Hidden Gems In r2land

pancake@nopcode.org

Explorations

Radare is a pretty big project with lot of tiny features and tricks that few people know in detail.

- This is what makes learning in **r2land** fun.

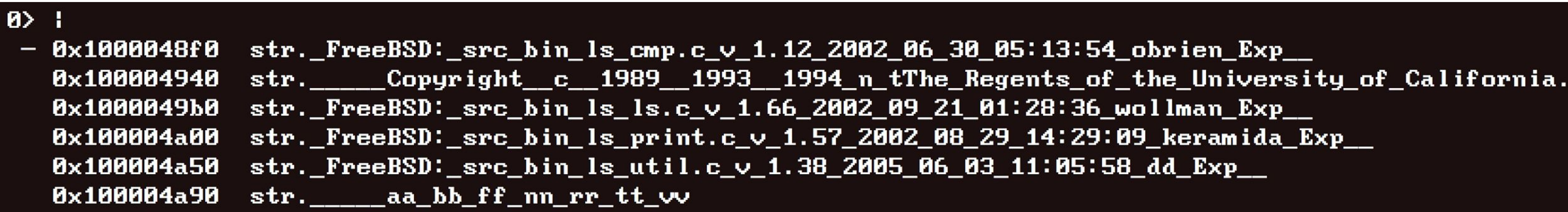
This talk will show a bunch of those barely used commands and features that may help you or enlight you into some other concepts while learning r2.

The Hud

The HUD is an visual autocompletion interface to select items from a list.

- V_
- ~...

Note that this is an extra dot over the internal less ~..



```
0> ¦
 - 0x1000048f0  str._FreeBSD:_src_bin_ls_cmp.c_v_1.12_2002_06_30_05:13:54_obrien_Exp__
   0x100004940  str._____Copyright__c__1989__1993__1994_n_tThe_Regents_of_the_University_of_California.
   0x1000049b0  str._FreeBSD:_src_bin_ls_ls.c_v_1.66_2002_09_21_01:28:36_wollman_Exp__
   0x100004a00  str._FreeBSD:_src_bin_ls_print.c_v_1.57_2002_08_29_14:29:09_keramida_Exp__
   0x100004a50  str._FreeBSD:_src_bin_ls_util.c_v_1.38_2005_06_03_11:05:58_dd_Exp__
   0x100004a90  str._____aa_bb_ff_nn_rr_tt_vv
```

Analysis SubCommands

There are a lot of analysis options and commands that may help you experiment with your target of choice. It is important to know them and choose wisely instead of just throwing aaaaa to the shell.

- aab
- aav
- aar
- aae

Analysis Options

Check `e??anal.`

- anal.hasnext
- anal.a2f
- anal.jmptbl
- anal.strings

Visual Tiled Panels

The VIsual tiled panels is a **dwm**-like interface on top of RCons and RCanvas, providing auto-layout panels that are filled with the output of a command.

- V!

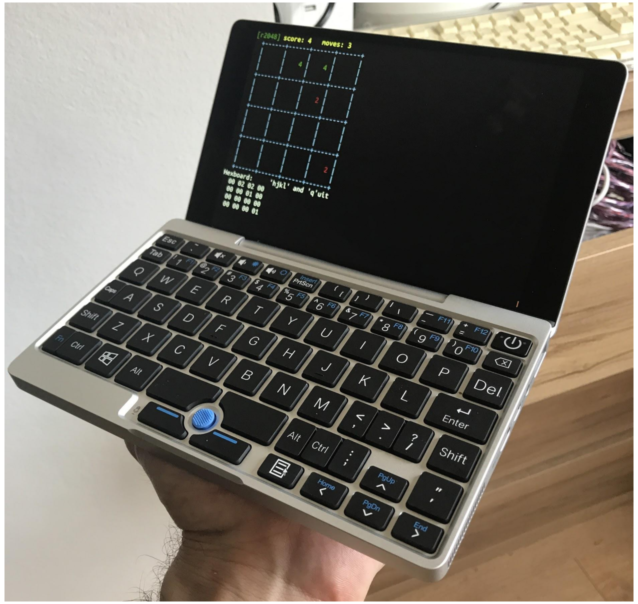
The lack of users complaining stuck the development but it's easy to extend and tweak to make it work for you if you are a tiled window manager user

2048

The famous game is also included!

Some time ago it was hidden inside an undocumented key in visual mode so users start it without noticing and causing a major productivity drop to the analyst.

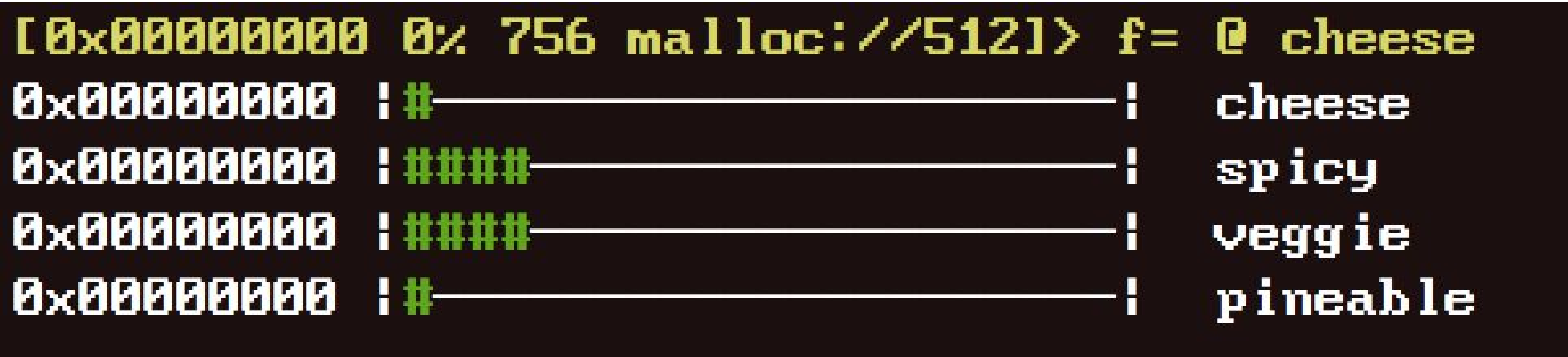
Now it's available in the top menu of the visual panels mode.



```
[r2048] score: 4   moves: 3
+------+------+------+------+
|      |     4|     4|      |
+------+------+------+------+
|      |      |     2|      |
+------+------+------+------+
|      |      |      |      |
+------+------+------+------+
|      |      |      |     2|
+------+------+------+------+
Hexboard:
 00 02 02 00     'hjkl' and 'q'uit
 00 00 01 00
 00 00 00 00
 00 00 00 01
```

