

# 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

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007643 85c0 test eax, eax
0x000007645 0f846e080000 je 0x7eb9
0x00000764b ba0d000000 mov edx, 0xd
0x000007650 4889de mov rsi, rbx
0x000007654 488d3db2e001. lea rdi, str.runas_groups
0x00000765f e8f1daffff call sym.imp.strncmp
0x000007661 85c0 test eax, eax
0x000007667 0f8410000000 je 0x7e02
0x00000766c 4889de mov rsi, rbx
0x00000766f 488d3da4e001. lea rdi, str.runas_uid
0x000007676 e8d50a1fffff call sym.imp.strncmp
0x00000767b 85c0 test eax, eax
0x00000767d 0f84fa0c0000 je 0x837d
0x000007683 ba0b000000 mov edx, 0xb
0x000007688 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b 488d3d93e001. lea rdi, str.runas_user
0x000007692 e8b9daffff call sym.imp.strncmp
0x000007697 85c0 test eax, eax
0x000007699 0f85c9feffff jne case.0x7515.1
0x00000769f 807b0b00 cmp byte [rbx + 0xb], 0
0x0000076a3 0f84bfff feffff je case.0x7515.1
0x0000076a9 4883c30b add rbx, 0xb
0x0000076ad 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd 85c0 test eax, eax
0x0000076cf 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# 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

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007645 85c0 test eax, eax
0x00000764b 0f846e080000 je 0x7eb9
0x000007650 ba0d000000 mov edx, 0xd
0x000007653 4889de mov rsi, rbx
0x00000765a 488d3db3e001. lea rdi, str.runas_groups
0x00000765f e8f1daffff call sym.imp.strncmp
0x000007661 85c0 test eax, eax
0x000007667 ba0a000000 je 0x7e93
0x00000766c 4889de mov rsi, rbx
0x00000766f 488d3da4e001. lea rdi, str.runas_uid
0x000007673 e8d5daffff call sym.imp.strncmp
0x00000767b 85c0 test eax, eax
0x00000767d 0f84fa0c0000 je 0x837d
0x000007680 ba0b000000 mov edx, 0xb
0x00000768b 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x000007692 488d3d93e001. lea rdi, str.runas_user
0x000007697 e8b9daffff call sym.imp.strncmp
0x000007699 85c0 test eax, eax
0x00000769f 0f85c9feffff jne case.0x7515.1
0x0000076a3 807b0b00 cmp byte [rbx + 0xb], 0
0x0000076a9 0f84bfff feffff je case.0x7515.1
0x0000076ad 4883c30b add rbx, 0xb
0x0000076b4 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd 85c0 test eax, eax
0x0000076cf 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# 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.



```
0x000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x000075f7 ba0b000000 mov edx, 0xb
0x000075fc 4889de mov rsi, rbx
0x000075ff 488d3de3e001. lea rdi, str.runas_egid
0x00007606 e845dbffff call sym.imp.strncmp
0x0000760b 85c0 test eax, eax
0x0000760d 0f849e040000 je 0x7ab1
0x00007613 ba0b000000 mov edx, 0xb
0x00007618 4889de mov rsi, rbx
0x0000761b 488d3dd3e001. lea rdi, str.runas_euid
0x00007622 e829dbffff call sym.imp.strncmp
0x00007627 85c0 test eax, eax
0x00007629 0f84be080000 je 0x7eed
0x0000762f ba0a000000 mov edx, 0xa
0x00007634 4889de mov rsi, rbx
0x00007637 488d3dc3e001. lea rdi, str.runas_gid
0x0000763e e80ddbffff call sym.imp.strncmp
0x00007642 85c0 test eax, eax
0x00007643 0f848e080000 je 0x7eb9
0x0000764b ba0d000000 mov edx, 0xd
0x00007650 4889de mov rsi, rbx
0x00007654 488d3db2e001. lea rdi, str.runas_group
0x0000765a e8f1daffff call sym.imp.strncmp
0x0000765f 85c0 test eax, eax
0x00007661 0f842c080000 je 0x7e93
0x00007667 ba0a000000 mov edx, 0xa
0x0000766c 4889de mov rsi, rbx
0x0000766f 488d3db2e001. lea rdi, str.runas_uid
0x00007676 e8d5daffff call sym.imp.strncmp
0x0000767b 85c0 test eax, eax
0x0000767d 0f848e080000 je 0x7837d
0x00007683 ba0b000000 mov edx, 0xb
0x00007688 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x0000768b 488d3d93e001. lea rdi, str.runas_user
0x00007692 e8b9daffff call sym.imp.strncmp
0x00007697 85c0 test eax, eax
0x00007699 0f85c9f1. jne case.0x7515.1
0x0000769f 807b0b00. cmp byte [rbx + 0xb], 0
0x000076a3 0f84bf. je case.0x7515.1
0x000076a9 4883c3. add rbx, 0xb
0x000076ad 48891d. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x000076b4 e9affef1. jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x000076b9 ba10000000 mov edx, 0x10
0x000076be 4889de mov rsi, rbx
0x000076c1 488d3d02e001. lea rdi, str.preserve_groups
0x000076c8 e883daffff call sym.imp.strncmp
0x000076cd 85c0 test eax, eax
0x000076cf 0f84d7020000 je 0x79ac
0x000076d5 ba0d000000 mov edx, 0xd
0x000076da 4889de mov rsi, rbx
0x000076dd 488d3df7df01. lea rdi, str.preserve_fds
0x000076e4 e867daffff call sym.imp.strncmp
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
0x000076e5 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x000076f2 e971fffff jmp case.0x7515.1
; from 0x7515
; size_t n
; const char *s2
; 0x256e9 ; "runas_egid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x256f5 ; "runas_euid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x25701 ; "runas_gid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x2570c ; "runas_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x25725 ; "runas_uid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x2572a ; "preserve_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
;
; size_t n
; const char *s2
; 0x256db ; "preserve_fds=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
```

