

Radare2 workshop

Спикеры



- Антон Кочков (xvilka)
 - Код безопасности
- Борис Рютин (dukebarman)
 - Цифровое оружие и защита (Esagelab)
- Отдельное спасибо Julien Voisin(jvoisin) и Maxim Morine (maiijn)

rasm2

```
darkair:~ dukebarman$ rasm2 -a x86 nop
90
darkair:~ dukebarman$ rasm2 -a x86 -d 'eb00'
jmp 2
darkair:~ dukebarman$ rasm2 -w cmpsb
cmp DS:[SI], ES:[edi] (esi++, edi++)
darkair:~ dukebarman$ rasm2 -w sqrtpd
compute square roots of packed double-fp values
darkair:~ dukebarman$ █
```

Команды в radare2

```
1. radare2
darkair:~ dukebarman$ r2 Documents/Work/animals/7eb449a0be9f008bee337c8d55ba921c
-- Rename a function using the 'afr <newname> @ <offset>' command.
[0x00432e50]> ?
Usage: [.] [times] [cmd] [~grep] [@[iter]addr!size] [!>pipe] ; ...
Append '?' to any char command to get detailed help
Prefix with number to repeat command N times (f.ex: 3x)
| %var =valueAlias for 'env' command
| *off[=[0x]value] Pointer read/write data/values (see ?v, wx, wv)
| (macro arg0 arg1) Manage scripting macros
| .[-!(m)|f!|sh|cmd] Define macro or load r2, cparse or rlang file
| = [cmd] Run this command via rap://
| / Search for bytes, regexps, patterns, ..
| ! [cmd] Run given command as in system(3)
| # [algo] [len] Calculate hash checksum of current block
| a Perform analysis of code
| b Get or change block size
| c [arg] Compare block with given data
| C Code metadata management
| d Debugger commands
| e [a=[b]] List/get/set config evaluable vars
| f [name][sz][at] Set flag at current address
| g [arg] Go compile shellcodes with r_egg
| i [file] Get info about opened file
| k [sdb-query] Run sdb-query. see k? for help, 'k *', 'k **' ...
| m Mountpoints commands
| o [file] ([offset]) Open file at optional address
| p [len] Print current block with format and length
| P Project management utilities
| q [ret] Quit program with a return value
| r [len] Resize file
| s [addr] Seek to address (also for '0x', '0x1' == 's 0x1')
| S Io section manipulation information
| t Cparse types management
| T [-] [num|msg] Text log utility
| V Enter visual mode (vcmds=visualvisual keystrokes)
| w [str] Multiple write operations
| x [len] Alias for 'px' (print hexadecimal)
| y [len] [[[@]addr] Yank/paste bytes from/to memory
| z Signatures management
| ?[??][expr] Help or evaluate math expression
| ?? Show available '$' variables and aliases
| ?@ Misc help for '@' (seek), '~' (grep) (see ~??)
[0x00432e50]> █
```