Ordering pizzas

Yeah the legend it's true. Once, in a private con with some friends we did the voting of pizzas by using r2 to show the stats of each pizza. See f=

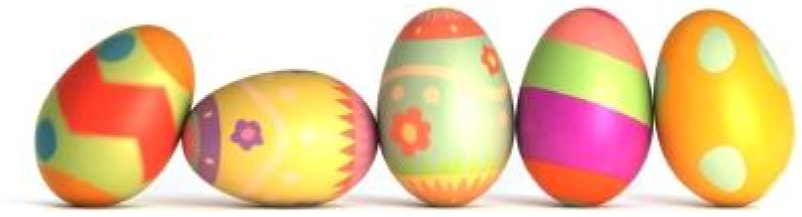


```
[0x00000000 0% 756 malloc://512]> f= @ cheese
0x00000000 |#-------------------------------------------| cheese
0x00000000 |#####---------------------------------------| spicy
0x00000000 |#####---------------------------------------| veggie
0x00000000 |#-------------------------------------------| pineable
```

Easter Eggs

There have been bunch of easter eggs in r2, they are all removed as soon as someone makes it public.

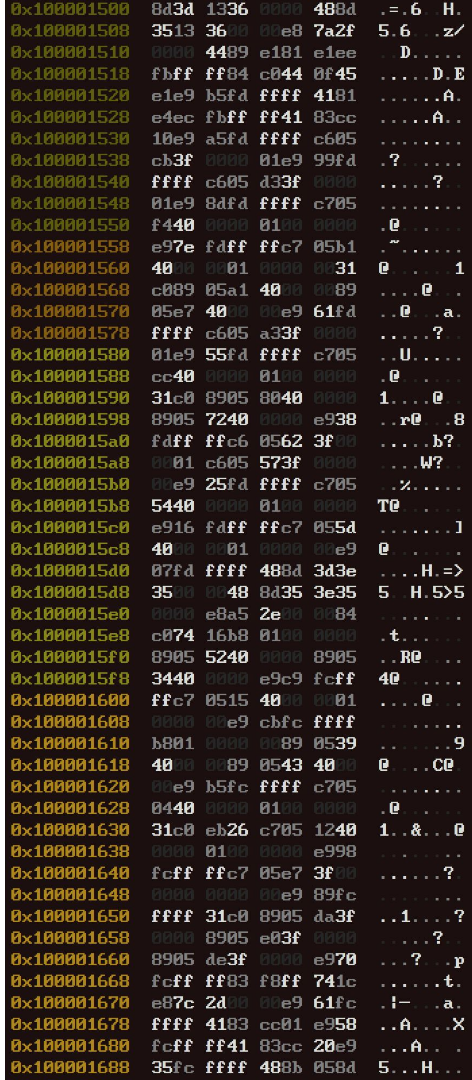
- May you find any in current master?



scr.rainbow

An experimental option makes the offset of the instruction in hexdump and disasm to be painted with a fixed color that depends on the offset itself.