# 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.

```
0x0000075eb    48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2    e971fffff   jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7    ba0b000000  mov edx, 0xb
0x0000075fc    4889de     mov rsi, rbx
0x0000075ff    488d3de3e001. lea rdi, str.runas_egid
0x000007606    e845dbffff  call sym.imp.strncmp
0x00000760b    85c0       test eax, eax
0x00000760d    0f849e040000 je 0x7ab1
0x000007613    ba0b000000  mov edx, 0xb
0x000007618    4889de     mov rsi, rbx
0x00000761b    488d3dd3e001. lea rdi, str.runas_euid
0x000007622    e829dbffff  call sym.imp.strncmp
0x000007627    85c0       test eax, eax
0x000007629    0f84be080000 je 0x7eed
0x00000762f    ba0a000000  mov edx, 0xa
0x000007634    4889de     mov rsi, rbx
0x000007637    488d3dc3e001. lea rdi, str.runas_gid
0x00000763e    e80ddbffff  call sym.imp.strncmp
0x000007643    85c0       test eax, eax
0x000007645    0f846e080000 je 0x7eb9
0x000007650    ba0a000000  mov edx, 0xa
0x000007653    4889de     mov rsi, rbx
0x00000765a    488d3db2e001. lea rdi, str.runas_groups
0x00000765f    e8f1daffff  call sym.imp.strncmp
0x000007661    85c0       test eax, eax
0x000007666    0f842c080000 je 0x7e93
0x000007667    ba0a000000  mov edx, 0xa
0x000007668    4889de     mov rsi, rbx
0x000007669    488d3a4e001. lea rdi, str.runas_uid
0x000007676    e8d5daffff  call sym.imp.strncmp
0x00000767b    85c0       test eax, eax
0x00000767d    0f84fa0c0000 je 0x837d
0x000007683    ba0b000000  mov edx, 0xb
0x000007688    4889de     mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b    488d3d93e001. lea rdi, str.runas_user
0x000007692    e8b9daffff  call sym.imp.strncmp
0x000007697    85c0       test eax, eax
0x000007699    0f85c9feffff jne case.0x7515.1
0x00000769f    807b0b00   cmp byte [rbx + 0xb], 0
0x0000076a3    0f84bfffefee je case.0x7515.1
0x0000076a9    4883c30b   add rbx, 0xb
0x0000076ad    48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4    e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9    ba10000000  mov edx, 0x10
0x0000076be    4889de     mov rsi, rbx
0x0000076c1    488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8    e883daffff  call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd    85c0       test eax, eax
0x0000076cf    0f84d7020000 je 0x79ac
0x0000076d5    ba0d000000  mov edx, 0xd
0x0000076da    4889de     mov rsi, rbx
0x0000076dd    488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4    e867daffff  call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

```

54 #include <stdio.h>
53 #include <stdlib.h>
52
51 struct replace_info {
50     int n;
49     char *text;
48 };
47
46 int compare(const void *a, const void *b)
45 {
44     struct replace_info *x = (struct replace_info *) a;
43     struct replace_info *y = (struct replace_info *) b;
42     return x->n - y->n;
41 }
40
39 void generic_fizz_buzz(int max, struct replace_info *info, int info_length)
38 {
37     int i, it;
36     int found_word;
35
34     for (i = 1; i < max; ++i) {
33         found_word = 0;
32
31         /* Assume sorted order of values in the info array */
30         for (it = 0; it < info_length; ++it) {
39             if (0 == i % info[it].n){
38                 printf("%s", info[it].text);
37                 found_word = 1;
36             }
35         }
34         if (0 == found_word)
33             printf("%d", i);
32
31         printf("\n");
30     }
39 }
38
37 int main(void)
36 {
35     struct replace_info info[3] = {
34         {5, "Buzz"},  

33         {7, "Baxx"},  

32         {3, "Fizz"}  

31     };
30
38     /* Sort information array */
37     qsort(info, 3, sizeof(struct replace_info), compare);
36
35     /* Print output for generic FizzBuzz */
34     generic_fizz_buzz(20, info, 3);
33     return 0;
32 }
31

```

