Reverse Engineering

go beyond the fear of Assembly Language

REPRESENTED BY: RAKAN ALDUWAYSAN

whoami

My name is Rakan AlDuwaysan

- Computer Engineer & Security Researcher.
- In love with Hardware & Linux & Reverse Engineering, etc.
- I've been hacking for more than 10 years for fun and profit :).

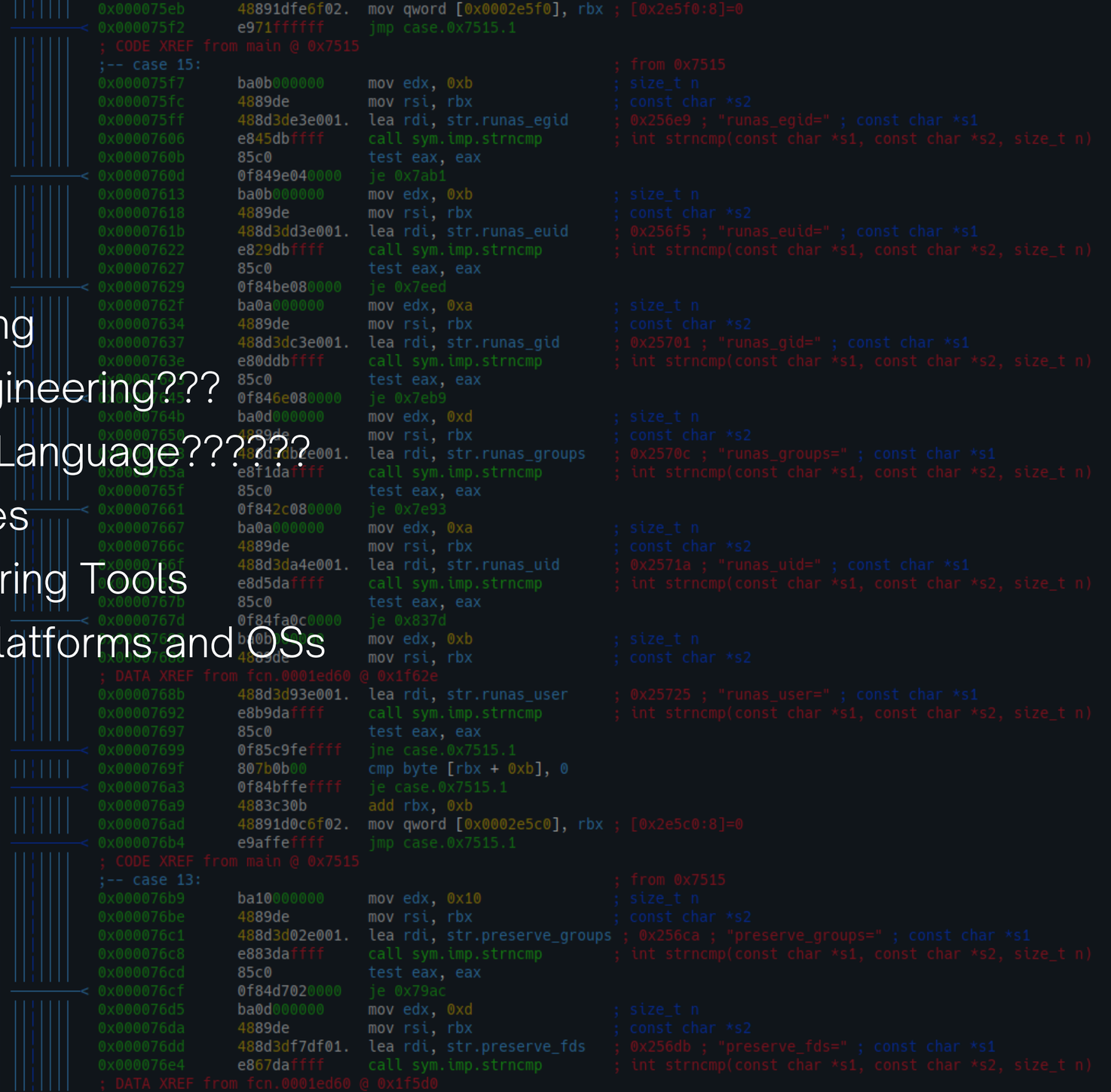
@R4kaaaN

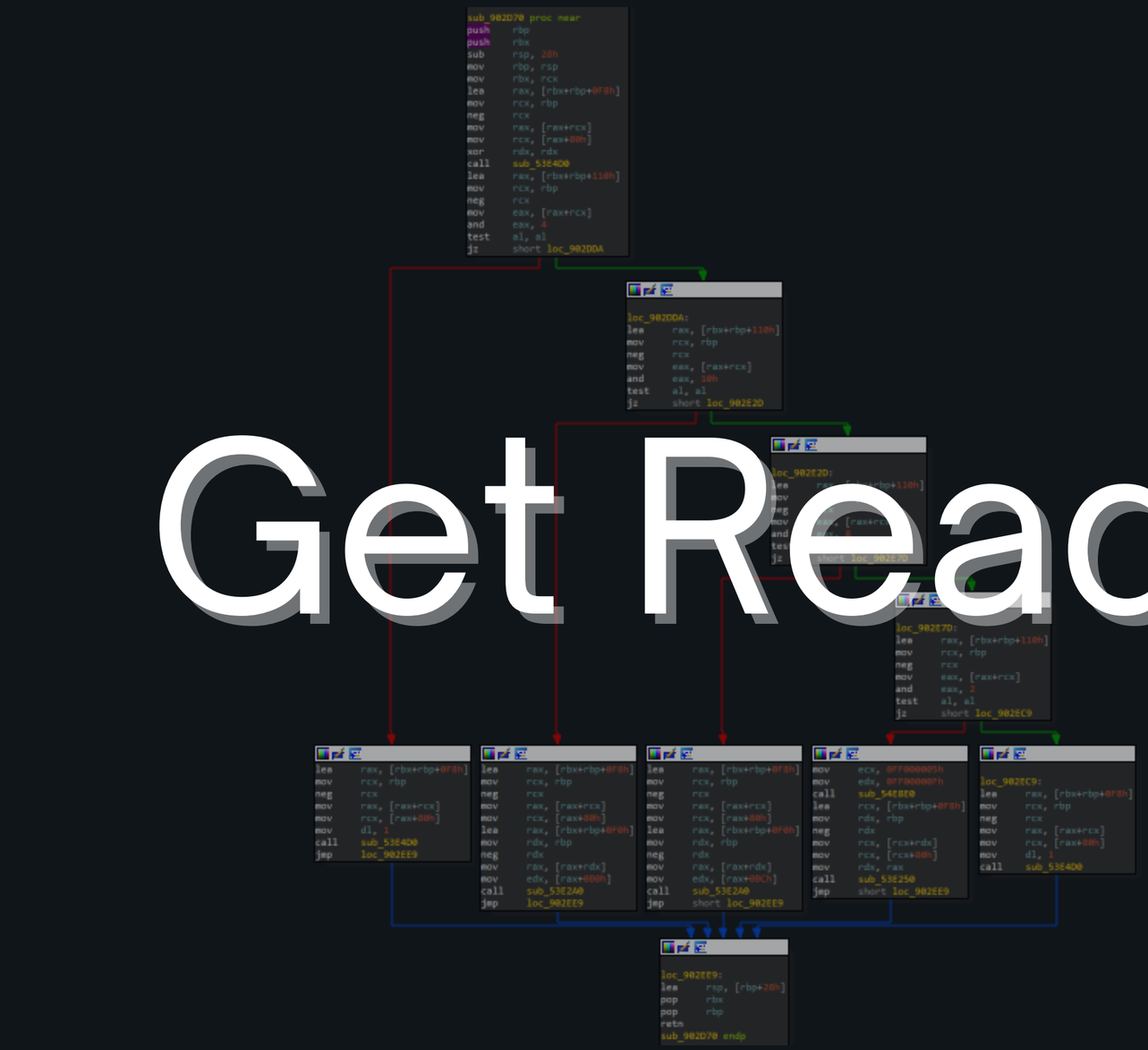
Rakan Alduwaysan

R4kaaan@protonmail.com

Agenda

- Definition of Reverse Engineering
- Why Do We Need Reverse Engineering???
- Do We Really Need Assembly Language??????
- Reverse Engineering Techniques
- Categories of Reverse Engineering Tools
- Difference between different Platforms and OSs
- Demos