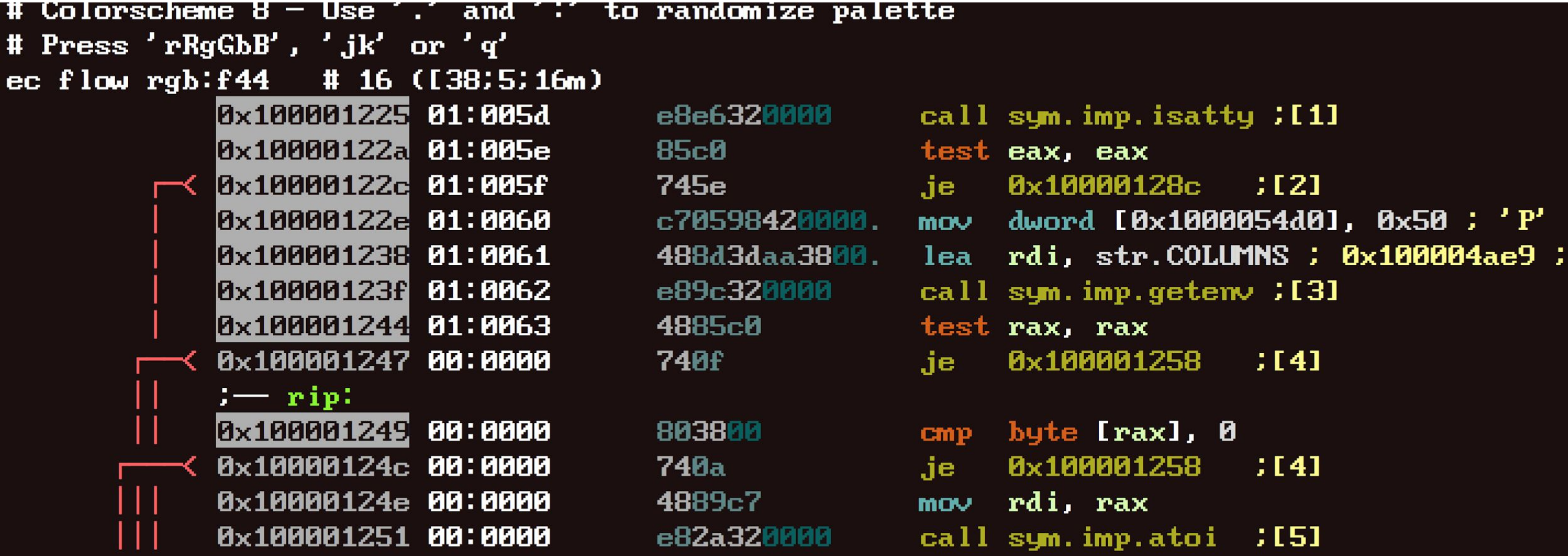
This way the reader can memorize a region by its color, not just its name or context.



```
0x100001500  8d3d 1336 0000 488d  .=.6..H.
0x100001508  3513 3600 00e8 7a2f  5.6...z/
0x100001510  0000 4489 e181 e1ee  ..D.....
0x100001518  fbff ff84 c044 0f45  .....D.E
0x100001520  e1e9 b5fd ffff 4181  ......A.
0x100001528  e4ec fbff ff41 83cc  .....A..
0x100001530  10e9 a5fd ffff c605  ........
0x100001538  cb3f 0000 01e9 99fd  .?......
0x100001540  ffff c605 d33f 0000  .....?..
0x100001548  01e9 8dfd ffff c705  ........
0x100001550  f440 0000 0100 0000  .@......
0x100001558  e97e fdff ffc7 05b1  .~......
0x100001560  4000 0001 0000 0031  @......1
0x100001568  c089 05a1 4000 0089  ....@...
0x100001570  05e7 4000 00e9 61fd  ..@...a.
0x100001578  ffff c605 a33f 0000  .....?..
0x100001580  01e9 55fd ffff c705  ..U.....
0x100001588  cc40 0000 0100 0000  .@......
0x100001590  31c0 8905 8040 0000  1....@..
0x100001598  8905 7240 0000 e938  ..r@...8
0x1000015a0  fdff ffc6 0562 3f00  .....b?.
0x1000015a8  0001 c605 573f 0000  ....W?..
0x1000015b0  00e9 25fd ffff c705  ..%.....
0x1000015b8  5440 0000 0100 0000  T@......
0x1000015c0  e916 fdff ffc7 055d  .......]
0x1000015c8  4000 0001 0000 00e9  @.......
0x1000015d0  07fd ffff 488d 3d3e  ....H.=>
0x1000015d8  3500 0048 8d35 3e35  5..H.5>5
0x1000015e0  0000 e8a5 2e00 0084  ........
0x1000015e8  c074 16b8 0100 0000  .t......
0x1000015f0  8905 5240 0000 8905  ..R@....
0x1000015f8  3440 0000 e9c9 fcff  4@......
0x100001600  ffc7 0515 4000 0001  ....@...
0x100001608  0000 00e9 cbfc ffff  ........
0x100001610  b801 0000 0089 0539  .......9
0x100001618  4000 0089 0543 4000  @....C@.
0x100001620  00e9 b5fc ffff c705  ........
0x100001628  0440 0000 0100 0000  .@......
0x100001630  31c0 eb26 c705 1240  1..&...@
0x100001638  0000 0100 0000 e998  ........
0x100001640  fcff ffc7 05e7 3f00  ......?.
0x100001648  0000 0000 00e9 89fc  ........
0x100001650  ffff 31c0 8905 da3f  ..1....?
0x100001658  0000 8905 e03f 0000  .....?..
0x100001660  8905 de3f 0000 e970  ...?...p
0x100001668  fcff ff83 f8ff 741c  ......t.
0x100001670  e87c 2d00 00e9 61fc  .|-...a.
0x100001678  ffff 4183 cc01 e958  ..A....X
0x100001680  fcff ff41 83cc 20e9  ...A.. .
0x100001688  35fc ffff 488b 058d  5...H...
```

Theme Editor

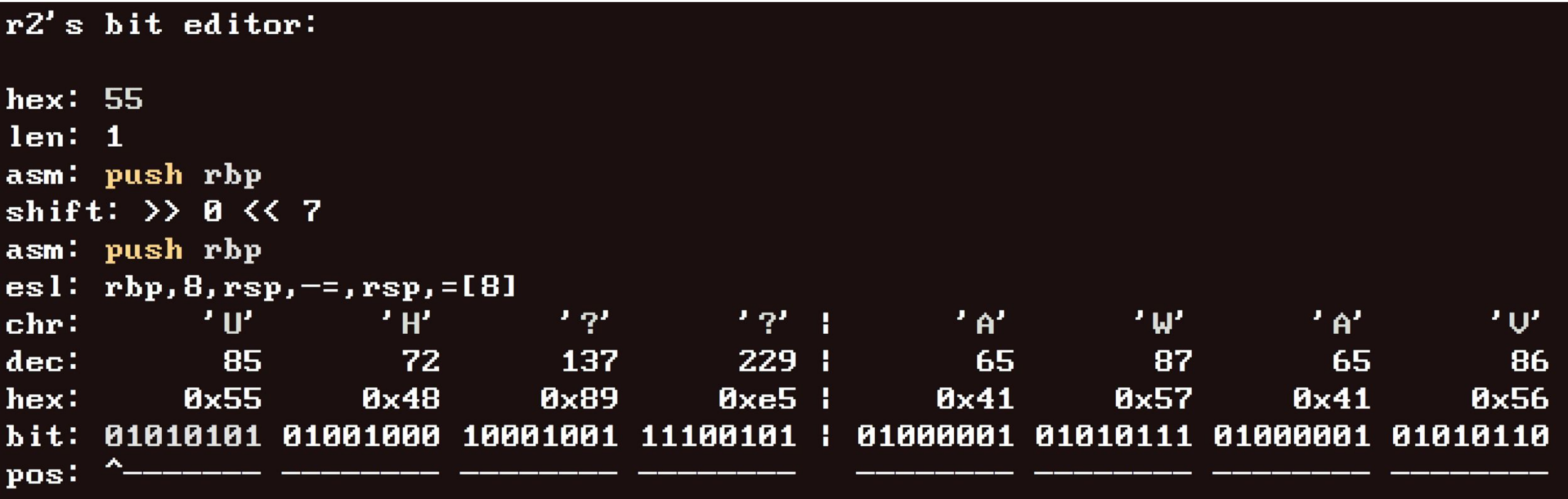
VE



```
# Colorscheme 8 - Use '.' and ':' to randomize palette
# Press 'rRgGbB', 'jk' or 'q'
ec flow rgb:f44    # 16 ([38;5;16m)
            0x100001225 01:005d      e8e6320000      call sym.imp.isatty ;[1]
            0x10000122a 01:005e      85c0            test eax, eax
        ┌─< 0x10000122c 01:005f      745e            je   0x10000128c    ;[2]
        │   0x10000122e 01:0060      c70598420000.   mov  dword [0x1000054d0], 0x50 ; 'P'
        │   0x100001238 01:0061      488d3daa3800.   lea  rdi, str.COLUMNS ; 0x100004ae9 ;
        │   0x10000123f 01:0062      e89c320000      call sym.imp.getenv ;[3]
        │   0x100001244 01:0063      4885c0          test rax, rax
       ┌──< 0x100001247 00:0000      740f            je   0x100001258    ;[4]
       ││   ;-- rip:
       ││   0x100001249 00:0000      803800          cmp  byte [rax], 0
      ┌───< 0x10000124c 00:0000      740a            je   0x100001258    ;[4]
      │││   0x10000124e 00:0000      4889c7          mov  rdi, rax
      │││   0x100001251 00:0000      e82a320000      call sym.imp.atoi   ;[5]
```