```

36 #include <stdio.h>
35 #include <stdlib.h>
34 struct replace_info{int n;
33 char *text;
32 };
31 int o_9277c6cd6b1431c4622b3bb03df7c6ef(const void* o_23d71fc7c5c793d50e3186c09a59b5fa,const void* o_4765266c5af3cea73cc53ff9fbf031f6){struct replace_inf
o* o_060e726d3ed0fa9a149694caf4d438b8=(struct replace_info*) o_23d71fc7c5c793d50e3186c09a59b5fa;
30 struct replace_info* o_39fe9db48e0f2ddc927f4db71306b889=(struct replace_info*) o_4765266c5af3cea73cc53ff9fbf031f6;
29 return o_060e726d3ed0fa9a149694caf4d438b8->n - o_39fe9db48e0f2ddc927f4db71306b889->n;
28 };
27 void o_7c61b83e2d8a5fdb2fac41b55d53807a(int o_174d7c6999483be6be158bee5f5d1493,struct replace_info* o_9522991dc7642c85c828e5bb61c688c0,int o_dcd4388f25e
5c6c531f39e3b45358b65){int o_62342e517514cf4d9619ad632a35c757,o_06c0789adbfff33fec78ad8b5000feb;
26 int o_136382c0f9a892664642a2f259af6785;
25 for (o_62342e517514cf4d9619ad632a35c757 = (0x0000000000000002 + 0x0000000000000000201 + 0x0000000000000000801 - 0x0000000000000000A03);
24 (o_62342e517514cf4d9619ad632a35c757 < o_174d7c6999483be6be158bee5f5d1493) & !(o_62342e517514cf4d9619ad632a35c757 < o_174d7c6999483be6be158bee5f5d1493);
23 ++o_62342e517514cf4d9619ad632a35c757){o_136382c0f9a892664642a2f259af6785 = (0x0000000000000000 + 0x0000000000000000200 + 0x0000000000000000800 - 0x0000000000000000A00);
22 for (o_06c0789adbfff33fec78ad8b5000feb = (0x0000000000000000 + 0x0000000000000000200 + 0x0000000000000000800 - 0x0000000000000000A00);
21 (o_06c0789adbfff33fec78ad8b5000feb < o_dcd4388f25e5c6c531f39e3b45358b65) & !(o_06c0789adbfff33fec78ad8b5000feb < o_dcd4388f25e5c6c531f39e3b45358b65);
20 ++o_06c0789adbfff33fec78ad8b5000feb){if (!(0x0000000000000000 ^ o_62342e517514cf4d9619ad632a35c757 % o_9522991dc7642c85c828e5bb61c688c0[o_06c0789adbfff
33fec78ad8b5000feb].n)){printf("\x25 \"%s\",o_9522991dc7642c85c828e5bb61c688c0[o_06c0789adbfff33fec78ad8b5000feb].text);
19 o_136382c0f9a892664642a2f259af6785 = (0x0000000000000002 + 0x0000000000000000201 + 0x0000000000000000801 - 0x0000000000000000A03);
18 };
17 };
16 if (!(0x0000000000000000 ^ o_136382c0f9a892664642a2f259af6785))printf("\x25 \"%d\",o_62342e517514cf4d9619ad632a35c757);
15 ;
14 printf("\x0A\"");
13 };
12 };
11 void main(){struct replace_info o_b3079ab0fec64ecdacdd99099466df50[(0x0000000000000006 + 0x0000000000000000203 + 0x0000000000000000803 - 0x0000000000000000A09)={
10 ,o_9277c6cd6b1431c4622b3bb03df7c6ef];
9 o_7c61b83e2d8a5fdb2fac41b55d53807a((0x0000000000000028 + 0x00000000000000214 + 0x00000000000000814 - 0x00000000000000A3C),o_b3079ab0fec64ecdacdd99099466df5
8 ,0,(0x0000000000000006 + 0x00000000000000203 + 0x00000000000000803 - 0x00000000000000A09));
7 4 return (0x0000000000000000 + 0x0000000000000000800 - 0x0000000000000000A03);
6 3 };
5 2 };
4 1 };
3 0 };
2 1 };
1 0 };

```

# Do We Really Need ASM?

*Depends!*

# Technical Fields of RE

- Hardware

- Software

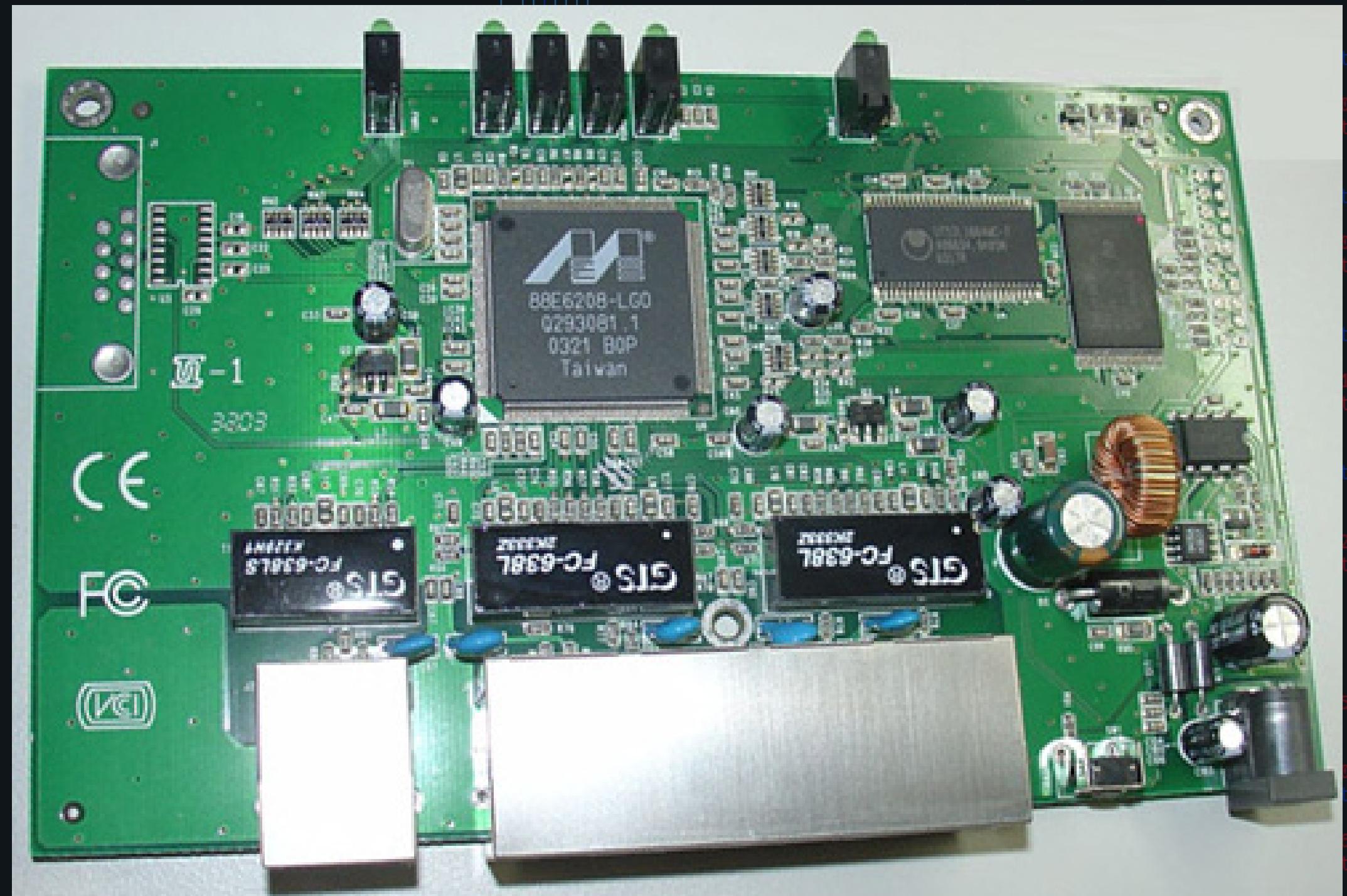
```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007643 85c0 test eax, eax
0x000007645 0f846e080000 je 0x7eb9
0x00000764b ba0d000000 mov edx, 0xd
0x000007650 4889de mov rsi, rbx
0x000007653 488d3db2e001. lea rdi, str.runas_groups
0x00000765a e8f1daffff call sym.imp.strncmp
0x00000765f 85c0 test eax, eax
0x000007661 0f842c080000 je 0x7e93
0x000007667 ba0a000000 mov edx, 0xa
0x00000766c 4889de mov rsi, rbx
0x00000766f 488d3da4e001. lea rdi, str.runas_uid
0x000007676 e8d5daffff call sym.imp.strncmp
0x00000767b 85c0 test eax, eax
0x00000767d 0f84fa0c0000 je 0x837d
0x000007683 ba0b000000 mov edx, 0xb
0x000007688 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b 488d3d93e001. lea rdi, str.runas_user
0x000007692 e8b9daffff call sym.imp.strncmp
0x000007697 85c0 test eax, eax
0x000007699 0f85c9feffff jne case.0x7515.1
0x00000769f 807b0b00 cmp byte [rbx + 0xb], 0
0x0000076a3 0f84bfff feffff je case.0x7515.1
0x0000076a9 4883c30b add rbx, 0xb
0x0000076ad 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affefffe jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd 85c0 test eax, eax
0x0000076cf 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# Hardware RE