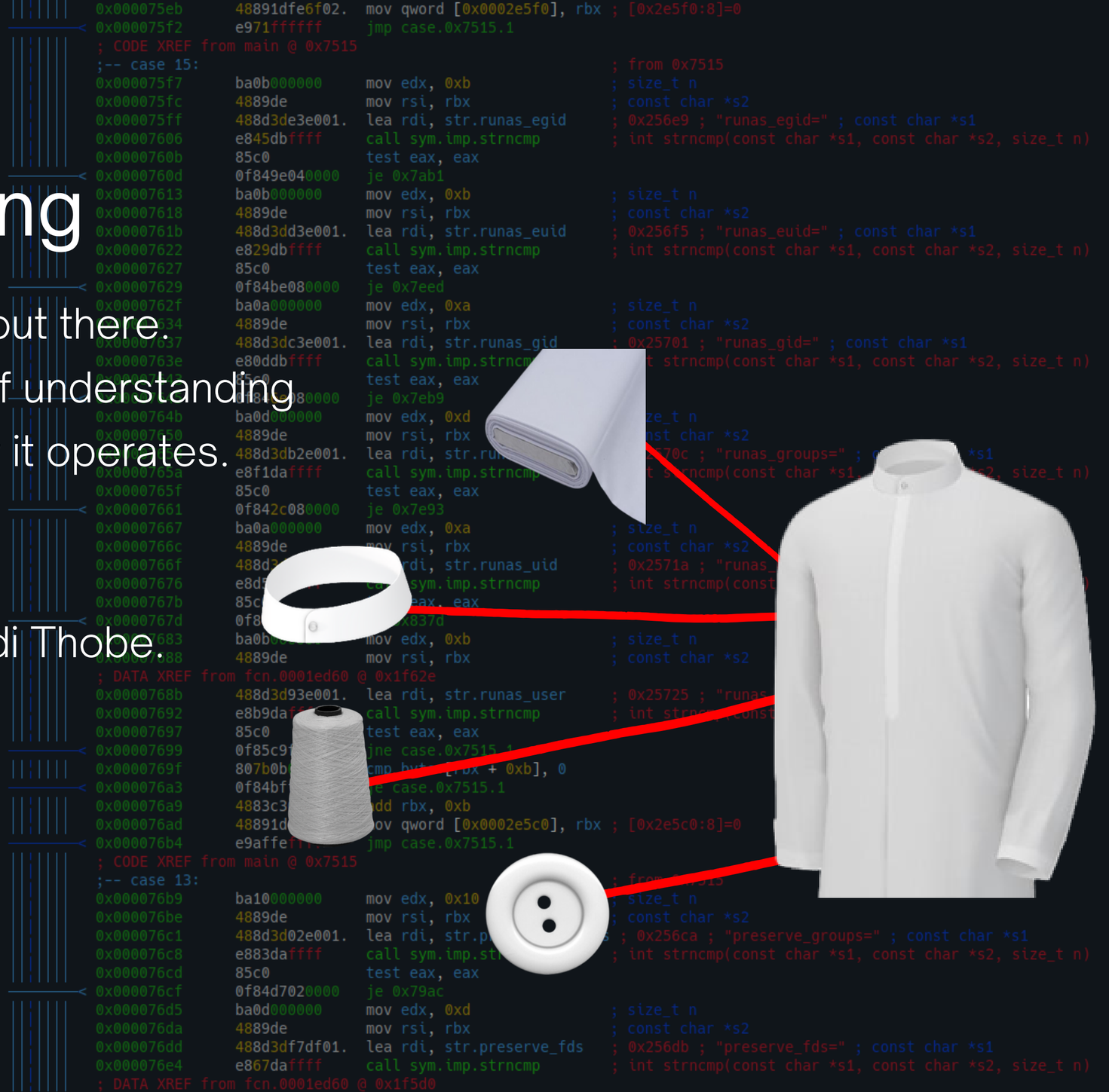

sub_902D70 proc near
push rbp
push rbx
sub rsp, 28h
mov rbp, rsp
mov rbx, rcx
lea rax, [rbx+rbp+0F8h]
mov rcx, rbp
neg rcx
mov rax, [rax+rcx]
mov rcx, [rax+88h]
xor rdx, rdx
call sub_53E4D0
lea rax, [rbx+rbp+110h]
mov rcx, rbp
neg rcx
mov eax, [rax+rcx]
and eax, 4
test al, al
jz short loc_902DDA
loc_902DDA:
lea rax, [rbx+rbp+110h]
mov rcx, rbp
neg rcx
mov eax, [rax+rcx]
and eax, 10h
test al, al
jz short loc_902E2D
loc_902E2D:
loc_902E7D:
lea rax, [rbx+rbp+110h]
mov rcx, rbp
neg rcx
mov eax, [rax+rcx]
and eax, 2
test al, al
jz short loc_902EC9
lea rax, [rbx+rbp+0F8h]
mov rcx, rbp
neg rcx
mov rax, [rax+rcx]
mov rcx, [rax+88h]
mov dl, 1
call sub_53E4D0
jmp loc_902EE9
lea rax, [rbx+rbp+0F8h]
mov rcx, rbp
neg rcx
mov rax, [rax+rcx]
mov rcx, [rax+88h]
lea rax, [rbx+rbp+0F0h]
mov rdx, rbp
neg rdx
mov rax, [rax+rdx]
mov edx, [rax+80h]
call sub_53E2A0
jmp loc_902EE9
lea rax, [rbx+rbp+0F8h]
mov rcx, rbp
neg rcx
mov rax, [rax+rcx]
mov rcx, [rax+88h]
lea rax, [rbx+rbp+0F0h]
mov rdx, rbp
neg rdx
mov rax, [rax+rdx]
mov edx, [rax+0Ch]
call sub_53E2A0
jmp short loc_902EE9
mov ecx, 0FF000005h
mov edx, 0FF00000Fh
call sub_542BE0
lea rcx, [rbx+rbp+0F8h]
mov rdx, rbp
neg rdx
mov rcx, [rcx+rdx]
mov rcx, [rcx+88h]
mov rdx, rax
call sub_53E250
jmp short loc_902EE9
loc_902EC9:
lea rax, [rbx+rbp+0F8h]
mov rcx, rbp
neg rcx
mov rax, [rax+rcx]
mov rcx, [rax+88h]
mov dl, 1
call sub_53E4D0
loc_902EE9:
lea rsp, [rbp+28h]
pop rbx
pop rbp
retn
sub_902D70 endp

Get Ready!