Visual Bit editor

> Vd1

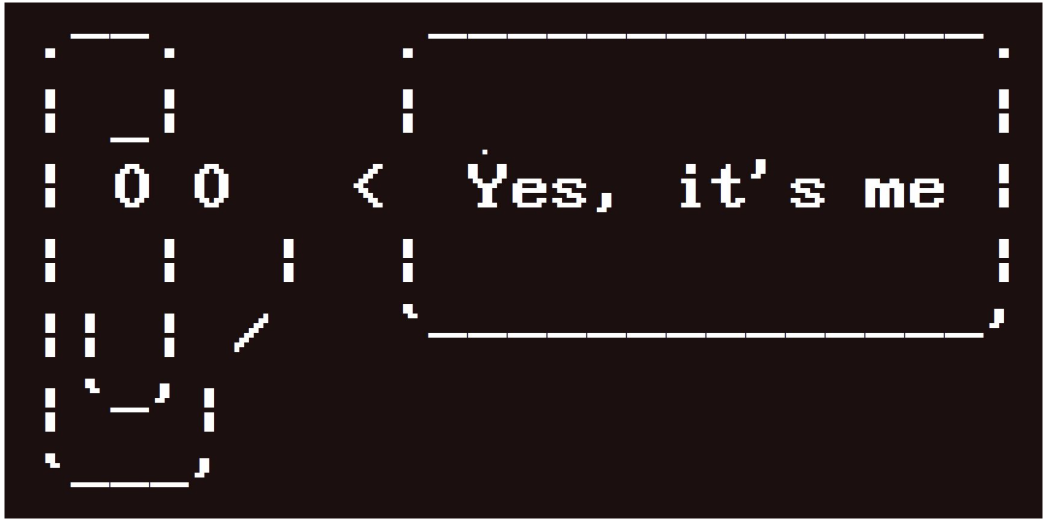


```
r2's bit editor:

hex: 55
len: 1
asm: push rbp
shift: >> 0 << 7
asm: push rbp
esl: rbp,8,rsp,-=,rsp,=[8]
chr:        'U'       'H'       '?'       '?'  |      'A'       'W'       'A'       'V'
dec:         85        72       137       229  |       65        87        65        86
hex:       0x55      0x48      0x89      0xe5  |     0x41      0x57      0x41      0x56
bit:   01010101  01001000  10001001  11100101  | 01000001  01010111  01000001  01010110
pos:   ^-------  --------  --------  --------    --------  --------  --------  --------
```

Clippy

?E



```
.__.     .______________.
| _|     |              |
| O O   <  Yes, it's me |
|  |  |  |              |
||  | /  `______________'
|`_'|
`___'
```

Visual Browsers

Walk around

- Classes / methods
- Flagspaces / flags
- Types / struct / enums..

By typing

- VF, VB, VT

r2pm

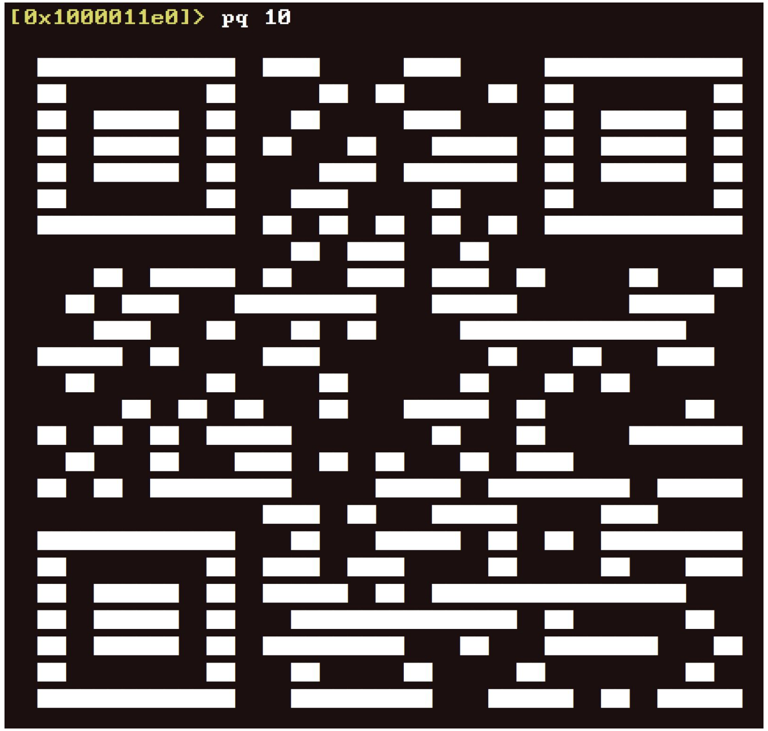
A package manager that ships with r2.

- Allows to install and update r2 from git
- Supports home and system-wide installations
- Builds and installs plugins
- Many tools, aims to be portable (across *nixes)

QR codes

The `pq` command will print a QR code containing N bytes (in this case 10) from the current seek.