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

```
0x0000075eb    48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2    e971fffff   jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7    ba0b000000  mov edx, 0xb
0x0000075fc    4889de     mov rsi, rbx
0x0000075ff    488d3de3e001. lea rdi, str.runas_egid
0x000007606    e845dbffff  call sym.imp.strncmp
0x00000760b    85c0       test eax, eax
0x00000760d    0f849e040000 je 0x7ab1
0x000007613    ba0b000000  mov edx, 0xb
0x000007618    4889de     mov rsi, rbx
0x00000761b    488d3dd3e001. lea rdi, str.runas_euid
0x000007622    e829dbffff  call sym.imp.strncmp
0x000007627    85c0       test eax, eax
0x000007629    0f84be080000 je 0x7eed
0x00000762f    ba0a000000  mov edx, 0xa
0x000007634    4889de     mov rsi, rbx
0x000007637    488d3dc3e001. lea rdi, str.runas_gid
0x00000763e    e80ddbffff  call sym.imp.strncmp
0x000007643    85c0       test eax, eax
0x000007645    0f846e080000 je 0x7eb9
0x00000764b    ba0d000000  mov edx, 0xd
0x000007650    4889de     mov rsi, rbx
0x000007653    488d3db2e001. lea rdi, str.runas_groups
0x00000765a    e8f1daffff  call sym.imp.strncmp
0x00000765f    85c0       test eax, eax
0x000007661    0f842c080000 je 0x7e93
0x000007666    ba0e000000  mov edx, 0xa
0x00000766c    4889de     mov rsi, rbx
0x00000766f    488d3ca4e001. lea rdi, str.runas_uid
0x000007676    e8d5daffff  call sym.imp.strncmp
0x00000767b    85c0       test eax, eax
0x00000767d    0f84fa0c0000 je 0x837d
0x000007683    ba0b000000  mov edx, 0xb
0x000007688    4889de     mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b    488d3d93e001. lea rdi, str.runas_user
0x000007692    e8b9daffff  call sym.imp.strncmp
0x000007697    85c0       test eax, eax
0x000007699    0f85c9feffff jne case.0x7515.1
0x00000769f    807b0b00    cmp byte [rbx + 0xb], 0
0x0000076a3    0f84bfffef  je case.0x7515.1
0x0000076a9    4883c30b    add rbx, 0xb
0x0000076ad    48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4    e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9    ba10000000  mov edx, 0x10
0x0000076be    4889de     mov rsi, rbx
0x0000076c1    488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8    e883daffff  call sym.imp.strncmp      ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd    85c0       test eax, eax
0x0000076cf    0f84d7020000 je 0x79ac
0x0000076d5    ba0d000000  mov edx, 0xd
0x0000076da    4889de     mov rsi, rbx
0x0000076dd    488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4    e867daffff  call sym.imp.strncmp      ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# Printed Circuit Board (PCB) RE.



```
0x000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x000075f7 ba0b000000 mov edx, 0xb
0x000075fc 4889de mov rsi, rbx
0x000075ff 488d3de3e001. lea rdi, str.runas_egid
0x00007606 e845dbffff call sym.imp.strncmp
0x0000760b 85c0 test eax, eax
0x0000760d @f849e040000 je 0x7ab1
0x00007610 ba0b000000 mov edx, 0xb
0x00007618 4889de mov rsi, rbx
0x0000761b 488d3de001. lea rdi, str.runas_euid
0x00007622 e829dbffff call sym.imp.strncmp
; size_t n
; const char *s2
; 0x256e9 ; "runas_egid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x256f5 ; "runas_euid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 01 ; "runas_gid=" ; const char *s1
; strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0c ; "runas_groups=" ; const char *s1
; strcmp(const char *s1, const char *s2, size_t n)