Reverse Engineering

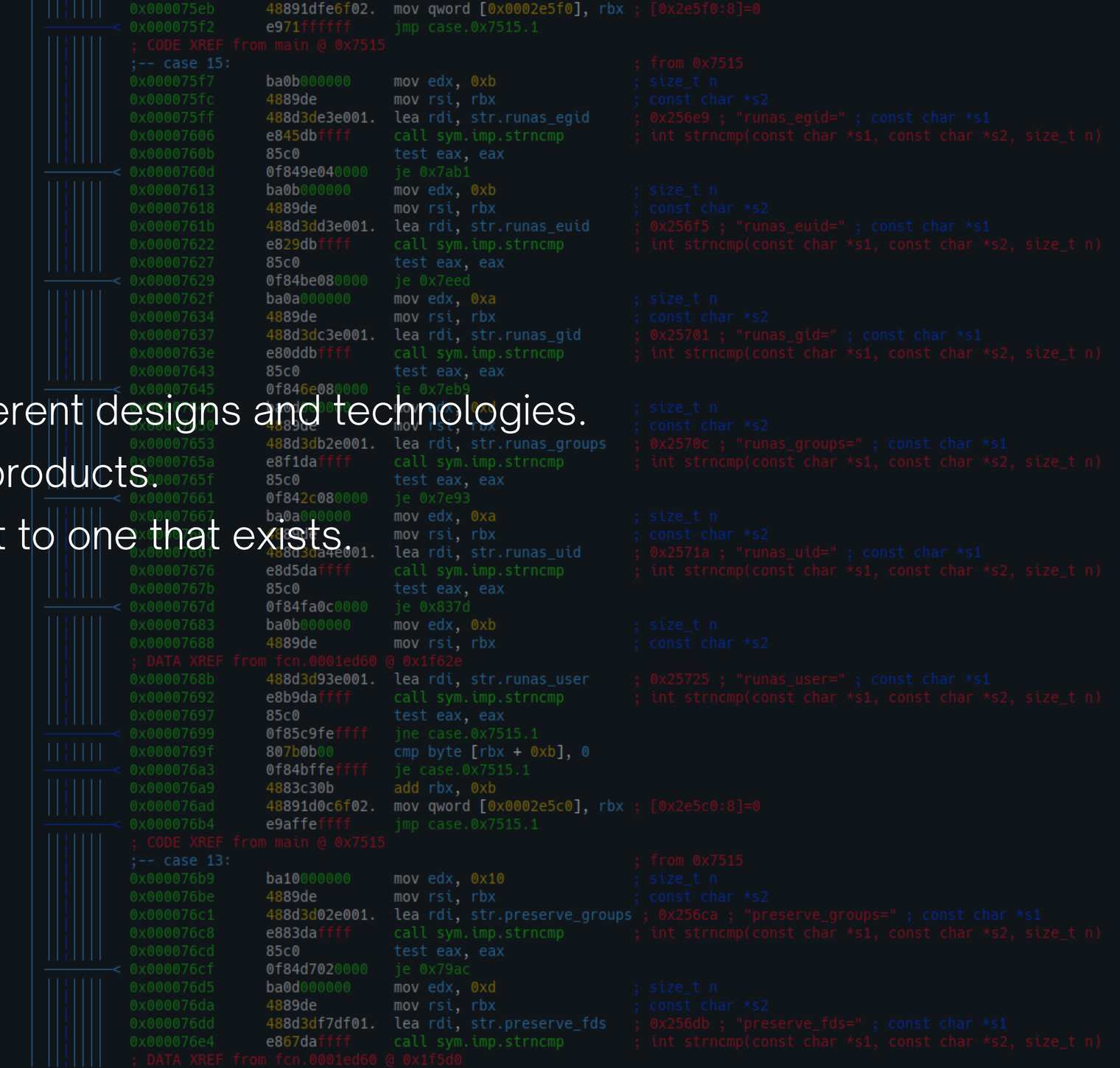
- There are plenty of definitions out there.
- But basically, It's the process of understanding how something is built and how it operates.

- Non-technical example:
- is to Reverse Engineer the Saudi Thobe.



The Need of RE

- First, It's a learning tool ;).
- Exploring the internals of different designs and technologies.
- Reconstructing an outdated products.
- Manufacture a similar product to one that exists.
- Malware Analysis.
- Exploit Development.



Do We Really Need ASM?

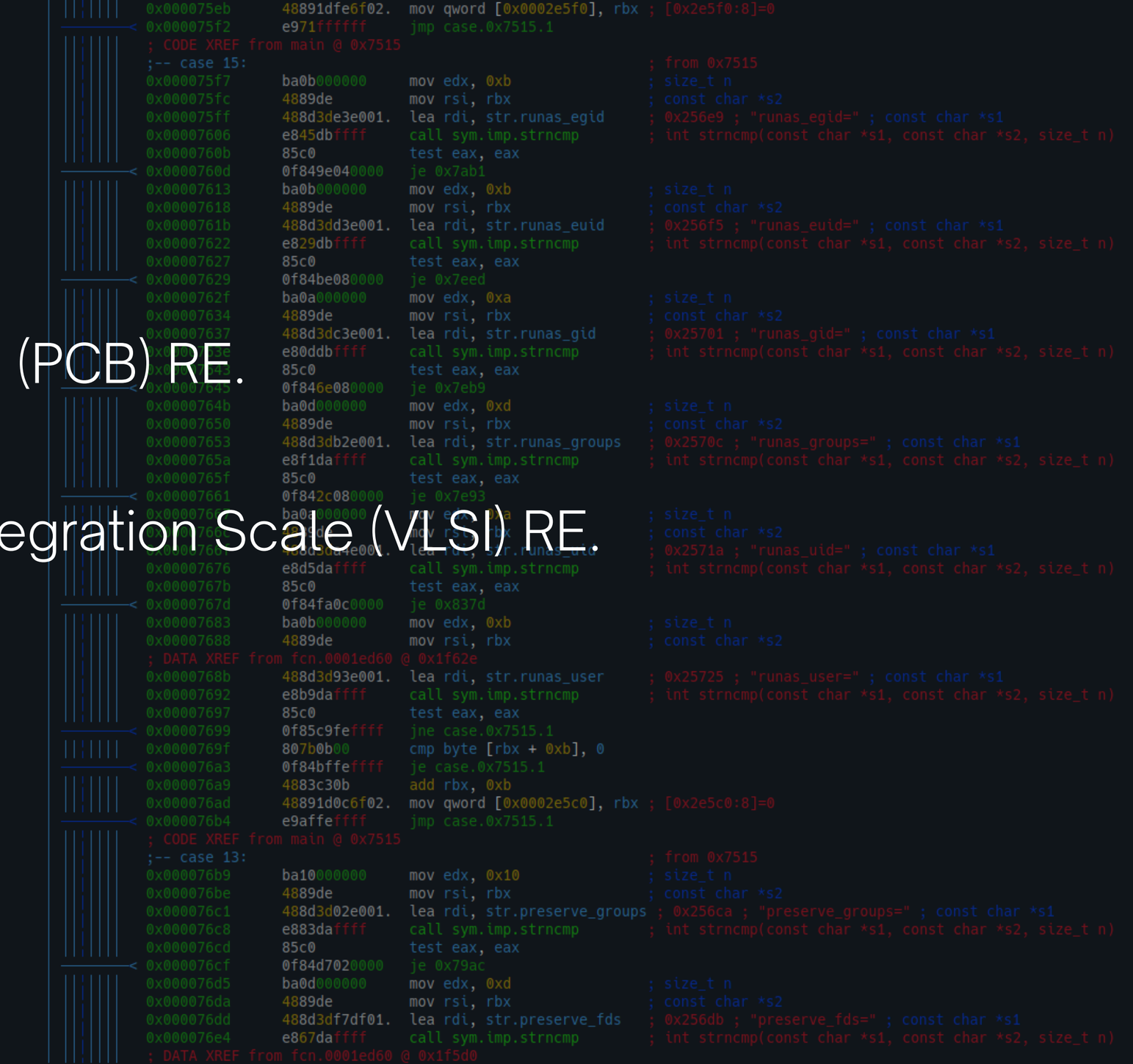
Depends!