Generate binary QR codes, not just strings at any size!



```
[0x1000011e0]> pq 10
```

Aliases and Macros

You can use aliases with $ to specify a list of commands to be run when the alias is typed.

- $foo=x

Also, macros add some extra logic allowing them to accept arguments.

- (Foo name,f $0=$$+rax)
- .(Foo test)

Foreach

The @ sign allows us to perform temporary seeks

- But we have @@ and @@@

Then we can do:

- aoj @@i
- pdc @@@ functions
- wx 90 @@/x 80

JSON processing

It is well known that most commands return JSON when the last char is a ‘j’. As well as we have an internal less, there’s also a json parser and indenter.

- **aoj~{}**
- **lj~{core.file}**
- **afij~{[0].offset}**

Interpreting the output as r2 commands

Prefixing any command with a dot results in running each line as commands typed in the prompt.

Useful for writing generator scripts or programs

Anal hints

Accessible with the **ah** command, it allows us to override analysis information for a specific instruction.

- ahb 16 # force thumb
- ahi s # argument is converted into a string
- ahi 10 # change immediate to base 10
- ahe 0,r0,= # override esil expression
- ...

Syscall database

R2 ships a syscall database that contains:

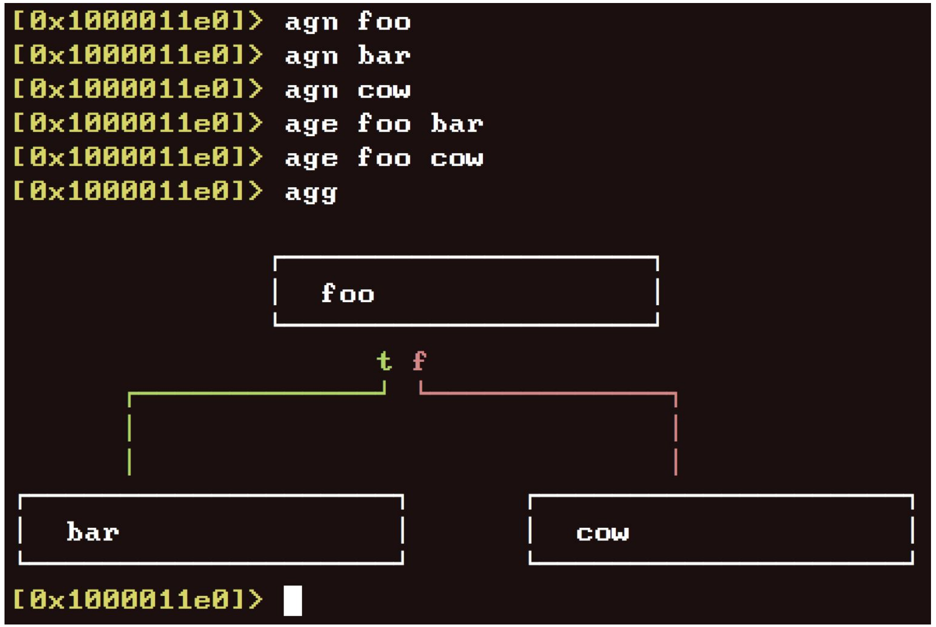
- Name
- Number (usually in A0 (rax, ...))
- Signature (return type and arguments)

Depends on:

- e asm.arch / asm.bits / asm.os

Doing your own graphs

The **ag** subcommands can be used to create a graph with your own info, then print it in ascii art, graphviz, etc..



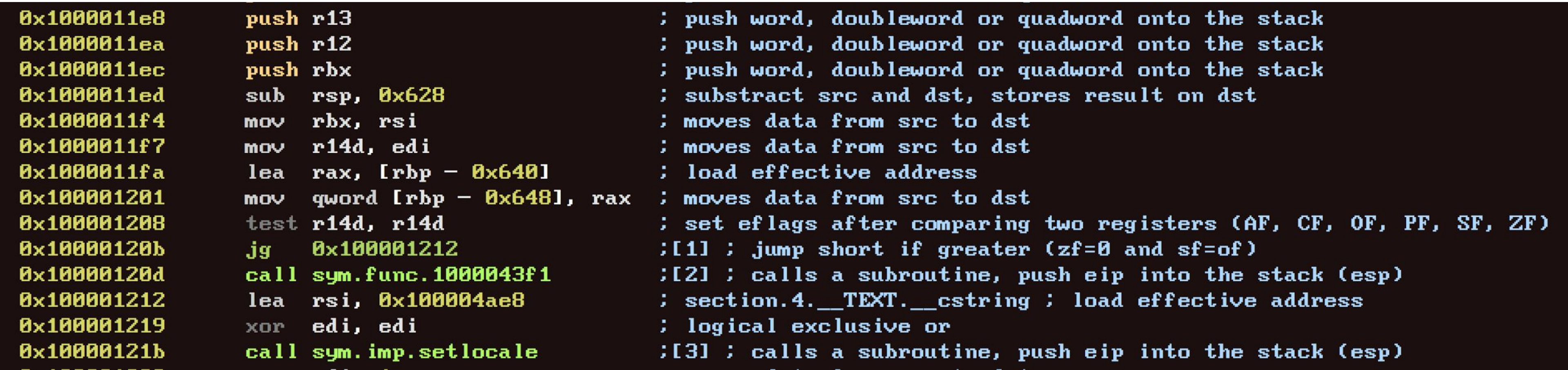
```
[0x1000011e0]> agn foo
[0x1000011e0]> agn bar
[0x1000011e0]> agn cow
[0x1000011e0]> age foo bar
[0x1000011e0]> age foo cow
[0x1000011e0]> agg

              .--------------------.
              |  foo               |
              `--------------------'
                      t f
       .--------------' '-------------.
       |                              |
       |                              |
 .--------------------.      .--------------------.
 |  bar               |      |  cow               |
 `--------------------'      `--------------------'
[0x1000011e0]>
```

Opcode Descriptions

When you don’t know what a specific instruction does you can enable the opcode descriptions to get some help.