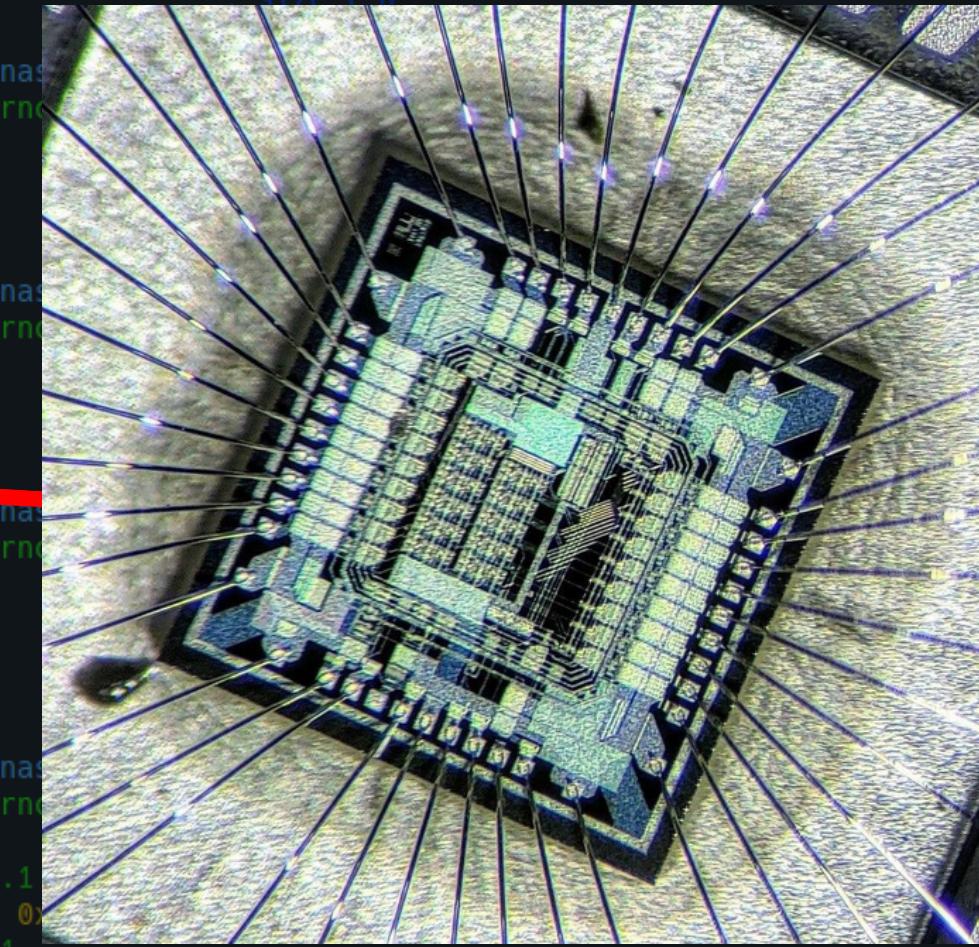
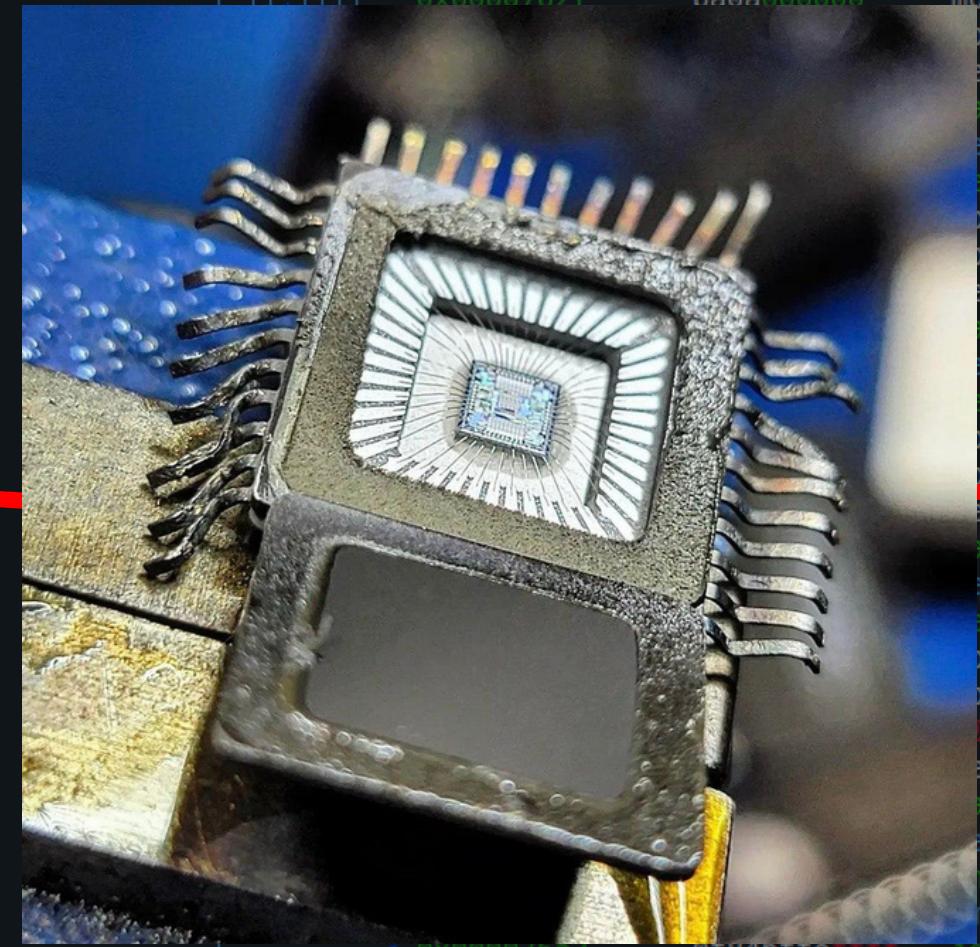
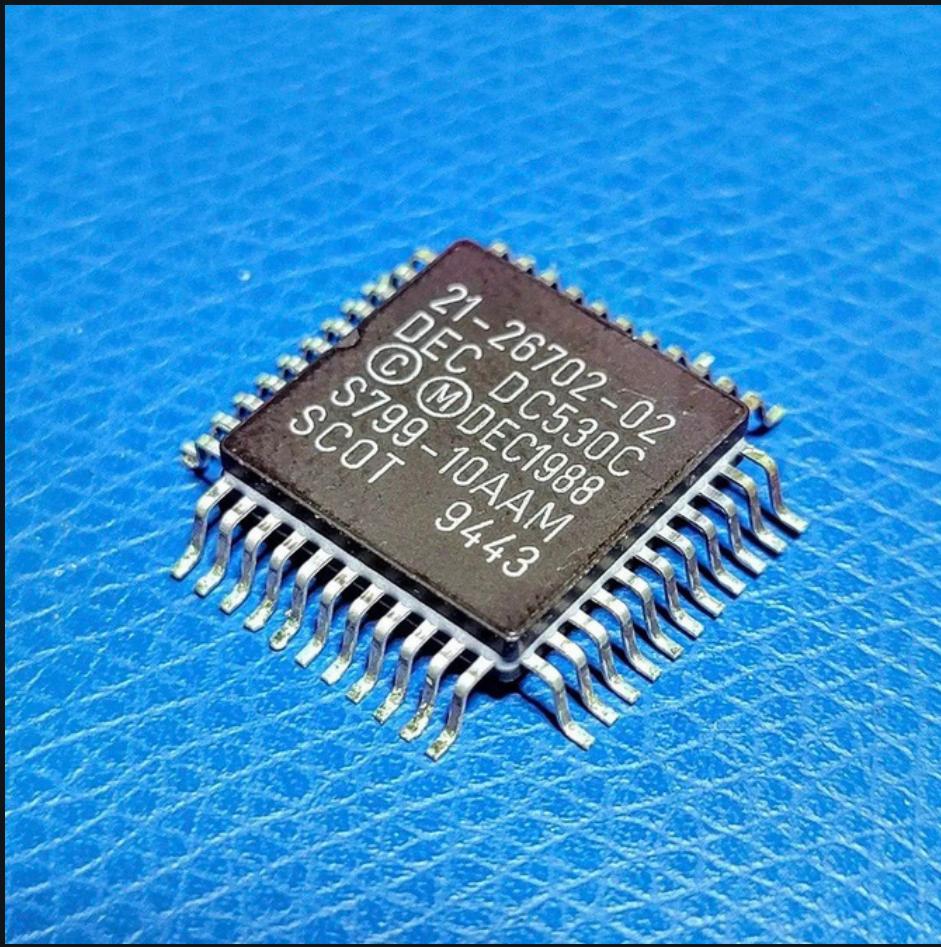
; size_t n
; const char *s2
; 1a ; "runas_uid=" ; const char *s1
; strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 25 ; "runas_user=" ; const char *s1
; strcmp(const char *s1, const char *s2, size_t n)