Быстрая обработка строк (iz, izz, rabin2 -h)

```
1. radare2
vaddr=0x0041e278 paddr=0x0001c878 ordinal=302 sz=7 len=6 section=.rdata type=a string=SaveDC
vaddr=0x0041e282 paddr=0x0001c882 ordinal=303 sz=7 len=6 section=.rdata type=a string=BitBlt
vaddr=0x0041e28c paddr=0x0001c88c ordinal=304 sz=13 len=12 section=.rdata type=a string=DeleteObject
vaddr=0x0041e29c paddr=0x0001c89c ordinal=305 sz=19 len=18 section=.rdata type=a string=CreatePatternBrush
vaddr=0x0041e2b2 paddr=0x0001c8b2 ordinal=306 sz=11 len=10 section=.rdata type=a string=GetClipRgn
vaddr=0x0041e2c0 paddr=0x0001c8c0 ordinal=307 sz=17 len=16 section=.rdata type=a string=GetViewportOrgEx
vaddr=0x0041e2d4 paddr=0x0001c8d4 ordinal=308 sz=15 len=14 section=.rdata type=a string=GetStockObject
vaddr=0x0041e2e6 paddr=0x0001c8e6 ordinal=309 sz=16 len=15 section=.rdata type=a string=SetPolyFillMode
vaddr=0x0041e2f8 paddr=0x0001c8f8 ordinal=310 sz=17 len=16 section=.rdata type=a string=CreateSolidBrush
vaddr=0x0041e30c paddr=0x0001c90c ordinal=311 sz=10 len=9 section=.rdata type=a string=SetBkMode
vaddr=0x0041e318 paddr=0x0001c918 ordinal=312 sz=10 len=9 section=.rdata type=a string=RestoreDC
vaddr=0x0041e324 paddr=0x0001c924 ordinal=313 sz=14 len=13 section=.rdata type=a string=GetDeviceCaps
vaddr=0x0041e334 paddr=0x0001c934 ordinal=314 sz=22 len=21 section=.rdata type=a string=GetTextExtentPoint32W
vaddr=0x0041e34c paddr=0x0001c94c ordinal=315 sz=14 len=13 section=.rdata type=a string=SelectPalette
vaddr=0x0041e35c paddr=0x0001c95c ordinal=316 sz=9 len=8 section=.rdata type=a string=TextOutW
vaddr=0x0041e368 paddr=0x0001c968 ordinal=317 sz=12 len=11 section=.rdata type=a string=CreateFontW
vaddr=0x0041e376 paddr=0x0001c976 ordinal=318 sz=9 len=8 section=.rdata type=a string=Polyline
vaddr=0x0041e382 paddr=0x0001c982 ordinal=319 sz=23 len=22 section=.rdata type=a string=CreateCompatibleBitmap
vaddr=0x0041e39c paddr=0x0001c99c ordinal=320 sz=18 len=17 section=.rdata type=a string=IntersectClipRect
vaddr=0x0041e3b0 paddr=0x0001c9b0 ordinal=321 sz=16 len=15 section=.rdata type=a string=GetTextMetricsW
vaddr=0x0041e3c2 paddr=0x0001c9c2 ordinal=322 sz=13 len=12 section=.rdata type=a string=SelectObject
vaddr=0x0041e3d2 paddr=0x0001c9d2 ordinal=323 sz=19 len=18 section=.rdata type=a string=CreateCompatibleDC
vaddr=0x0041e3e8 paddr=0x0001c9e8 ordinal=324 sz=7 len=6 section=.rdata type=a string=LPTODP
vaddr=0x0041e3f2 paddr=0x0001c9f2 ordinal=325 sz=11 len=10 section=.rdata type=a string=SetBkColor
vaddr=0x0041e400 paddr=0x0001ca00 ordinal=326 sz=13 len=12 section=.rdata type=a string=SetTextColor
vaddr=0x0041e410 paddr=0x0001ca10 ordinal=327 sz=11 len=10 section=.rdata type=a string=GetClipBox
vaddr=0x0041e41e paddr=0x0001ca1e ordinal=328 sz=13 len=12 section=.rdata type=a string=CreateBitmap
vaddr=0x0041e42e paddr=0x0001ca2e ordinal=329 sz=8 len=7 section=.rdata type=a string=Polygon
vaddr=0x0041e438 paddr=0x0001ca38 ordinal=330 sz=9 len=8 section=.rdata type=a string=DeleteDC
vaddr=0x0041e444 paddr=0x0001ca44 ordinal=331 sz=11 len=10 section=.rdata type=a string=GetObjectW
vaddr=0x0041e452 paddr=0x0001ca52 ordinal=332 sz=10 len=9 section=.rdata type=a string=CreatePen
vaddr=0x0041e45e paddr=0x0001ca5e ordinal=333 sz=14 len=13 section=.rdata type=a string=SelectClipRgn
vaddr=0x0041e46c paddr=0x0001ca6c ordinal=334 sz=10 len=9 section=.rdata type=a string=GDI32.dll
vaddr=0x0041e476 paddr=0x0001ca76 ordinal=335 sz=13 len=12 section=.rdata type=a string=OLEAUT32.dll
vaddr=0x0041e486 paddr=0x0001ca86 ordinal=336 sz=15 len=14 section=.rdata type=a string=VerQueryValueW
vaddr=0x0041e498 paddr=0x0001ca98 ordinal=337 sz=24 len=23 section=.rdata type=a string=GetFileVersionInfoSizeW
vaddr=0x0041e4b2 paddr=0x0001cab2 ordinal=338 sz=20 len=19 section=.rdata type=a string=GetFileVersionInfoW
vaddr=0x0041e4c6 paddr=0x0001cac6 ordinal=339 sz=12 len=11 section=.rdata type=a string=VERSION.dll
vaddr=0x0041e4d2 paddr=0x0001cad2 ordinal=340 sz=12 len=11 section=.rdata type=a string=WSOCK32.dll
vaddr=0x0041e4e0 paddr=0x0001cae0 ordinal=341 sz=5 len=4 section=.rdata type=a string=RSDS
vaddr=0x0041e4f8 paddr=0x0001caf8 ordinal=342 sz=56 len=55 section=.rdata type=a string=E:\spwi36\acki\pvlf69yk\4x3k\kw782i\c799y\jz1s43r
24.pdb
[0x00401fb4]> █
```

Информация о файле (i?, rabin2 -h)

```
[0x00401fb4]> iS
[Sections]
idx=00 vaddr=0x00401000 paddr=0x00000400 sz=107008 vsz=106592 perm=-r-x name=.text
idx=01 vaddr=0x00401c00 paddr=0x0001a600 sz=9728 vsz=9522 perm=-r-- name=.rdata
idx=02 vaddr=0x00401f00 paddr=0x0001cc00 sz=30208 vsz=49240 perm=-rw- name=.data
idx=03 vaddr=0x00402c00 paddr=0x00024200 sz=3584 vsz=3432 perm=-r-- name=.rsrc
idx=04 vaddr=0x00402d00 paddr=0x00025000 sz=10240 vsz=10044 perm=-r-- name=.reloc

5 sections

[0x00401fb4]> █
```