> e asm.describe=true

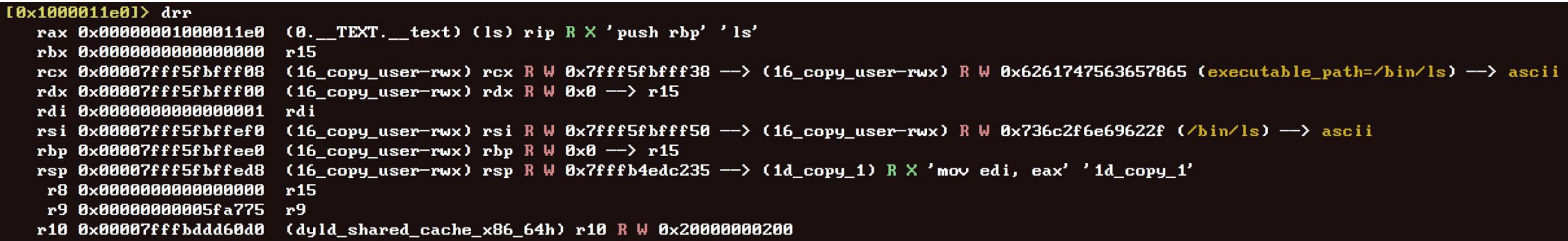


```
0x1000011e8      push r13                  ; push word, doubleword or quadword onto the stack
0x1000011ea      push r12                  ; push word, doubleword or quadword onto the stack
0x1000011ec      push rbx                  ; push word, doubleword or quadword onto the stack
0x1000011ed      sub  rsp, 0x628           ; substract src and dst, stores result on dst
0x1000011f4      mov  rbx, rsi             ; moves data from src to dst
0x1000011f7      mov  r14d, edi            ; moves data from src to dst
0x1000011fa      lea  rax, [rbp - 0x640]   ; load effective address
0x100001201      mov  qword [rbp - 0x648], rax  ; moves data from src to dst
0x100001208      test r14d, r14d           ; set eflags after comparing two registers (AF, CF, OF, PF, SF, ZF)
0x10000120b      jg   0x100001212          ;[1] ; jump short if greater (zf=0 and sf=of)
0x10000120d      call sym.func.1000043f1   ;[2] ; calls a subroutine, push eip into the stack (esp)
0x100001212      lea  rsi, 0x100004ae8     ; section.4.__TEXT.__cstring ; load effective address
0x100001219      xor  edi, edi             ; logical exclusive or
0x10000121b      call sym.imp.setlocale    ;[3] ; calls a subroutine, push eip into the stack (esp)
```

Telescoping

Telescoping is a way to go deep into a value to determine the contents of that memory address that points.

- drr -> telescoping debug registers
- pxr -> telescoping memory (@r:SP)



```
[0x1000011e0]> drr
   rax 0x00000001000011e0  (0.__TEXT.__text) (ls) rip R X 'push rbp' 'ls'
   rbx 0x0000000000000000  r15
   rcx 0x00007fff5fbfff08  (16_copy_user-rwx) rcx R W 0x7fff5fbfff38 --> (16_copy_user-rwx) R W 0x6261747563657865 (executable_path=/bin/ls) --> ascii
   rdx 0x00007fff5fbfff00  (16_copy_user-rwx) rdx R W 0x0 --> r15
   rdi 0x0000000000000001  rdi
   rsi 0x00007fff5fbffef0  (16_copy_user-rwx) rsi R W 0x7fff5fbfff50 --> (16_copy_user-rwx) R W 0x736c2f6e69622f (/bin/ls) --> ascii
   rbp 0x00007fff5fbffee0  (16_copy_user-rwx) rbp R W 0x0 --> r15
   rsp 0x00007fff5fbffed8  (16_copy_user-rwx) rsp R W 0x7fffb4edc235 --> (1d_copy_1) R X 'mov edi, eax' '1d_copy_1'
    r8 0x0000000000000000  r15
    r9 0x00000000005fa775  r9
   r10 0x00007fffbddd60d0  (dyld_shared_cache_x86_64h) r10 R W 0x20000000200
```

Emulation

Initialize with aeim, aeip, aeis,...

- aesu 0x100001208

Better commands in disasm references

- e asm.emustr=true

LIKELY / UNLIKELY

Tracing

- e asm.trace = true
- e dbg.trace = true

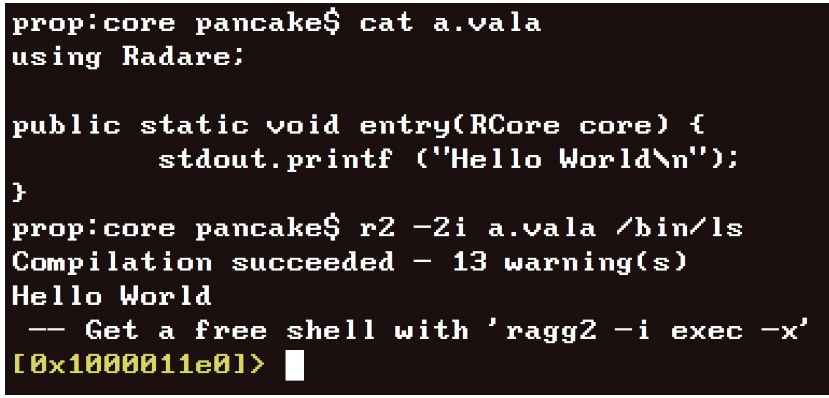
We can then go into visual and step a bit

- Dt - show traces
- Dtd - show disasm of each instruction traced

Scripting in C and Vala

Using the #!c we can inject code into r2 and use the api itself by just having the instance to core.