; DATA XREF from fcn.0001ed60 @ 0x1f5d0
0x000076cf 0f84d7020000 je 0x79ac
0x000076d5 ba0d000000 mov edx, 0xd
0x000076da 4889de mov rsi, rbx
0x000076dd 488d3df7df01. lea rdi, str.preserve_fds
0x000076e4 e867daffff call sym.imp.strncmp
; size_t n
; const char *s2
; 0x256db ; "preserve_fds=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; DATA XREF from fcn.0001ed60 @ 0x1f5d0
0x000076f0 0f84d7020000 je 0x79ac
0x000076f5 ba0d000000 mov edx, 0xd
0x000076f8 4889de mov rsi, rbx
0x000076f9 488d3df7df01. lea rdi, str.preserve_fds
0x000076fa e867daffff call sym.imp.strncmp
; size_t n
; const char *s2
; 0x256ca ; "preserve_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
```

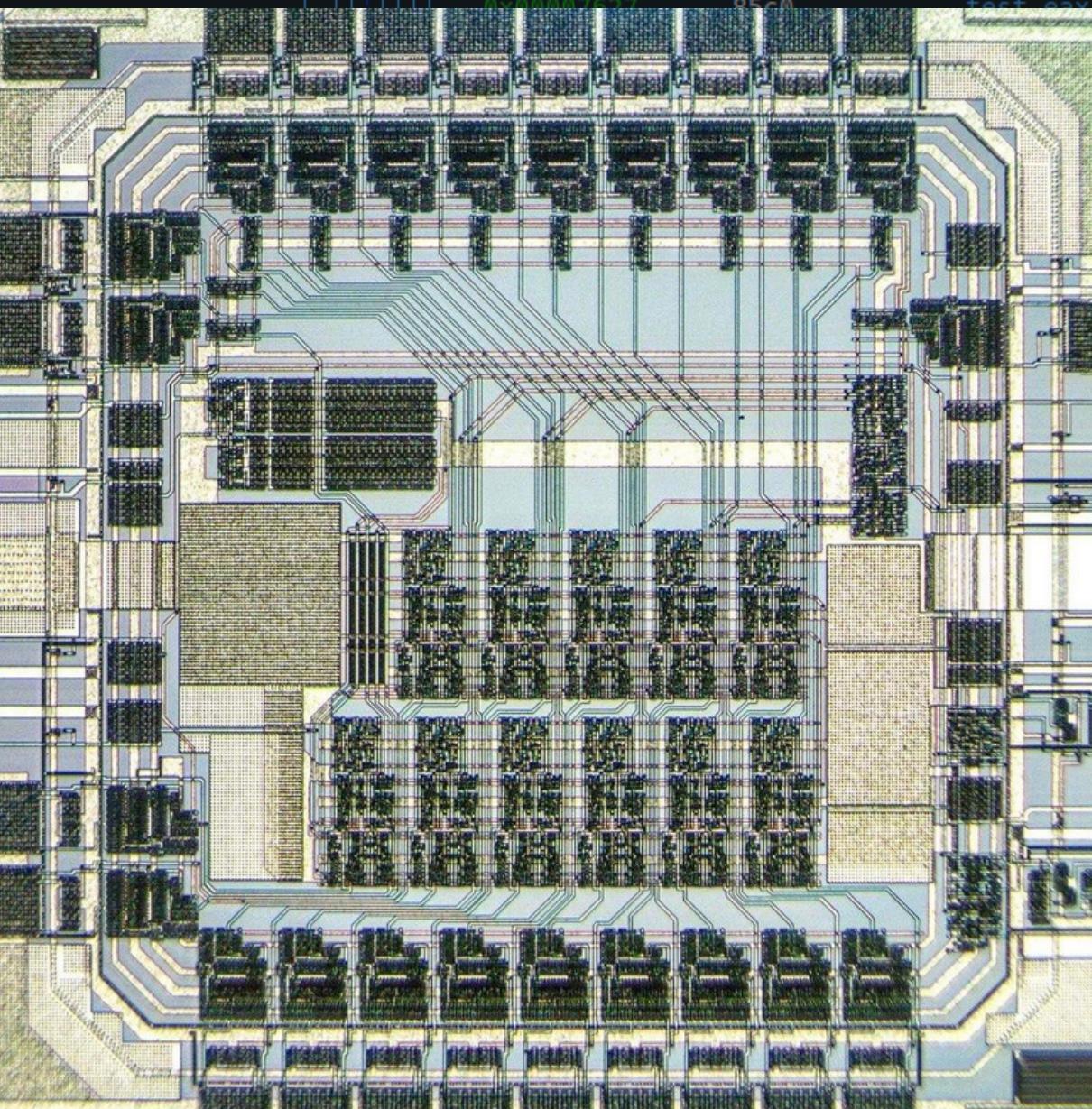
# VLSI RE



CREDITS TO @EVILMONKEYDESIGNZ

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
; size_t n
; const char *s2
; 0x256e9 ; "runas_egid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
; CODE XREF from main @ 0x7515
;-- case 15:
0x000007633 0f84d1feffff je case.0x7515.1
0x0000076a9 4883c30b add rbx, 0xb
0x0000076ad 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c0 488d3de3e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076cd e813da4f7f call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cf 85c0 test eax, eax
0x0000076d0 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# VLSI RE (Con.)



CREDITS TO @EVILMONKEYDESIGNZ

```
0x000075eb    48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x000075f2    e971ffff    jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x000075f7    ba0b000000  mov edx, 0xb
0x000075fc    4889de     mov rsi, rbx
0x000075ff    488d3de3e001. lea rdi, str.runas_egid
0x00007606    e845dbffff  call sym.imp.strncmp
0x0000760b    85c0       test eax, eax
0x0000760d    0f849e040000 je 0x7ab1
0x00007613    ba0b000000  mov edx, 0xb
0x00007618    4889de     mov rsi, rbx
0x0000761b    488d3dd3e001. lea rdi, str.runas_euid
0x00007622    e829dbffff  call sym.imp.strncmp
0x00007627    85c0       test eax, eax
; size_t n
; const char *s2
; 0x256e9 ; "runas_egid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x0000762a    0f849e040000 je 0x7ab1
0x0000762d    ba0b000000  mov edx, 0xb
0x00007632    4889de     mov rsi, rbx
0x00007635    488d3de3e001. lea rdi, str.runas_gid
0x0000763b    e845dbffff  call sym.imp.strncmp
0x0000763f    85c0       test eax, eax
; size_t n
; const char *s2
; 0x25701 ; "runas_gid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x00007642    0f849e040000 je 0x7ab1
0x00007645    ba0b000000  mov edx, 0xb
0x0000764a    4889de     mov rsi, rbx
0x0000764d    488d3de3e001. lea rdi, str.runas_groups
0x00007653    e845dbffff  call sym.imp.strncmp
0x0000765b    85c0       test eax, eax
; size_t n
; const char *s2
; 0x2570c ; "runas_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x0000765e    0f849e040000 je 0x7ab1
0x00007661    ba0b000000  mov edx, 0xb
0x00007666    4889de     mov rsi, rbx
0x00007669    488d3de3e001. lea rdi, str.runas_uid
0x00007675    e845dbffff  call sym.imp.strncmp
0x0000767d    85c0       test eax, eax
; size_t n
; const char *s2
; 0x2571a ; "runas_uid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x00007680    0f849e040000 je 0x7ab1
0x00007683    ba0b000000  mov edx, 0xb
0x00007688    4889de     mov rsi, rbx
0x0000768b    488d3de3e001. lea rdi, str.runas_user
0x00007691    e845dbffff  call sym.imp.strncmp
0x00007699    85c0       test eax, eax
0x0000769f    0f849e040000 je 0x7515.1
0x000076a2    ba0b000000  mov edx, [rbx + 0xb], 0
0x000076a5    4889de     mov rsi, rbx
0x000076a8    488d3de3e001. lea rdi, str.preserve_groups
0x000076b4    e845dbffff  call sym.imp.strncmp
0x000076b8    85c0       test eax, eax
; size_t n
; const char *s2
; 0x256ca ; "preserve_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x000076b9    ba10000000  mov edx, 0x10
0x000076be    4889de     mov rsi, rbx
0x000076c1    488d3de3e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x000076cd    e845dbffff  call sym.imp.strncmp
0x000076cf    85c0       test eax, eax
; from 0x7515
; size_t n
; const char *s2
; 0x256ca ; "preserve_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