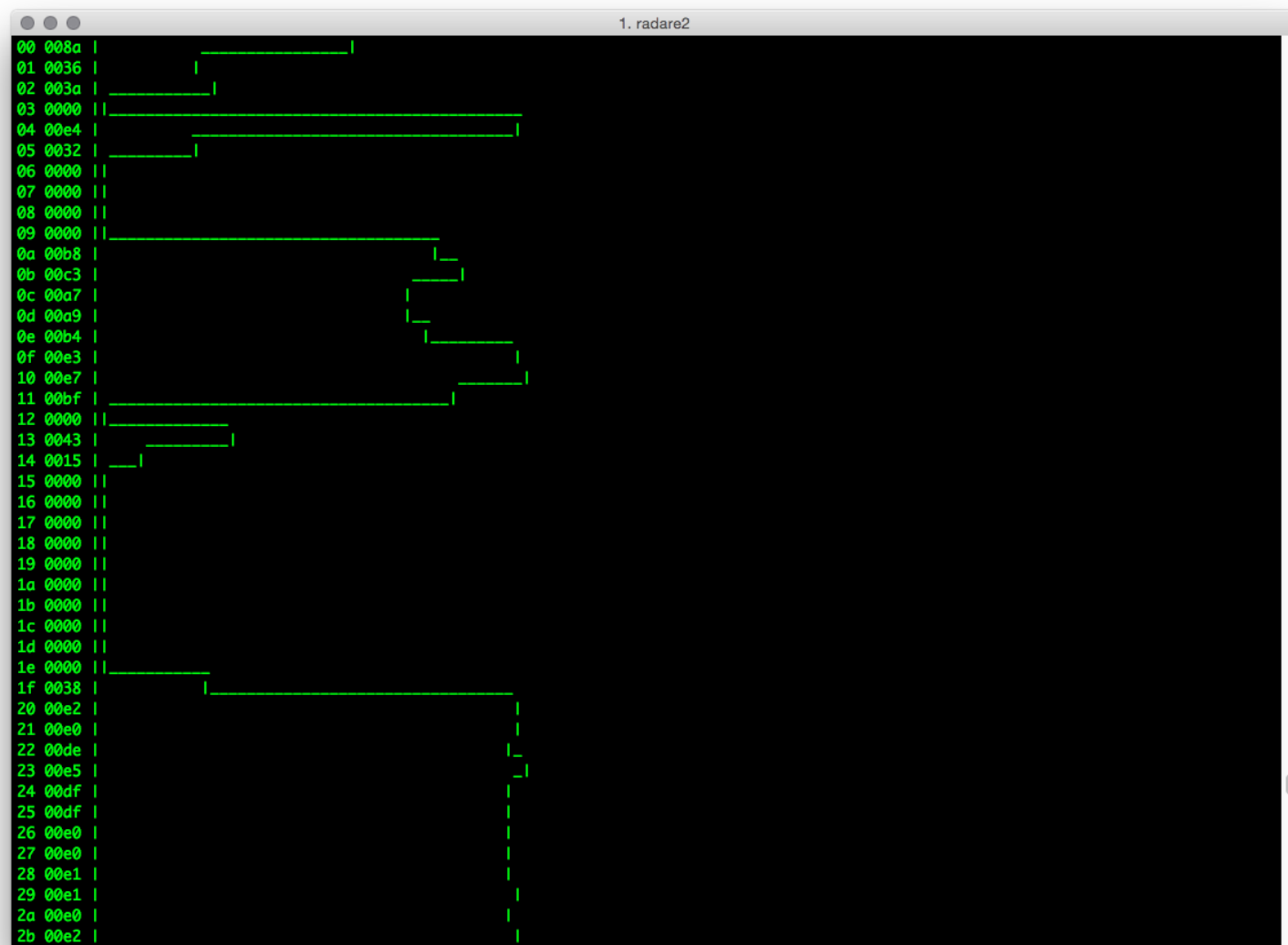
Формат файла (pf?)

```
[0x00401fb4]> pf pe_dos_header
0x00401fb4 = 0x41aac7a1
0x00401fb8 = 9028097
0x00401fbc = (octal) 037200370061
0x00401fc0 = 0x00401fc0 -> 0xffffffffc901e1bc <00000000
<00000000000000000000000000000000000000000000000000000000000000000000
<00000000000000000000000000000000000000000000000000000000000000000000
0x00401fc4 = 4351013
[0x00401fb4]> █
```

Формат файла (pf?)

```
[0x00401fb4]> pf pe_dos_header
0x00401fb4 = 0x41aac7a1
0x00401fb8 = 9028097
0x00401fbc = (octal) 037200370061
0x00401fc0 = 0x00401fc0 -> 0xffffffffc901e1bc <00000000
<00000000000000000000000000000000000000000000000000000000000000000000
<00000000000000000000000000000000000000000000000000000000000000000000
0x00401fc4 = 4351013
[0x00401fb4]> █
```


Обнаруживаем упаковщик и шифрование (#?, p=?, yara, /C?, /m?)