Technical Fields of RE

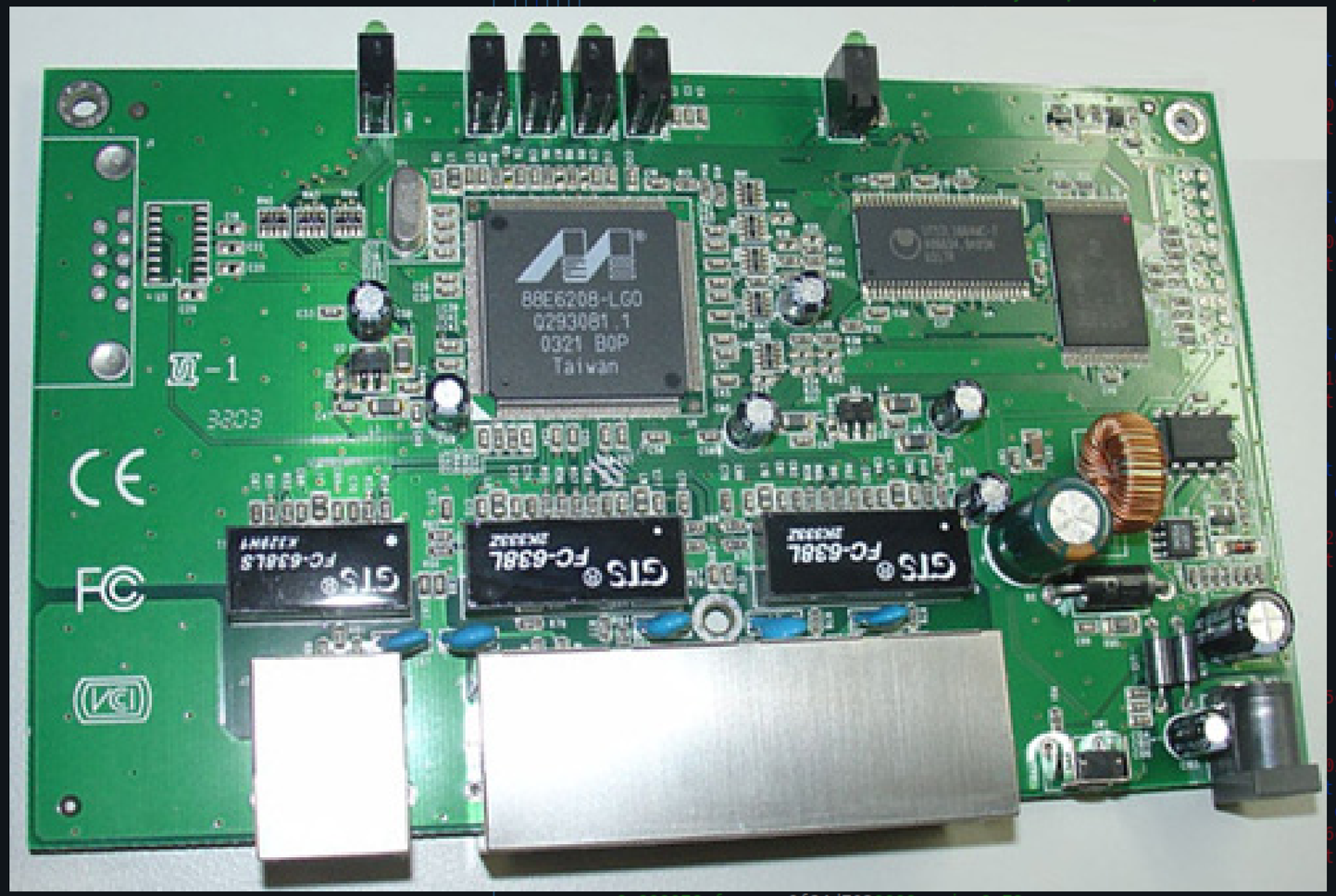
- Hardware
- Software

Hardware RE

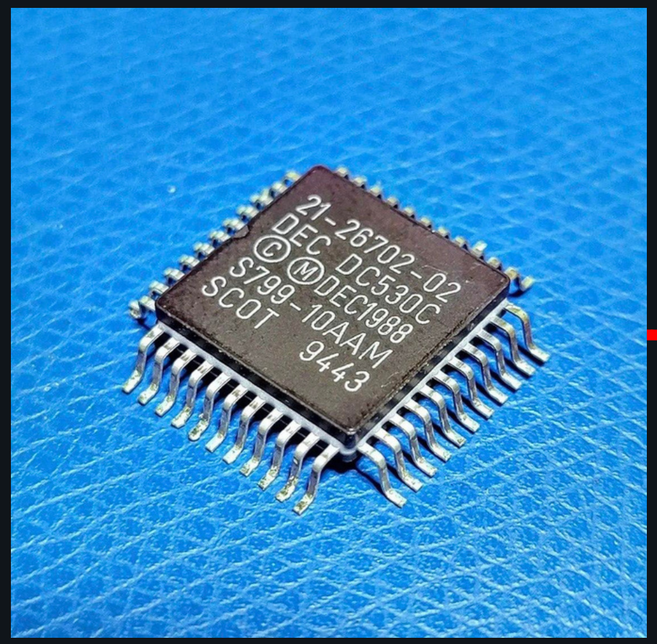
- Printed Circuit Board (PCB) RE.
- Very Large-Scale Integration Scale (VLSI) RE.
- Firmware Extraction.

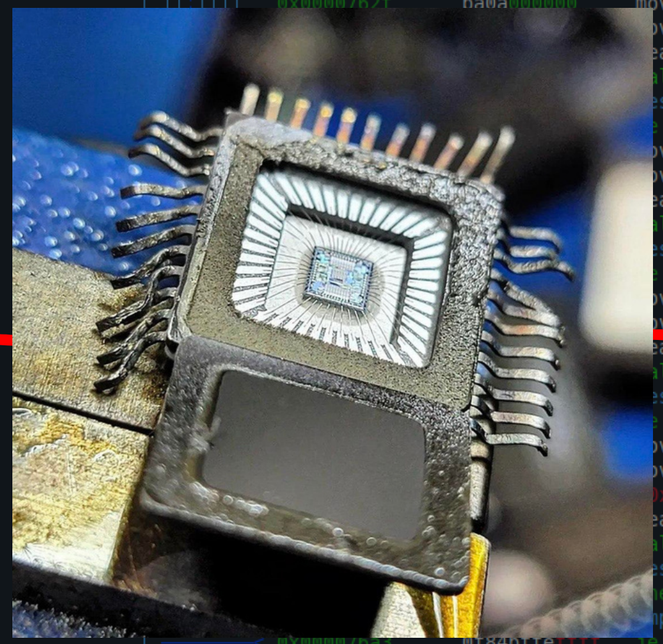


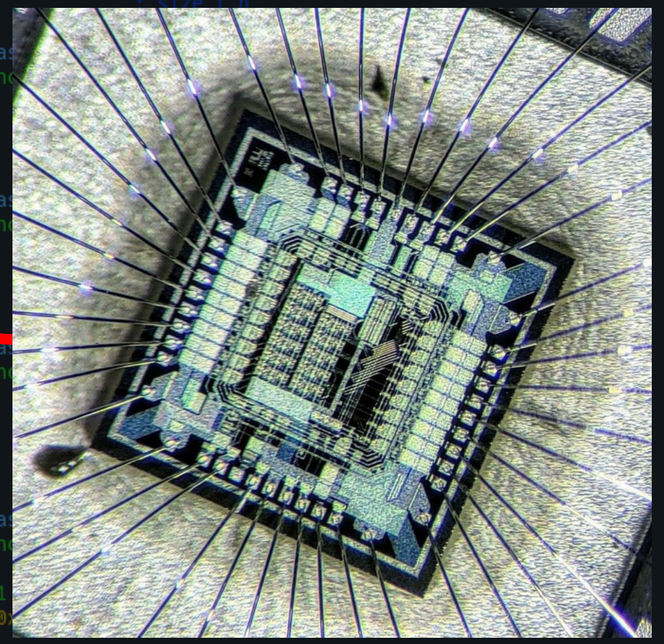
Printed Circuit Board (PCB) RE.