0x000076d5    ba0d000000  mov edx, 0xd
0x000076da    4889de     mov rsi, rbx
0x000076dd    488d3df7df01. lea rdi, str.preserve_fds
0x000076e4    e867daffff  call sym.imp.strncmp
0x000076e8    0f84d7020000 je 0x79ac
; size_t n
; const char *s2
; 0x256db ; "preserve_fds=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# Firmware Extraction



0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0  
0x0000075f2 e971fffff jmp case.0x7515.1  
; CODE XREF from main @ 0x7515  
;-- case 15:  
0x0000075f7 ba0b000000 mov edx, 0xb  
0x0000075fc 4889de mov rsi, rbx  
0x0000075ff 488d3de3e001. lea rdi, str.runas\_egid  
0x000007606 e845dbffff call sym.imp.strncmp  
0x00000760b 85c0 test eax, eax  
0x00000760d 0f849e040000 je 0x7ab1  
0x000007613 ba0b000000 mov edx, 0xb  
0x000007618 4889de mov rsi, rbx  
0x00000761b 488d3dd3e001. lea rdi, str.runas\_euid  
0x000007622 e829dhffff call sym.imp.strncmp  
; from 0x7515  
; size\_t n  
; const char \*s2  
; 0x256e9 ; "runas\_egid=" ; const char \*s1  
; int strcmp(const char \*s1, const char \*s2, size\_t n)  
; size\_t n  
; const char \*s2  
; 0x256f5 ; "runas\_euid=" ; const char \*s1  
; int strcmp(const char \*s1, const char \*s2, size\_t n)

EZP2019+ USB High Speed Programmer

File(F) Buffer(B) Operation(O) Device(Y) Language(L) Help(H)

Open Save Read ERASE Write Verify AUTO Bank FILL Device About

Chip Type: SPI\_FLASH  
Manufacturer: WINBOND  
Name: W25Q128

Test Find

Size: 16MB  
Pagesize: 256  
Speed: 375KHz

Auto Options: Erase (checked), Program (checked), Verify (checked)

AUTO

0001 0203 0405 0607 0809 0A0B 0C0D 0EOF 0123456789ABCDEF  
E51F F004 2273 CD33 E51F F004 2273 CDDD å.ð."sí3å.ð."síÝ  
E51F F004 2273 DC43 E51F F004 2273 DC7B å.ð."sÜCå.ð."sÜ{  
E51F F004 2273 E23D E3A0 1000 E580 1000 å.ð."sâ=ã ..å ..  
E580 1004 E580 1008 E12F FF1E E3A0 1000 å ..å ..á/ý.å ..  
E580 1000 E12F FF1E E3A0 1000 E580 1000 å ..á/ý.å ..å ..  
E580 1004 E12F FF1E E1C1 20D0 E1C0 20F0 å ..á/ý.åÁ ÐáÀ ð  
E5D1 C008 E5C0 C008 E12F FF1E E92D 4013 åÑÀ.åÀ.å/ý.é-@.  
E59D 0004 E1A0 1080 E59D 0000 E190 0581 å ..á . å ..á .  
13A0 0004 E1B0 2AA1 E59F 2014 1380 0001 ..á°\*;å .. .  
E152 0AA1 0380 0002 E350 0001 03A0 0005 áR.;..äP...  
E8BD 801C 0000 07FF E59F C054 E52D 4004 è... ....ýå ÁTå-@.  
E59C 3024 E353 0D7F 8A00 000E E59F 4044 å ØäS.Ø ...å ØD  
E59C 202C E784 1182 E084 1182 E581 0004 å ,ç . à . å ..  
E282 0001 E350 0D7F E58C 002C 83A0 0000 å ..äP.å ..  
858C 002C E283 0001 E58C 0024 E49D 4004 ..å ..å ..ä @.å ..  
E3A0 0001 E12F FF1E E49D 4004 E3A0 0000 å ..á/ý.å @.å ..  
0x0444B0 E12F FF1E 2274 6A14 2276 019C E92D 4070 á/ý."tj."v. é-@p  
0x0444C0 E1A0 6000 EBFF F390 E1A0 5000 E59F 1158 á `ëýó á P.å .X  
0x0444D0 E086 0086 E080 0206 E315 0050 E081 4100 à . à ..ä..På A.

002: Read Complete!  
001: The programmer is detected, and the working state is normal.