The screenshot shows a Radare2 debugger window titled "1. radare2". The main area displays assembly code with green annotations. The code is organized into blocks, with horizontal lines indicating the boundaries of these blocks. The annotations include vertical lines and horizontal bars that highlight specific instructions and their relationships within the code blocks.

```
00 008a |  
01 0036 |  
02 003a |  
03 0000 ||  
04 00e4 |  
05 0032 |  
06 0000 ||  
07 0000 ||  
08 0000 ||  
09 0000 ||  
0a 00b8 |  
0b 00c3 |  
0c 00a7 |  
0d 00a9 |  
0e 00b4 |  
0f 00e3 |  
10 00e7 |  
11 00bf |  
12 0000 ||  
13 0043 |  
14 0015 |  
15 0000 ||  
16 0000 ||  
17 0000 ||  
18 0000 ||  
19 0000 ||  
1a 0000 ||  
1b 0000 ||  
1c 0000 ||  
1d 0000 ||  
1e 0000 ||  
1f 0038 |  
20 00e2 |  
21 00e0 |  
22 00de |  
23 00e5 |  
24 00df |  
25 00df |  
26 00e0 |  
27 00e0 |  
28 00e1 |  
29 00e1 |  
2a 00e0 |  
2b 00e2 |
```

Первые шаги (а?, radare2 -A)

```
[0x00404e59]> aa
[0x00404e59]> b 20
[0x00404e59]> pd
/ (fcn) entry0 186
|      0x00404e59  55          push ebp
|      0x00404e5a  8bec        mov ebp, esp
|      0x00404e5c  81ec90080000 sub esp, 0x890
|      0x00404e62  83e000      and eax, 0
|      0x00404e65  89b500ffffff mov dword [ebp - 0x100], esi
|      0x00404e6b  89bdfcfeffff mov dword [ebp - 0x104], edi
|      0x00404e71  899df8feffff mov dword [ebp - 0x108], ebx
|      0x00404e77  8d742418    lea esi, dword [esp + 0x18]
|      0x00404e7b  c70613010000 mov dword [esi], 0x113
|      0x00404e81  896e08      mov dword [esi + 8], ebp ; [:4]=0
|      ; JMP XREF from 0x00404e94 (entry0)
|      .-> 0x00404e84  56          push esi
|      | 0x00404e85  ff1518704200 call dword [reloc.KERNEL32.dll_GetVersionExW_24]
|      |      unk(unk, unk, unk) ; section_end..reloc+-4370945
|      | 0x00404e8b  ff06        inc dword [esi]
|      | 0x00404e8d  36813e18010. cmp dword ss:[esi], 0x118
|      |      <= 0x00404e94  72ee        jb 0x404e84
|      | 0x00404e96  fc          cld
|      | 0x00404e97  56          push esi
|      | 0x00404e98  ff1528704200 call dword [reloc.KERNEL32.dll_GetModuleHandleA_40]
|      |      unk(unk) ; section_end..reloc+-4370945
|      | 0x00404e9e  c7442404000. mov dword [esp + 4], 0 ; [:4]=0
|      | 0x00404ea6  6a00        push 0
[0x00404e59]> █
```

Базовые команды (/?)

```
[0x00404e59]> /a call eax
# 6 [0x401000-0x42b0f8]
hits: 2
0x0040e00a hit0_0 ffd0
0x0041eac6 hit0_1 ffd0
[0x00404e59]> /a call ebx
# 6 [0x401000-0x42b0f8]
hits: 3
0x00405e29 hit1_0 ffd3
0x0041c76f hit1_1 ffd3
0x004253a9 hit1_2 ffd3
[0x00404e59]> █
```

Базовые команды (p?)

```
[0x00404e59]> p?  
|Usage: p[=68abcdDfiImrstuxz] [arg|len]  
| p=[bep?] [blks] show entropy/printable chars/chars bars  
| p2 [len] 8x8 2bpp-tiles  
| p6[de] [len] base64 decode/encode  
| p8 [len] 8bit hexpair list of bytes  
| pa[ed] [hex|asm] assemble (pa) disasm (pad) or esil (pae) from hexpairs  
| p[bB] [len] bitstream of N bytes  
| pc[p] [len] output C (or python) format  
| p[dD][lf] [l] disassemble N opcodes/bytes (see pd?)  
| pf[?|.nam] [fmt] print formatted data (pf.name, pf.name $<expr>)  
| p[iI][df] [len] print N instructions/bytes (f=func) (see pi? and pdi)  
| pm [magic] print libmagic data (pm? for more information)  
| pr [len] print N raw bytes  
| p[kK] [len] print key in randomart (K is for mosaic)  
| ps[pwz] [len] print pascal/wide/zero-terminated strings  
| pt[dn?] [len] print different timestamps  
| pu[w] [len] print N url encoded bytes (w=wide)  
| pv[jh] [mode] bar|json|histogram blocks (mode: e?search.in)  
| p[xX][owq] [len] hexdump of N bytes (o=octal, w=32bit, q=64bit)  
| pz [len] print zoom view (see pz? for help)  
| pwd display current working directory  
[0x00404e59]> █
```