VLSI RE

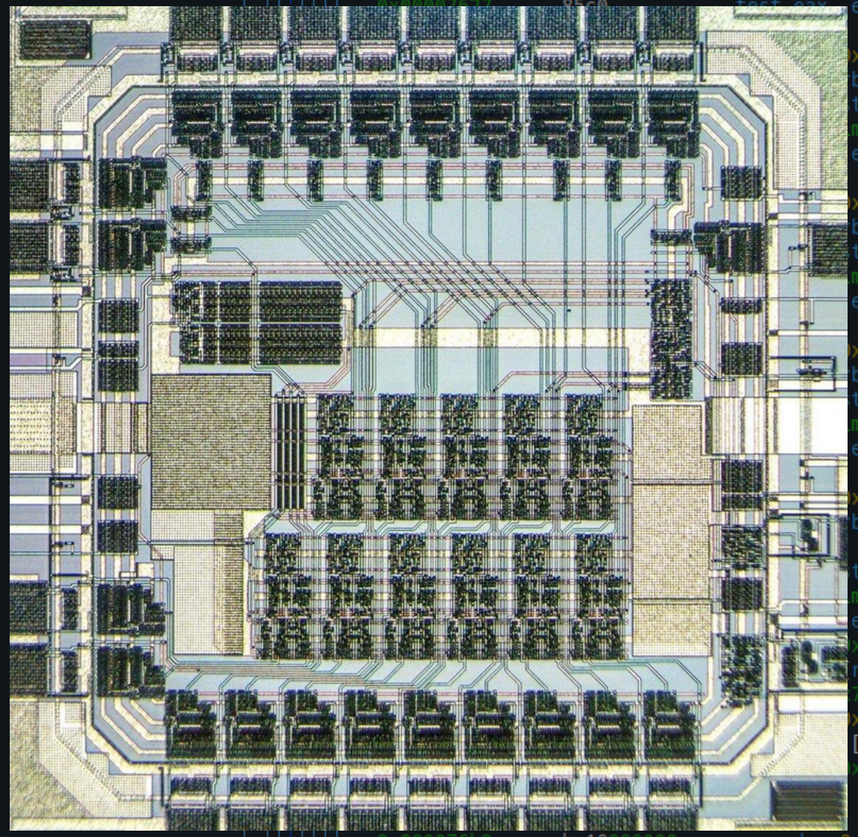






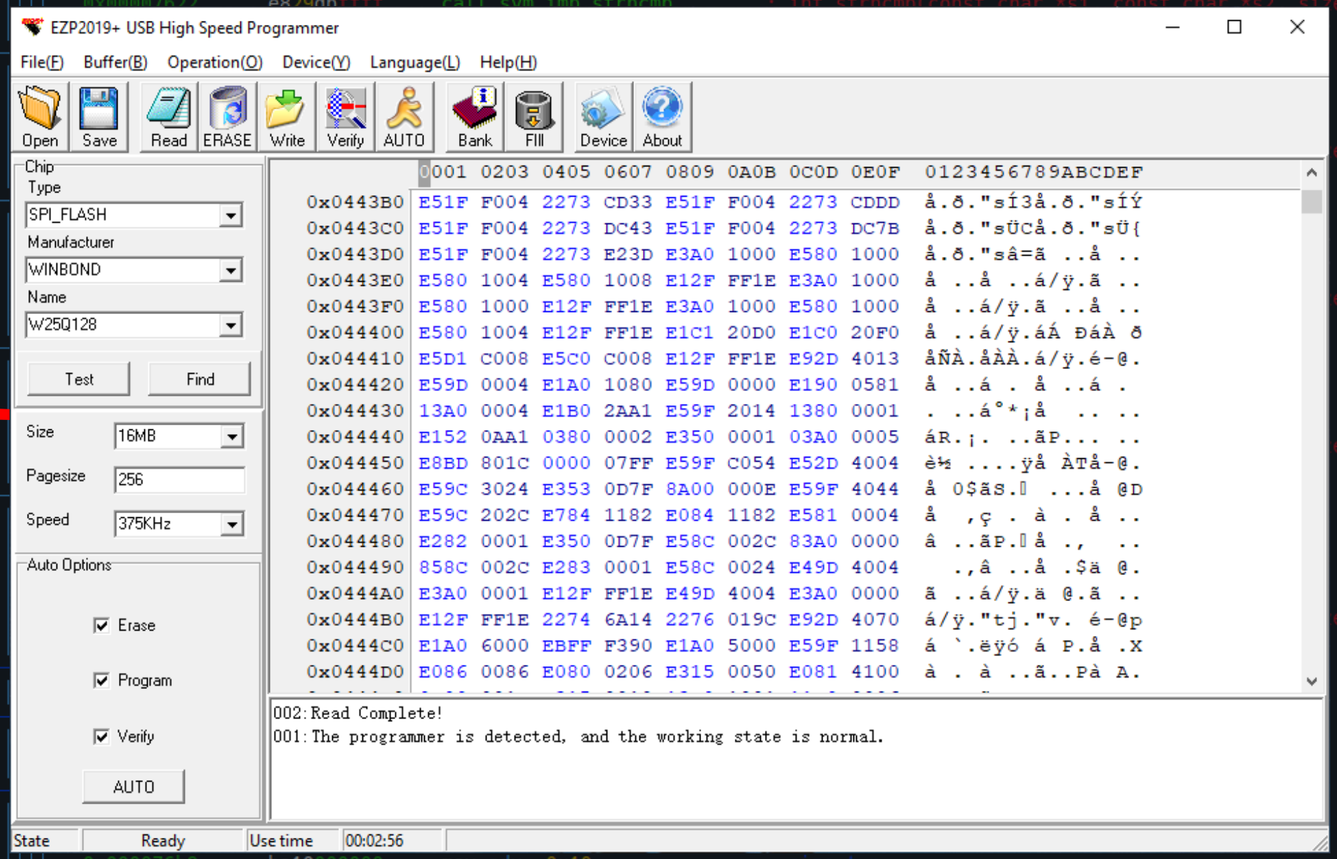
CREDITS TO @EVILMONKEYZDESIGNZ

VLSI RE (Con.)



CREDITS TO @EVILMONKEYZDESIGNZ

Firmware Extraction



CREDITS TO JESÚS MARÍA GÓMEZ MORENO

Software RE

- Malware Analysis
- Exploit Development
- Cracking (Software Piracy)
- Development

RE Techniques

- Static Analysis:
  Examine the code without execution.

