5C9..1...3..
0x1000011b4  41bc 0100 0000 bf01 0000 00e8 7233 0000  A...........r3..
0x1000011c4  85c0 7461 c705 fe42 0000 5000 0000 488d  ..ta...B..P...H.
0x1000011d4  3d18 3900 00e8 2e33 0000 4885 c074 0f80  =.9....3..H..t..
0x1000011e4  3800 740a 4889 c7e8 bc32 0000 eb22 488d  8.t.H....2..."H.
0x1000011f4  55d0 bf01 0000 00be 6874 0840 31c0 e829  U.......ht.@1..)
0x100001204  3300 0083 f8ff 740e 0fb7 45d2 85c0 7406  3.....t...E...t.
0x100001214  8905 b642 0000 c705 f043 0000 0100 0000  ...B.....C......
0x100001224  4531 e4eb 1f48 8d3d c138 0000 e8d7 3200  E1...H.=.8....2.
0x100001234  0048 85c0 740e 4889 c7e8 6a32 0000 8905  .H..t.H...j2....
0x100001244  8842 0000 e8d1 3200 0041 bd10 0000 0085  .B....2..A......
0x100001254  c074 0cc7 85a8 f9ff ff00 0000 00eb 11c7  .t..............
0x100001264  85a8 f9ff ff00 0000 00c6 058c 4200 0001  ............B...
0x100001274  c785 b4f9 ffff 0000 0000 c785 acf9 ffff  ................
0x100001284  0000 0000 c785 b0f9 ffff 0000 0000 31c9  ..............1.
```

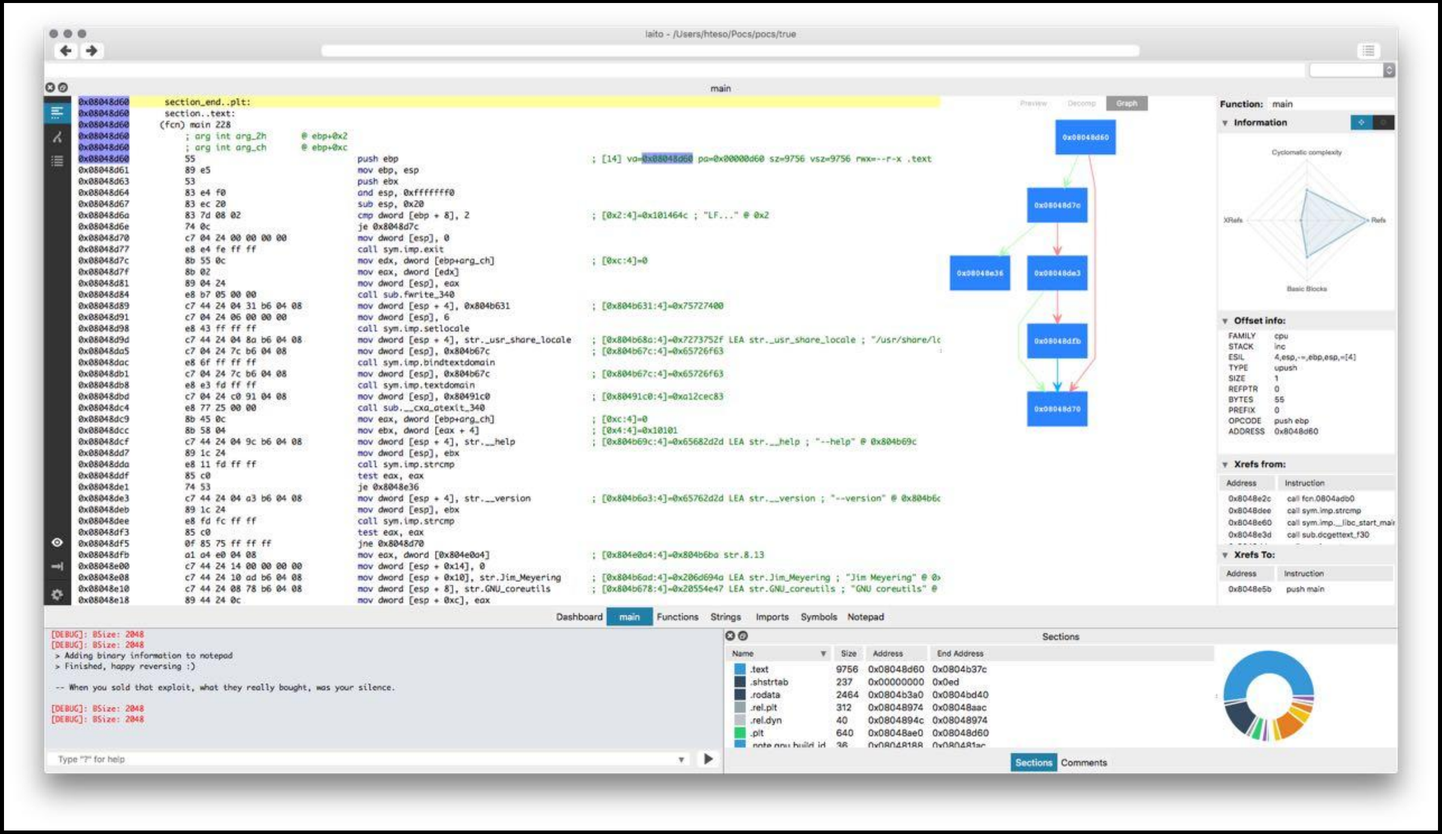
## Solution 3: Native App: Ragui

Ragui - never published full native GUI written in Vala and GtkON 6 years ago.. In the works to be ready for full GTK3



# Solution 3: Native App

$ r2pro