Базовые команды (w?)

```
[0x00404e59]> w?  
|Usage: w[x] [str] [<file] [<<EOF] [@addr]  
| wc list all write changes  
| w[1248][+~][n] increment/decrement byte,word..  
| w foobar write string 'foobar'  
| wh r2 whereis/which shell command  
| wr 10 write 10 random bytes  
| ww foobar write wide string 'f\x00o\x00o\x00b\x00a\x00r\x00'  
| wa push ebp write opcode, separated by ';' (use '"' around the command)  
| waf file assemble file and write bytes  
| wA r 0 alter/modify opcode at current seek (see wA?)  
| wb 010203 fill current block with cyclic hexpairs  
| wc[ir*?] write cache undo/commit/reset/list (io.cache)  
| wd [off] [n] duplicate N bytes from offset at current seek (memcpy) (see y?)  
| wx 9090 write two intel nops  
| ww eip+34 write 32-64 bit value  
| wo? hex write in block with operation. 'wo?' fmi  
| wm f0ff set binary mask hexpair to be used as cyclic write mask  
| ws pstring write 1 byte for length and then the string  
| wf -|file write contents of file at current offset  
| wF -|file write contents of hexpairs file here  
| wp -|file apply radare patch file. See wp? fmi  
| wt file [sz] write to file (from current seek, blocksize or sz bytes)  
[0x00404e59]> █
```