- Dynamic Analysis (Debugging):
  Evaluation the app during runtime.

```
gdb-peda$ r
Starting program: /usr/bin/ls
[----------------------------------registers-----------------------------------]
RAX: 0xffffffff
RBX: 0x0
RCX: 0x5a1
RDX: 0x5a1
RSI: 0x1
RDI: 0x5a1
RBP: 0x0
RSP: 0x7fffffffdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>:   mov    r14,rax)
RIP: 0x7ffff7fec2f0 (<malloc>:  push   rbp)
R8 : 0x20000000 ('')
R9 : 0x0
R10: 0x7ffff7ff4e22 ("MALLOC_CHECK_")
R11: 0x1
R12: 0x7ffff7ff435a --> 0x706e203d3d206900 ('')
R13: 0x0
R14: 0x4f4c5f4543415254 ('TRACE_LO')
R15: 0x10
EFLAGS: 0x293 (CARRY parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x7ffff7fec2e1 <unsetenv+113>:       jmp    0x7ffff7fec2ad <unsetenv+61>
   0x7ffff7fec2e3:      nop    WORD PTR cs:[rax+rax*1+0x0]
   0x7ffff7fec2ed:      nop    DWORD PTR [rax]
=> 0x7ffff7fec2f0 <malloc>:     push   rbp
   0x7ffff7fec2f1 <malloc+1>:   push   rbx
   0x7ffff7fec2f2 <malloc+2>:   mov    rbx,rdi
   0x7ffff7fec2f5 <malloc+5>:   sub    rsp,0x8
   0x7ffff7fec2f9 <malloc+9>:   mov    rdx,QWORD PTR [rip+0x11e20]        # 0x7ffff7ffe120 <alloc_end>
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>:  mov    r14,rax)
0008| 0x7fffffffdcf0 --> 0x0
0016| 0x7fffffffdcf8 --> 0x7ffff7ff435a --> 0x706e203d3d206900 ('')
0024| 0x7fffffffdd00 --> 0x20000000 ('')
0032| 0x7fffffffdd08 --> 0x100
0040| 0x7fffffffdd10 --> 0x1
0048| 0x7fffffffdd18 --> 0x2000000000000000 ('')
0056| 0x7fffffffdd20 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, malloc (n=0x5a1) at dl-minimal.c:50
50      dl-minimal.c: No such file or directory.
gdb-peda$ x/128wzx $rsp
0x7fffffffdce8: 0xf7fddce0      0x00007fff      0x00000000      0x00000000
0x7fffffffdcf8: 0xf7ff435a      0x00007fff      0x20000000      0x00000000
0x7fffffffdd08: 0x00000100      0x00000000      0x00000001      0x00000000
0x7fffffffdd18: 0x00000000      0x20000000      0x00000000      0x00000000
0x7fffffffdd28: 0x55554040      0x00005555      0xffffdff0      0x00007fff
0x7fffffffdd38: 0x00000000      0x00000000      0xf7ff3018      0x00007fff
```

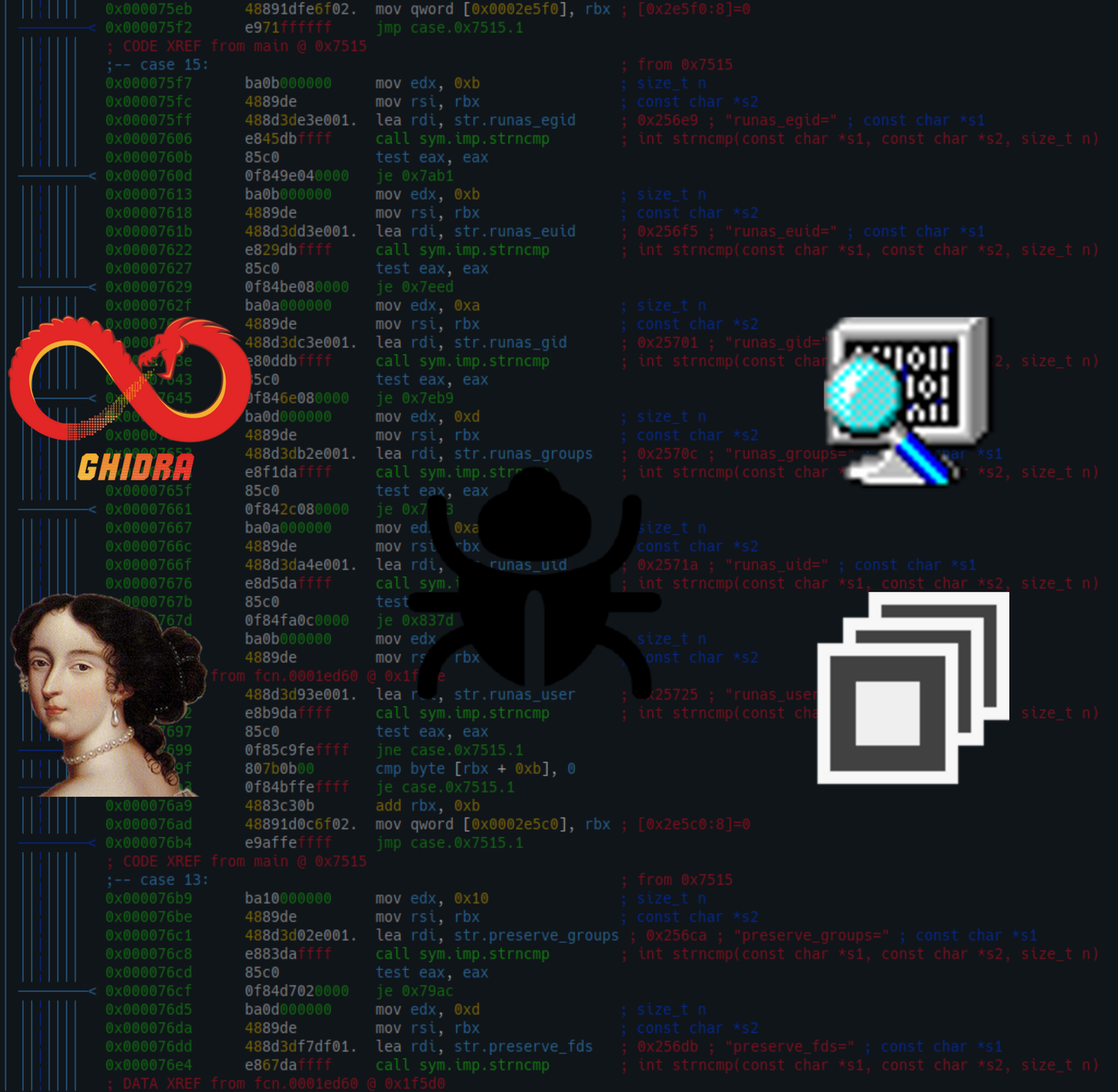
How to get the job done?

Using Tools & Brain ;)

The first thing is available to download
but, the second .... Unfortunately not!

RE Tools

- Decompiler
- Disassembler
- Debugger



Differences Between OSs & Compilers

Hello World Program

```
0000000000001135 <main>:
#include <stdio.h>

void main(){
    1135:       55                      push   rbp
    1136:       48 89 e5                mov    rbp,rsp

    printf("Hello World");
    1139:       48 8d 3d c4 0e 00 00    lea    rdi,[rip+0xec4]
    1140:       b8 00 00 00 00          mov    eax,0x0
    1145:       e8 e6 fe ff ff          call   1030 <printf@plt>

}
```