- r2 -qi test.vala /bin/ls



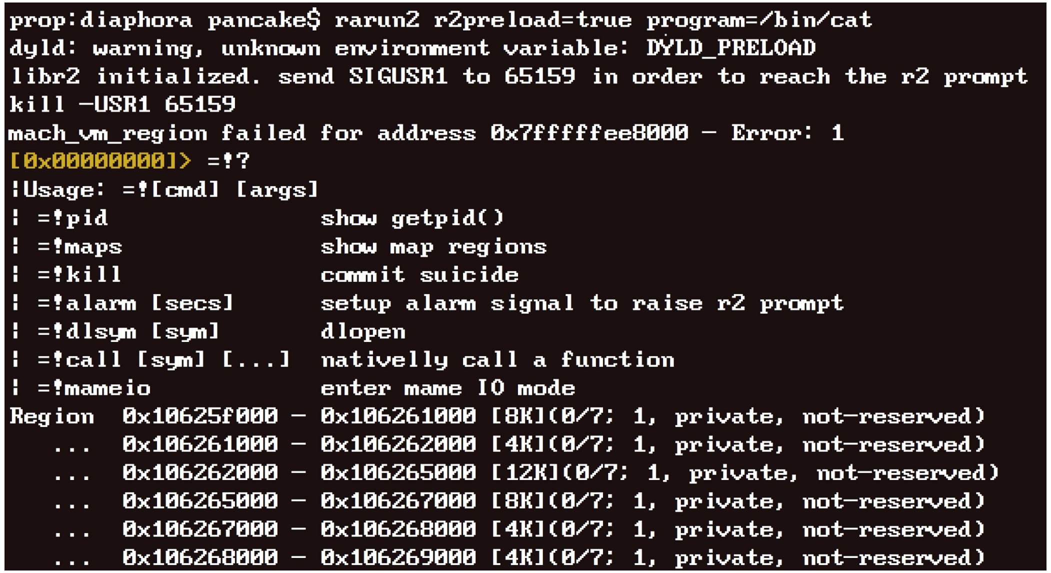
```
prop:core pancake$ cat a.vala
using Radare;

public static void entry(RCore core) {
        stdout.printf ("Hello World\n");
}
prop:core pancake$ r2 -2i a.vala /bin/ls
Compilation succeeded - 13 warning(s)
Hello World
 -- Get a free shell with 'ragg2 -i exec -x'
[0x1000011e0]>
```

Vapi files makes harder to make mistakes by passing invalid parameters to a function.

r2preload

- rarun2 can LD_PRELOAD r2 inside new processes
- We can use r2frida to do that too.



```
prop:diaphora pancake$ rarun2 r2preload=true program=/bin/cat
dyld: warning, unknown environment variable: DYLD_PRELOAD
libr2 initialized. send SIGUSR1 to 65159 in order to reach the r2 prompt
kill -USR1 65159
mach_vm_region failed for address 0x7ffffee8000 - Error: 1
[0x00000000]> =!?
|Usage: =![cmd] [args]
| =!pid                show getpid()
| =!maps               show map regions
| =!kill               commit suicide
| =!alarm [secs]       setup alarm signal to raise r2 prompt
| =!dlsym [sym]        dlopen
| =!call [sym] [...]   natively call a function
| =!mameio             enter mame IO mode
Region   0x10625f000 - 0x106261000 [8K](0/7; 1, private, not-reserved)
   ...   0x106261000 - 0x106262000 [4K](0/7; 1, private, not-reserved)
   ...   0x106262000 - 0x106265000 [12K](0/7; 1, private, not-reserved)
   ...   0x106265000 - 0x106267000 [8K](0/7; 1, private, not-reserved)
   ...   0x106267000 - 0x106268000 [4K](0/7; 1, private, not-reserved)
   ...   0x106268000 - 0x106269000 [4K](0/7; 1, private, not-reserved)
```

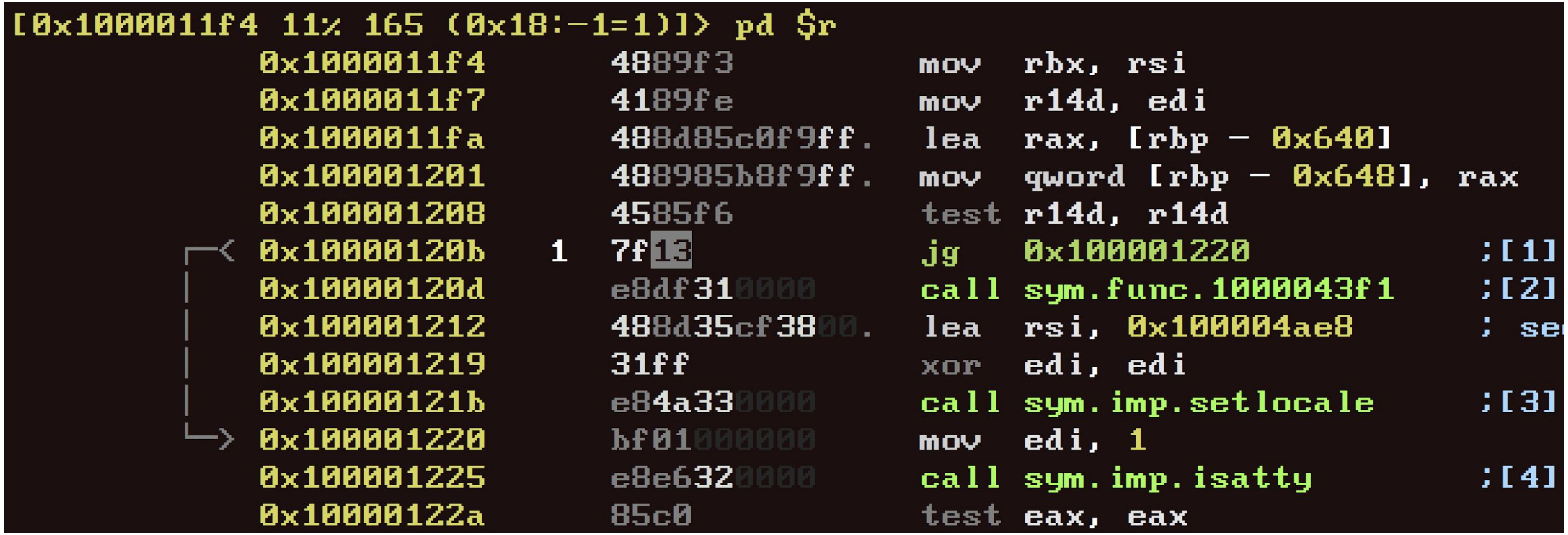
Busybox

r2 shell comes with some unix shell commands that may be useful in some situations, like embedded systems or for portability in scripts.

- Ls
- Rm
- Cp
- ...