Crackme

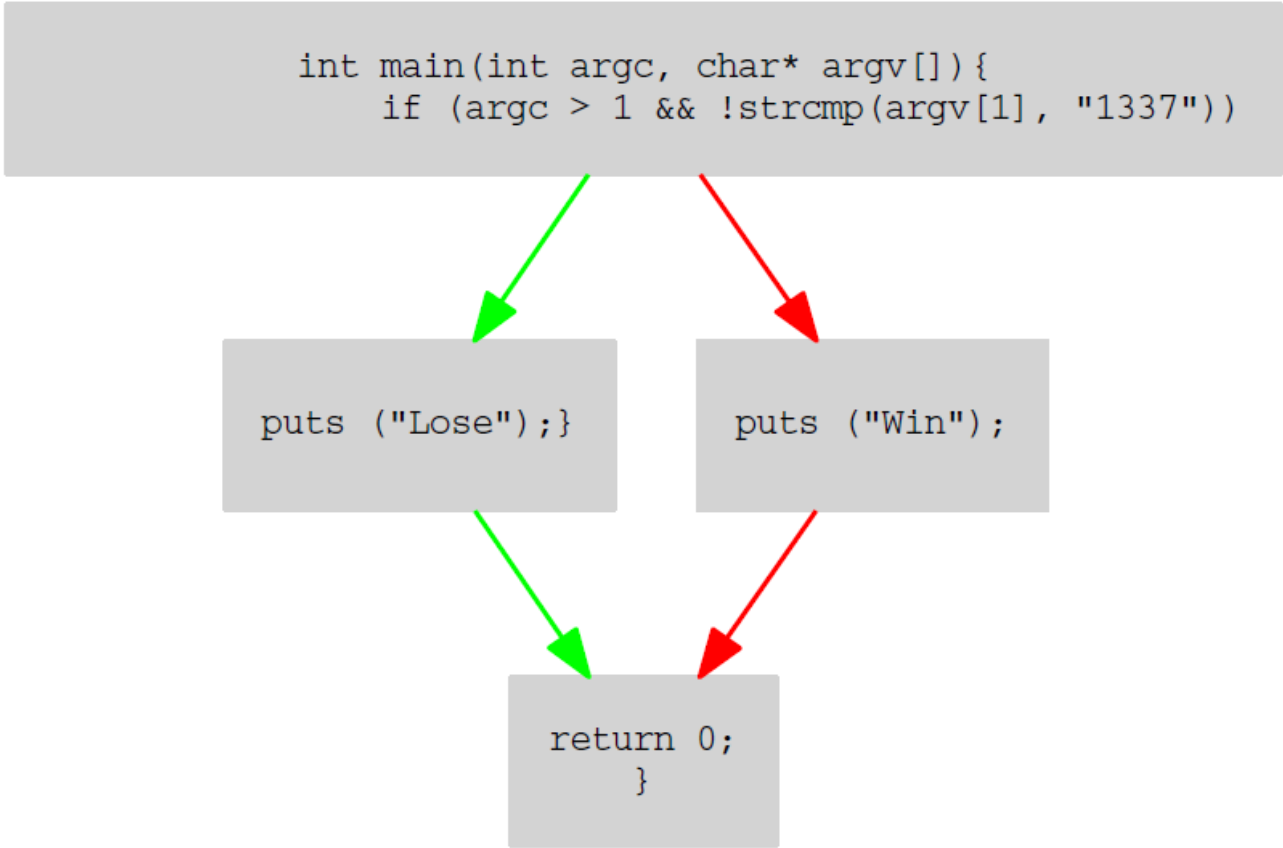
```
0x0804808a b9f8900408 mov ecx, str._nPassword_ ; 0x080490f8
0x0804808f ba0d000000 mov edx, 0xd ; 0x0000000d
0x08048094 cd80 int 0x80
    syscall[0x80][0]=? ; section_end..shstrtab+91
0x08048096 ba00010000 mov edx, 0x100 ; 0x00000100
0x0804809b b91b910408 mov ecx, 0x804911b ; 0x0804911b
0x080480a0 bb00000000 mov ebx, 0x0
0x080480a5 b803000000 mov eax, 0x3 ; 0x00000003
0x080480aa cd80 int 0x80
    syscall[0x80][0]=? ; section_end..shstrtab+91
0x080480ac be26910408 mov esi, str.QTBXCTU ; 0x08049126
0x080480b1 89f7 mov edi, esi
0x080480b3 31db xor ebx, ebx
0x080480b5 fc cld
; JMP XREF from 0x080480c3 (section..text)
--> 0x080480b6 ac lods b
0x080480b7 3421 xor al, 0x21
0x080480b9 aa stos b
0x080480ba 43 inc ebx
0x080480bb 81fb07000000 cmp ebx, 0x7
=> 0x080480c1 7402 je 0x80480c5 ; (section..text)
<= 0x080480c3 e2f1 loop 0x80480b6 ; (section..text)
; JMP XREF from 0x080480c1 (section..text)
-> 0x080480c5 be1b910408 mov esi, 0x804911b ; 0x0804911b
0x080480ca bf26910408 mov edi, str.QTBXCTU ; 0x08049126
0x080480cf b907000000 mov ecx, 0x7 ; 0x00000007
0x080480d4 fc cld
0x080480d5 f3a6 repe cmpsb
<=> 0x080480d7 7516 jne 0x80480ef ; (section..text)
0x080480d9 b804000000 mov eax, 0x4 ; 0x00000004
0x080480de bb01000000 mov ebx, 0x1 ; 0x00000001
0x080480e3 b905910408 mov ecx, str.Great_you_did_it____n_n ; 0x08049105
0x080480e8 ba16000000 mov edx, 0x16 ; 0x00000016
0x080480ed cd80 int 0x80
    syscall[0x80][0]=? ; section_end..shstrtab+91
; JMP XREF from 0x080480d7 (section..text)
--> 0x080480ef b801000000 mov eax, 0x1 ; 0x00000001
0x080480f4 cd80 int 0x80
    syscall[0x80][0]=? ; section_end..shstrtab+91
;-- section_end..text:
0x080480f6 0000 add [eax], al
```