Linux w/ gcc

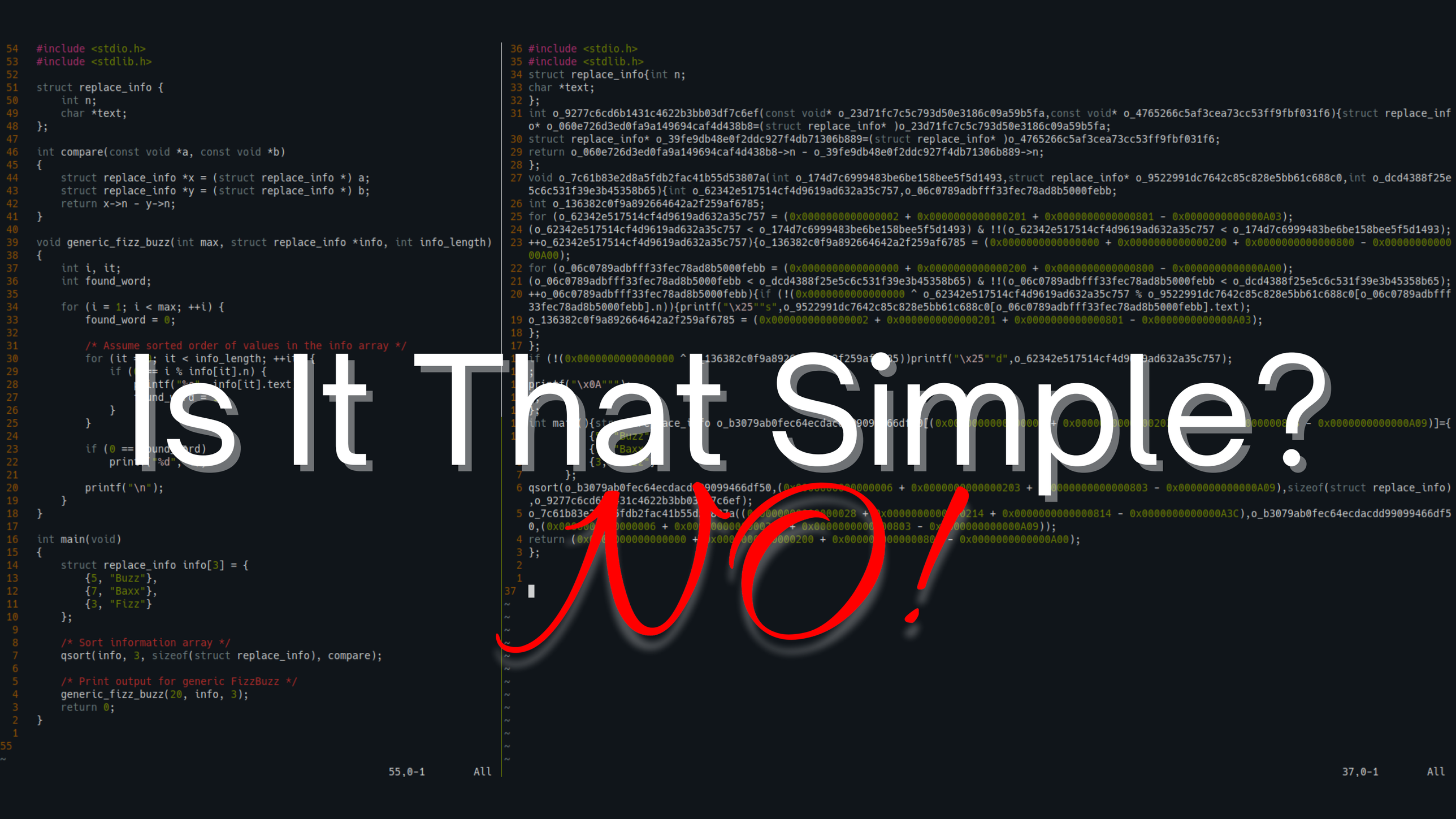
```
0x000075eb      48891dfe6f02.  mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x000075f2      e971ffffff     jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:                                              ; from 0x7515
0x000075f7      ba0b000000     mov edx, 0xb                ; size_t n
0x000075fc      4889de         mov rsi, rbx                ; const char *s2
0x000075ff      488d3de3e001.  lea rdi, str.runas_egid     ; 0x256e9 ; "runas_egid=" ; const char *s1
0x00007606      e845dbffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const char *s2, size_t n)
0x0000760b      85c0           test eax, eax
0x0000760d      0f849e040000   je 0x7ab1
0x00007613      ba0b000000     mov edx, 0xb                ; size_t n
0x00007618      4889de         mov rsi, rbx                ; const char *s2
0x0000761b      488d3dd3e001.  lea rdi, str.runas_euid     ; 0x256f5 ; "runas_euid=" ; const char *s1
0x00007622      e829dbffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const char *s2, size_t n)
0x00007627      85c0           test eax, eax
0x00007629      0f84be080000   je 0x7eed
0x0000762f      ba0a000000     mov edx, 0xa                ; size_t n
0x00007634      4889de         mov rsi, rbx                ; const char *s2
0x00007637      488d3dc3e001.  lea rdi, str.runas_gid      ; 0x25701 ; "runas_gid=" ; const char *s1
0x0000763e      e80ddbffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const char *s2, size_t n)
```

```
--- C:\Users\r4kaa\Source\Repos\HelloWorld\HelloWorld\HelloWorld.c -------------
#include <stdio.h>


void main() {
00007FF791121820  push         rbp
00007FF791121822  push         rdi
00007FF791121823  sub          rsp,0E8h
00007FF79112182A  lea          rbp,[rsp+20h]

    printf("Hello World");
00007FF79112182F  lea          rcx,[string "Hello World" (07FF791129C28h)]
00007FF791121836  call         printf (07FF79112118Bh)


}
```

```
0x000076b4      e9affeffff     jmp case.0x7515.1
; CODE XREF from main @ 0x7[illegible]
;-- case 13:                                              ; from 0x7[illegible]