Patching Jumps with cursor

Vp and then hjkl use +- to change the value of the byte



```
[0x1000011f4 11% 165 (0x18:-1=1)]> pd $r
            0x1000011f4      4889f3         mov   rbx, rsi
            0x1000011f7      4189fe         mov   r14d, edi
            0x1000011fa      488d85c0f9ff.  lea   rax, [rbp - 0x640]
            0x100001201      488985b8f9ff.  mov   qword [rbp - 0x648], rax
            0x100001208      4585f6         test  r14d, r14d
        ┌─< 0x10000120b    1 7f13           jg    0x100001220          ;[1]
        │   0x10000120d      e8df310000     call  sym.func.1000043f1   ;[2]
        │   0x100001212      488d35cf3800.  lea   rsi, 0x100004ae8     ; se
        │   0x100001219      31ff           xor   edi, edi
        │   0x10000121b      e84a330000     call  sym.imp.setlocale    ;[3]
        └─> 0x100001220      bf01000000     mov   edi, 1
            0x100001225      e8e6320000     call  sym.imp.isatty       ;[4]
            0x10000122a      85c0           test  eax, eax
```

omfg

There are some commands in r2 that may sound fun.

- wtf
- lol
- eta
- wen
- ...

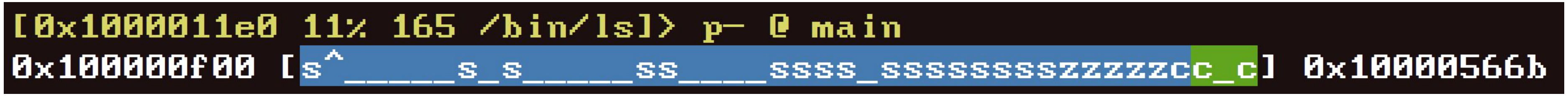
Data statistics

p=?

- Entropy
- Number of printable chars
- Count 0x00 or 0xff in block
- Split the whole area in blocks of size/nblocks
- Number of call instructions
- Number of flags
- ...

Navigation Bar

p-



```
[0x1000011e0 11% 165 /bin/ls]> p- @ main
0x100000f00 [s^______s_s______ss_____ssss_sssssssssZZZZZcc_c] 0x10000566b
```

Custom Visual modes

Press = or | in Visual mode to add commands to run on top or in one side of the visual mode

- e cmd.visual

External windows

Wake up the webserver in background

- =h&

In another terminal run

- r2 -C http://localhost:9999

Egg Compiler And Tiny Bins

Do you need to inject a relocatable snippet written in C .. or in ragg lang?

- Ragg2-cc test.c
- Ragg2 -F hello.r

dbg.slow

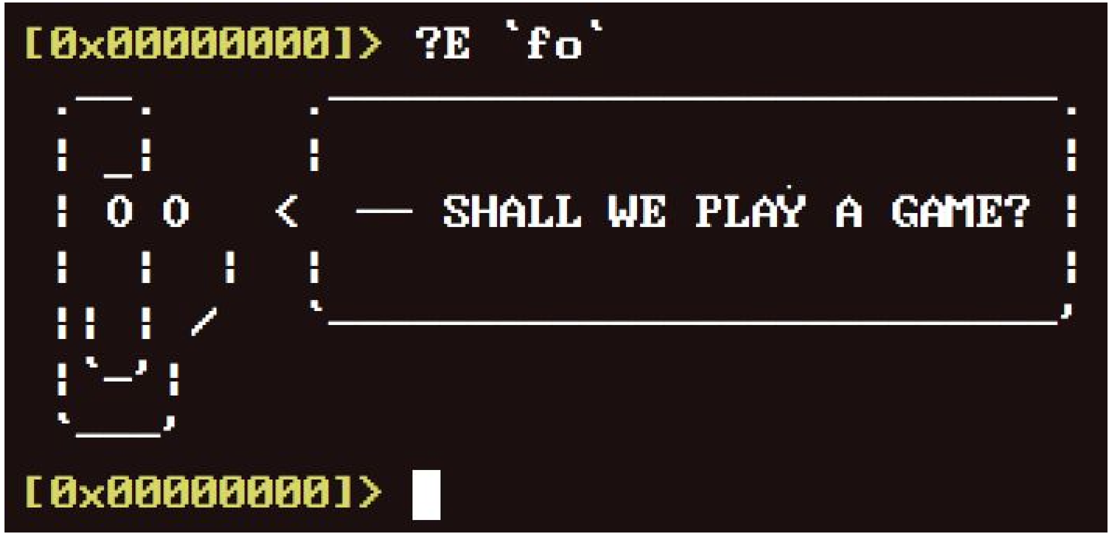
Makes the visual mode slower, but fancier

- Adds telescoping for registers and stack
- Shows backtrace
- Enables esil and extra mem reference reads in disasm

Slower because implies more read operations on the target

Offensive Fortunes

- creepy
- fun
- nsfw
- tips



```
[0x00000000]> ?E `fo`
 .--.     .---------------------------------------------.
 | _|     |                                             |
 | O O   <  --  SHALL WE PLAY A GAME?                   |
 |  |  |  |                                             |
 || | /   `---------------------------------------------'
 |`-'|
 `---'
[0x00000000]>
```

ShellCodes

Ragg2 can generate payloads, fill them with patterns, traps, nops, ...

- ragg2 -p n10 -n 0x1234
- 9090909090909090909034120000

HTML output

When **scr.html=true** is enabled, r2 will output in HTML instead of ANSI, this is handy for piping to a file and including a snippet of your session into a web page.

r2pipe

Simplest interface to script r2, single function (run a command and get output, optionally parsed as json).

- Spawn (fork)
- Pipe (current r2)
- Http (remote r2 webserver)
- Native (dlopen)

r2pipe.js

- r2pipe
  - Sync
  - Async
- r2pipe-promise
  - Uses js promises

Supports multiple targets

- Http, pipe-spawn, tcp...
- More can be added

Native r2pipe

Runs the RCore api by resolving the needed methods to implement an r2pipe interface.

- The fastest r2pipe
- Segfaults may crash your app

r_core_new, r_core_cmd_str, r_core_free

- Check the newlisp implementation

Questions?