Crackme

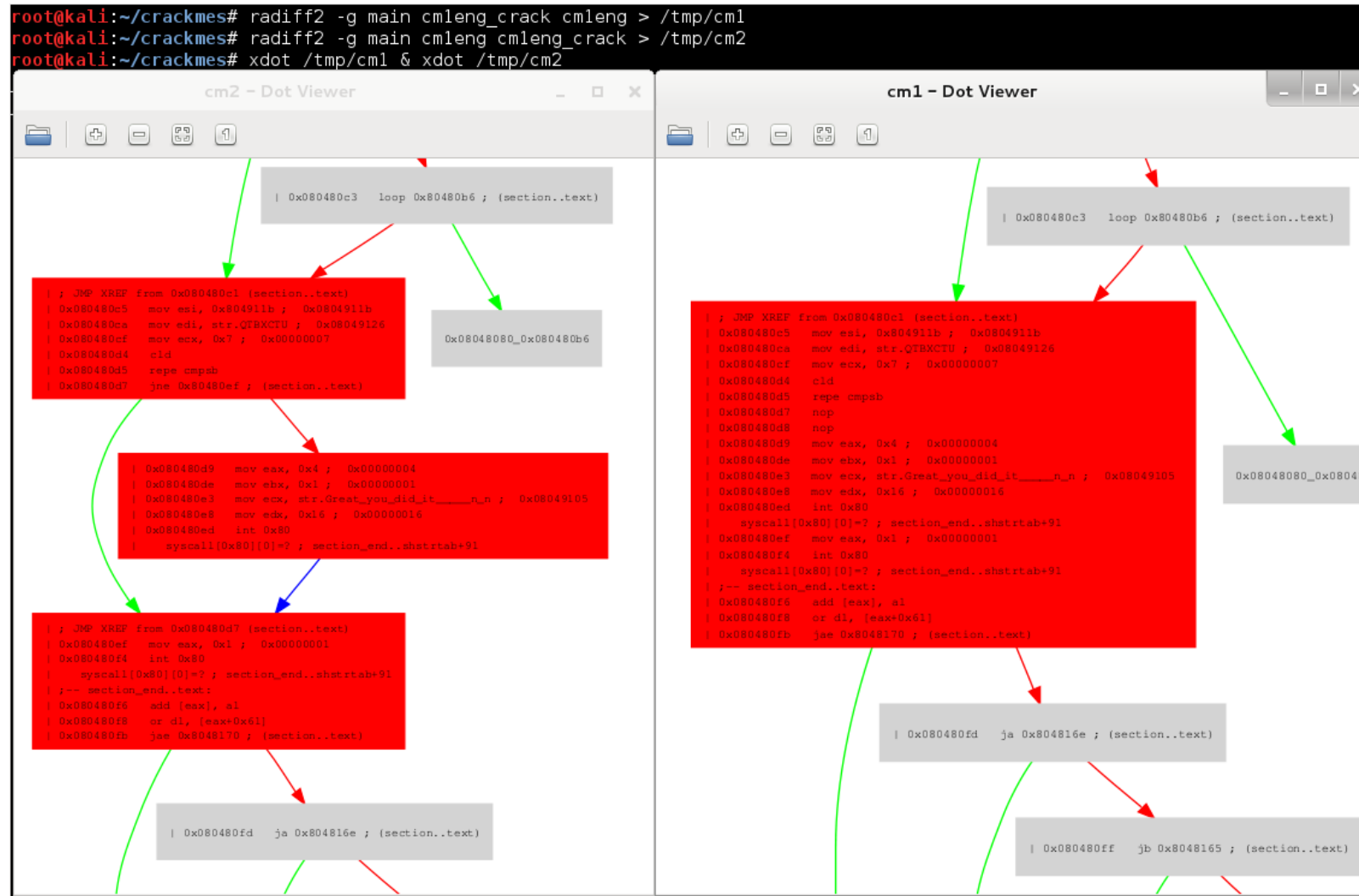
```
[0x080480d7]> wx eb00
[0x080480d7]> pd 10
   =< 0x080480d7    eb00          jmp 0x80480d9
   -> 0x080480d9    b804000000   mov eax, 0x4 ; 0x00000004
      0x080480de    bb01000000   mov ebx, 0x1 ; 0x00000001
      0x080480e3    b905910408   mov ecx, str.Great_you_did_it____n_n ; 0x08049105
      0x080480e8    ba16000000   mov edx, 0x16 ; 0x00000016
      0x080480ed    cd80         int 0x80
      syscall[0x80][0]=? ; section_end..shstrtab
      0x080480ef    b801000000   mov eax, 0x1 ; 0x00000001
      0x080480f4    cd80         int 0x80
      syscall[0x80][0]=? ; section_end..shstrtab
      ;-- section_end..text:
      0x080480f6    0000         add [eax], al
      0x080480f8    0a5061      or dl, [eax+0x61]
[0x080480d7]> q
root@kali:~/crackmes# ./cmleng_crack

Password : dukebarman
Great you did it !:)
```