0x000076b9      ba10000000     mov edx, 0x10               ; size_t n
0x000076be      4889de         mov rsi, rbx                ; const char *s2
0x000076c1      488d3d02e001.  lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x000076c8      e883daffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const char *s2, size_t n)
0x000076cd      85c0           test eax, eax
0x000076cf      0f84d7020000   je 0x79ac
0x000076d5      ba0d000000     mov edx, 0xd                ; size_t n
0x000076da      4889de         mov rsi, rbx                ; const char *s2
0x000076dd      488d3df7df01.  lea rdi, str.preserve_fds   ; 0x256db ; "preserve_fds=" ; const char *s1
0x000076e4      e867daffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

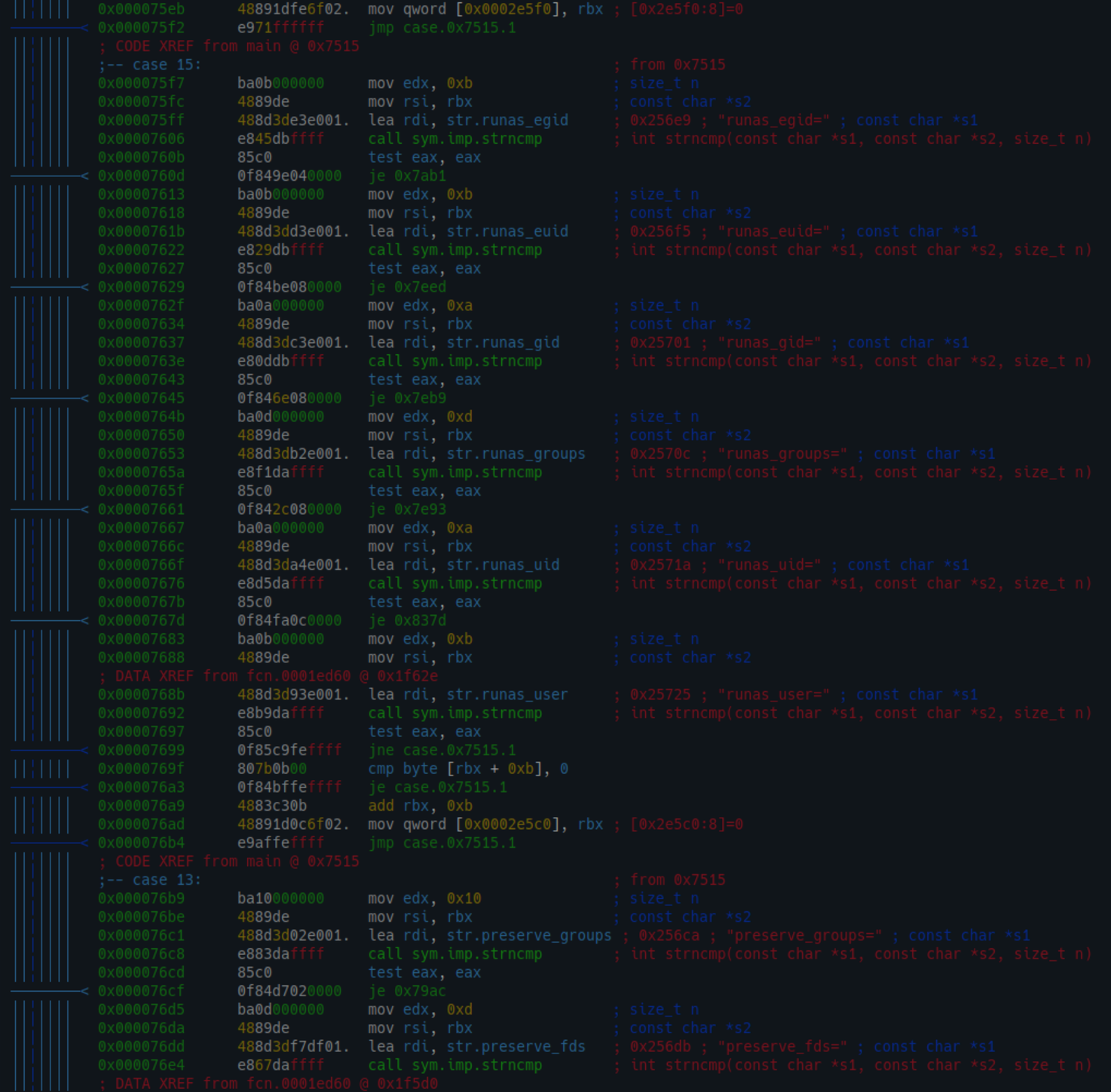
Win w/ MS Visual C++

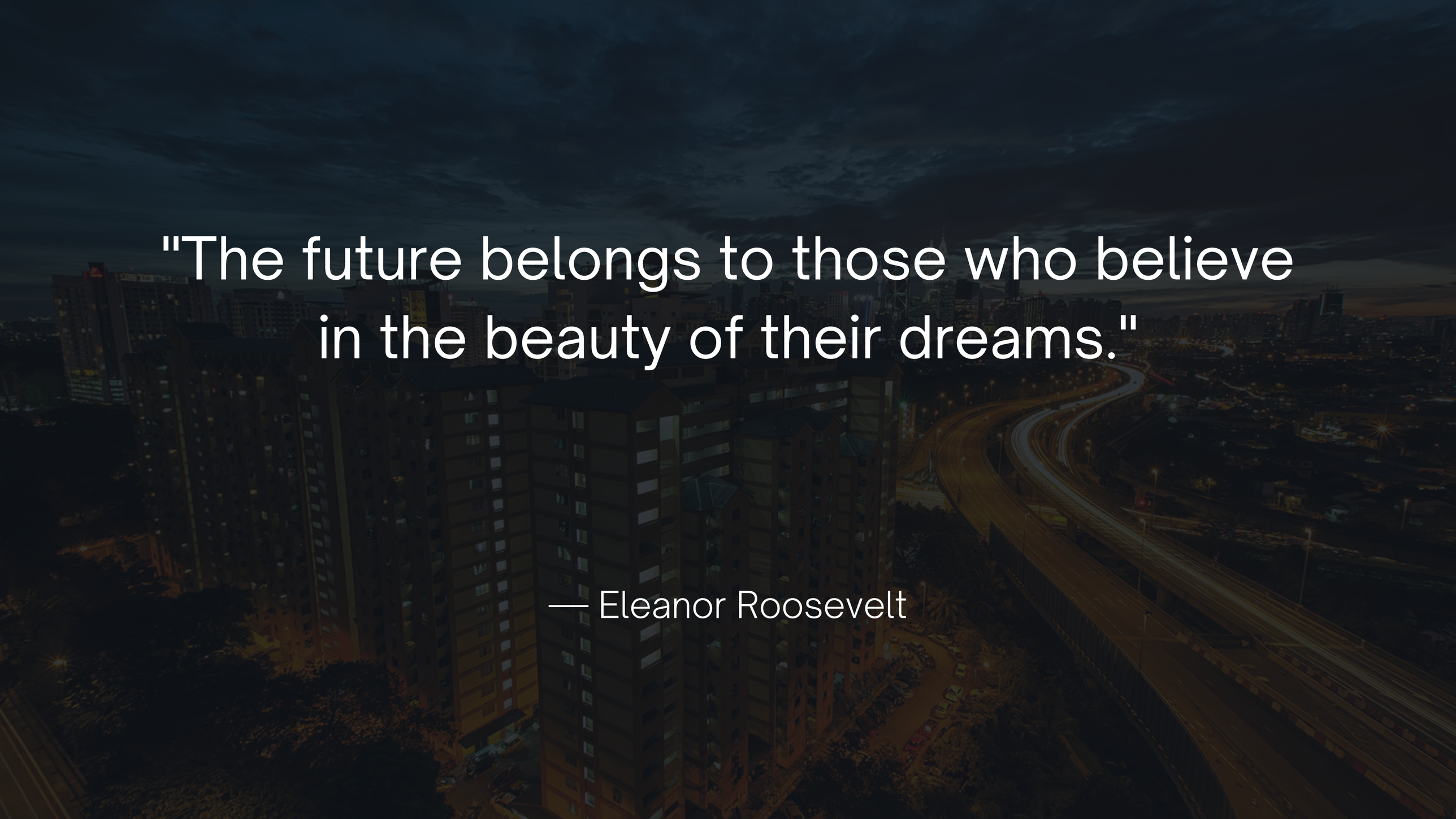

Demo Time :)


Is It That Simple?
NO!

RE Obstacles

- Code Obfuscation
- Packers
- Anti-Reverse
- Anti-Debug
- Anti-Sandbox
- etc...




"The future belongs to those who believe
in the beauty of their dreams."
— Eleanor Roosevelt


Program received signal SIGSEGV, Segmentation fault.
[----------registers----------]
RAX: 0xffffffff
RBX: 0x0
RCX: 0x5a1
RDX: 0x5a1
RSI: 0x1
RDI: 0x5a1
RBP: 0x0
RSP: 0x7fffffffdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>: mov r14,rax)
RIP: 0x1337
R8 : 0x20000000 ('')
R9 : 0x0
R10: 0x7ffff7ff4e22 ("MALLOC_CHECK_")
R11: 0x1
R12: 0x7ffff7ff435a --> 0x706e203d3d206900 ('')
R13: 0x0
R14: 0x4f4c5f
R15: 0x10
EFLAGS: 0x10
[----------code----------]
Invalid $PC address: 0x1337
[----------stack----------]
0000| 0x7fffffffdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>: mov r14,rax)
0008| 0x7fffffffdcf0 --> 0x0
0016| 0x7fffffffdcf8 --> 0x7ffff7ff435a --> 0x706e203d3d206900 ('')
0024| 0x7fffffffdd00 --> 0x20000000 ('')
0032| 0x7fffffffdd08 --> 0x100
0040| 0x7fffffffdd10 --> 0x1
0048| 0x7fffffffdd18 --> 0x2000000000000000 ('')
0056| 0x7fffffffdd20 --> 0x0
[----------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000001337 in ?? ()
gdb-peda$

Thank You!