```
0x08048d60        section_end..plt:
0x08048d60        section..text:
0x08048d60       (fcn) main 228
0x08048d60           ; arg int arg_2h       @ ebp+0x2
0x08048d60           ; arg int arg_ch       @ ebp+0xc
0x08048d60           55                     push ebp                                ; [14] va=0x08048d60 pa=0x00000d60 sz=9756 vsz=9756 rwx=--r-x .text
0x08048d61           89 e5                  mov ebp, esp
0x08048d63           53                     push ebx
0x08048d64           83 e4 f0               and esp, 0xfffffff0
0x08048d67           83 ec 20               sub esp, 0x20
0x08048d6a           83 7d 08 02            cmp dword [ebp + 8], 2                  ; [0x2:4]=0x101464c ; "LF..." @ 0x2
0x08048d6e           74 0c                  je 0x8048d7c
0x08048d70           c7 04 24 00 00 00 00   mov dword [esp], 0
0x08048d77           e8 e4 fe ff ff         call sym.imp.exit
0x08048d7c           8b 55 0c               mov edx, dword [ebp+arg_ch]             ; [0xc:4]=0
0x08048d7f           8b 02                  mov eax, dword [edx]
0x08048d81           89 04 24               mov dword [esp], eax
0x08048d84           e8 b7 05 00 00         call sub.fwrite_340
0x08048d89           c7 44 24 04 31 b6 04 08  mov dword [esp + 4], 0x804b631       ; [0x804b631:4]=0x75727400
0x08048d91           c7 04 24 06 00 00 00   mov dword [esp], 6
0x08048d98           e8 43 ff ff ff         call sym.imp.setlocale
0x08048d9d           c7 44 24 04 8a b6 04 08  mov dword [esp + 4], str._usr_share_locale   ; [0x804b68a:4]=0x7273752f LEA str._usr_share_locale ; "/usr/share/lc
0x08048da5           c7 04 24 7c b6 04 08   mov dword [esp], 0x804b67c              ; [0x804b67c:4]=0x65726f63
0x08048dac           e8 6f ff ff ff         call sym.imp.bindtextdomain
0x08048db1           c7 04 24 7c b6 04 08   mov dword [esp], 0x804b67c              ; [0x804b67c:4]=0x65726f63
0x08048db8           e8 e3 fd ff ff         call sym.imp.textdomain
0x08048dbd           c7 04 24 c0 91 04 08   mov dword [esp], 0x80491c0              ; [0x80491c0:4]=0xa12cec83
0x08048dc4           e8 77 25 00 00         call sub.__cxa_atexit_340
0x08048dc9           8b 45 0c               mov eax, dword [ebp+arg_ch]             ; [0xc:4]=0
0x08048dcc           8b 58 04               mov ebx, dword [eax + 4]                ; [0x4:4]=0x10101
0x08048dcf           c7 44 24 04 9c b6 04 08  mov dword [esp + 4], str.__help      ; [0x804b69c:4]=0x65682d2d LEA str.__help ; "--help" @ 0x804b69c
0x08048dd7           89 1c 24               mov dword [esp], ebx
0x08048dda           e8 11 fd ff ff         call sym.imp.strcmp
0x08048ddf           85 c0                  test eax, eax
0x08048de1           74 53                  je 0x8048e36
0x08048de3           c7 44 24 04 a3 b6 04 08  mov dword [esp + 4], str.__version  ; [0x804b6a3:4]=0x65762d2d LEA str.__version ; "--version" @ 0x804b6c
0x08048deb           89 1c 24               mov dword [esp], ebx
0x08048dee           e8 fd fc ff ff         call sym.imp.strcmp
0x08048df3           85 c0                  test eax, eax
0x08048df5           0f 85 75 ff ff ff      jne 0x8048d70
0x08048dfb           a1 a4 e0 04 08         mov eax, dword [0x804e0a4]              ; [0x804e0a4:4]=0x804b6ba str.8.13
0x08048e00           c7 44 24 14 00 00 00 00  mov dword [esp + 0x14], 0
0x08048e08           c7 44 24 10 ad b6 04 08  mov dword [esp + 0x10], str.Jim_Meyering   ; [0x804b6ad:4]=0x206d694a LEA str.Jim_Meyering ; "Jim Meyering" @ 0x
0x08048e10           c7 44 24 08 78 b6 04 08  mov dword [esp + 8], str.GNU_coreutils     ; [0x804b678:4]=0x20554e47 LEA str.GNU_coreutils ; "GNU coreutils" @
0x08048e18           89 44 24 0c            mov dword [esp + 0xc], eax
```

Function: main

Information

Offset info:

| | |
|---|---|
| FAMILY | cpu |
| STACK | inc |
| ESIL | 4,esp,-=,ebp,esp,=[4] |
| TYPE | upush |
| SIZE | 1 |
| REFPTR | 0 |
| BYTES | 55 |
| PREFIX | 0 |
| OPCODE | push ebp |
| ADDRESS | 0x8048d60 |

Xrefs from:

| Address | Instruction |
|---|---|
| 0x8048e2c | call fcn.0804adb0 |
| 0x8048dee | call sym.imp.strcmp |
| 0x8048e60 | call sym.imp.__libc_start_mai |
| 0x8048e3d | call sub.dcgettext_f30 |

Xrefs To:

| Address | Instruction |
|---|---|
| 0x8048e5b | push main |

```
[DEBUG]: BSize: 2048
[DEBUG]: BSize: 2048
 > Adding binary information to notepad
 > Finished, happy reversing :)

 -- When you sold that exploit, what they really bought, was your silence.

[DEBUG]: BSize: 2048
[DEBUG]: BSize: 2048
```

Type "?" for help

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 |
| 🟦 .note.gnu.build.id | 36 | 0x08048188 | 0x080481ac |

## Questions?