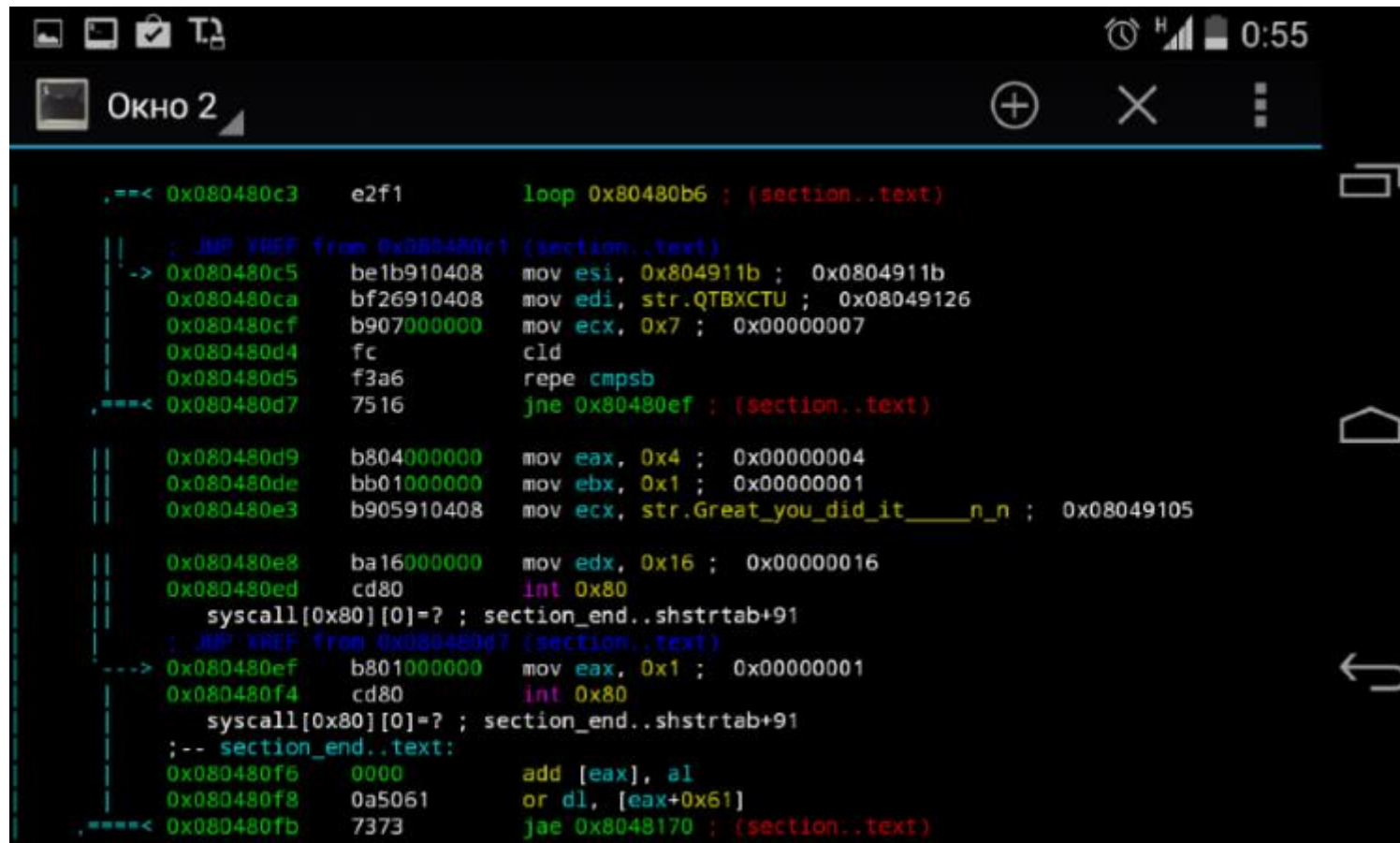
Graph (a?)



Graph (a?)



Android



The screenshot shows an Android phone screen with a debugger window titled "Окно 2". The window displays assembly code with addresses, hex values, and instructions. The code includes instructions like `loop`, `jmp`, `mov`, `cdq`, `repe cmpsb`, `jne`, `mov`, `int`, `syscall`, `add`, `or`, and `jae`. The code is color-coded with green for addresses and hex values, and red for instructions and comments. The phone's status bar at the top shows the time as 0:55 and various icons. The debugger window has a dark background with a light blue border.

```

.--< 0x080480c3  e2f1      loop 0x80480b6 ; (section..text)
|
|   ; JMP XREF from 0x080480c1 (section..text)
|   -> 0x080480c5  be1b910408  mov esi, 0x804911b ; 0x0804911b
|       0x080480ca  bf26910408  mov edi, str.QTBXCTU ; 0x08049126
|       0x080480cf  b907000000  mov ecx, 0x7 ; 0x00000007
|       0x080480d4  fc          cld
|       0x080480d5  f3a6       repe cmpsb
|   ----< 0x080480d7  7516       jne 0x80480ef ; (section..text)
|
|       0x080480d9  b804000000  mov eax, 0x4 ; 0x00000004
|       0x080480de  bb01000000  mov ebx, 0x1 ; 0x00000001
|       0x080480e3  b905910408  mov ecx, str.Great_you_did_it_____n_n ; 0x08049105
|
|       0x080480e8  ba16000000  mov edx, 0x16 ; 0x00000016
|       0x080480ed  cd80       int 0x80
|       syscall[0x80][0]=? ; section_end..shstrtab+91
|       ; JMP XREF from 0x080480d7 (section..text)
|   --> 0x080480ef  b801000000  mov eax, 0x1 ; 0x00000001
|       0x080480f4  cd80       int 0x80
|       syscall[0x80][0]=? ; section_end..shstrtab+91
|   ;-- section_end..text:
|       0x080480f6  0000       add [eax], al
|       0x080480f8  0a5061    or dl, [eax+0x61]
|   ----< 0x080480fb  7373       jae 0x8048170 ; (section..text)

```

Время для практики



r2 для разработки эксплойтов

r^2 для анализа прошивок

Ресурсы



- [Github repo](#)
- [Official website](#)
- [The r2 blog](#)
- [The r2 book](#)
- [Установка r2](#)
- [Twitter](#)
- [VM для воркшопа](#)