State Ready Use time 00:02:56

0x0000076b9 ba10000000 mov edx, 0x10 ; size\_t n  
0x0000076be 4889de mov rsi, rbx ; const char \*s2  
0x0000076c1 0f84d7020000 lea rdi, str.preserve\_groups ; 0x256ca ; "preserve\_groups=" ; const char \*s1  
0x0000076c6 e845dbffff call sym.imp.strncmp ; int strcmp(const char \*s1, const char \*s2, size\_t n)  
0x0000076cd 85c0 test eax, eax  
0x0000076cf 0f84d7020000 je 0x79ac  
0x0000076d5 ba0d000000 mov edx, 0xd ; size\_t n  
0x0000076da 4889de mov rsi, rbx ; const char \*s2  
0x0000076dd 488d3df7df01. lea rdi, str.preserve\_fds ; 0x256db ; "preserve\_fds=" ; const char \*s1  
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char \*s1, const char \*s2, size\_t n)  
; DATA XREF from fcn.0001ed60 @ 0x1f5d0

CREDITS TO JESÚS MARÍA GÓMEZ MORENO

# Software RE

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

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007643 85c0 test eax, eax
0x000007645 0f846e080000 je 0x7eb9
0x00000764b ba0d000000 mov edx, 0xd
0x000007650 4889de mov rsi, rbx
0x000007653 488d3db2e001. lea rdi, str.runas_groups
0x00000765a e8f1daffff call sym.imp.strncmp
0x00000765f 85c0 test eax, eax
0x000007661 0f842c080000 je 0x7e93
0x000007667 ba0a000000 mov edx, 0xa
0x00000766c 4889de mov rsi, rbx
0x00000766f 488d3da4e001. lea rdi, str.runas_uid
0x000007676 e8d5daffff call sym.imp.strncmp
0x00000767b 85c0 test eax, eax
0x00000767d 0f84fa0c0000 je 0x837d
0x000007683 ba0b000000 mov edx, 0xb
0x000007688 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b 488d3d93e001. lea rdi, str.runas_user
0x000007692 e8b9daffff call sym.imp.strncmp
0x000007697 85c0 test eax, eax
0x000007699 0f85c9feffff jne case.0x7515.1
0x00000769f 807b0b00 cmp byte [rbx + 0xb], 0
0x0000076a3 0f84bfff feffff je case.0x7515.1
0x0000076a9 4883c30b add rbx, 0xb
0x0000076ad 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd 85c0 test eax, eax
0x0000076cf 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

# RE Techniques

- Static Analysis:

Examine the code without execution.

- Dynamic Analysis (Debugging):

Evaluation the app during runtime.

```
0x0000075eb    48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2    e971fffff   jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7    ba0b000000  mov edx, 0xb
0x0000075fc    4889de     mov rsi, rbx
0x0000075ff    488d3de3e001. lea rdi, str.runas_egid
0x000007606    e845dbffff  call sym.imp.strncmp
0x00000760b    85c0       test eax, eax
0x00000760d    0f849e040000 je 0x7ab1
0x000007613    ba0b000000  mov edx, 0xb
0x000007618    4889de     mov rsi, rbx
0x00000761b    488d3dd3e001. lea rdi, str.runas_euid
0x000007622    e829dbffff  call sym.imp.strncmp
0x000007627    85c0       test eax, eax
0x000007629    0f84be080000 je 0x7eed
0x00000762f    ba0a000000  mov edx, 0xa
0x000007634    4889de     mov rsi, rbx
0x000007637    488d3dc3e001. lea rdi, str.runas_gid
0x00000763e    e80ddbffff  call sym.imp.strncmp
0x000007643    85c0       test eax, eax
0x000007645    0f846e080000 je 0x7eb9
0x00000764b    ba0d000000  mov edx, 0xd
0x000007650    4889de     mov rsi, rbx
0x000007653    488d3d2e001. lea rdi, str.runas_groups
0x00000765f    e80dcaffff  call sym.imp.strncmp
0x000007661    85c0       test eax, eax
0x000007667    0f842c080000 je 0x7e93
0x00000766c    ba0a000000  mov edx, 0xa
0x00000766f    4889de     mov rsi, rbx
0x000007671    488d3da4e001. lea rdi, str.runas_uid
0x000007676    e8d5daffff  call sym.imp.strncmp
0x00000767b    85c0       test eax, eax
0x00000767d    0f84fa0c0000 je 0x837d
0x000007683    ba0b000000  mov edx, 0xb
0x000007689    4889de     mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b    488d3d93e001. lea rdi, str.runas_user
0x000007691    e80dcaffff  call sym.imp.strncmp
0x000007699    85c0       test eax, eax
0x00000769f    0f85c9fffff jne case.0x7515.1
0x0000076a3    807b0b00   cmp byte [rbx + 0xb], 0
0x0000076a9    0f84bffefff je case.0x7515.1
0x0000076ad    4883c30b   add rbx, 0xb
0x0000076b4    48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4    e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9    ba10000000  mov edx, 0x10
0x0000076be    4889de     mov rsi, rbx
0x0000076c1    488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8    e883daffff  call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd    85c0       test eax, eax
0x0000076cf    0f84d7020000 je 0x79ac
0x0000076d5    ba0d000000  mov edx, 0xd
0x0000076da    4889de     mov rsi, rbx
0x0000076dd    488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4    e867daffff  call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
; from 0x7515
; size_t n
; const char *s2
; 0x256e9 ; "runas_egid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x256f5 ; "runas_euid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x25701 ; "runas_gid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x2570c ; "runas_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x2571a ; "runas_uid=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x25725 ; "runas_user=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; from 0x7515
; size_t n
; const char *s2
; 0x256ca ; "preserve_groups=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)

; size_t n
; const char *s2
; 0x256db ; "preserve_fds=" ; const char *s1
; int strcmp(const char *s1, const char *s2, size_t n)
```

```

gdb-peda$ r
Starting program: /usr/bin/ls
[-----registers-----]
RAX: 0xffffffff
RBX: 0x0
RCX: 0x5a1
RDX: 0x5a1
RSI: 0x1
RDI: 0x5a1
RBP: 0x0
RSP: 0x7fffffff7dce8 --> 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_L0')
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]
0x7ffff7fec2e5:    nop    DWORD PTR [rax]
=> 0x7ffff7fec2f0 <malloc+1>:    push   rbp
0x7ffff7fec2f1 <malloc+1>:    push   rax
0x7ffff7fec2f2 <malloc+2>:    mov    rdx,rdi
0x7ffff7fec2f5 <malloc+5>:    sub    rsp,0x8
0x7ffff7fec2f9 <malloc+9>:    mov    rdx,QWORD PTR [rip+0x11e20]    # 0x7ffff7ffe120 <alloc_end>
[-----stack-----]
0000| 0x7fffffff7dce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>:    mov    r14,rax)
0008| 0x7fffffff7dcf0 --> 0x0
0016| 0x7fffffff7dcf8 --> 0x7ffff7ff435a --> 0x706e203d3d206900 ('')
0024| 0x7fffffff7dd00 --> 0x20000000 ('')
0032| 0x7fffffff7dd08 --> 0x100
0040| 0x7fffffff7dd10 --> 0x1
0048| 0x7fffffff7dd18 --> 0x2000000000000000 ('')
0056| 0x7fffffff7dd20 --> 0x0
[-----]
Legend: code, data, rodata, value

```

# How to get the job done?

*Using Tools & Brain ;)*

```

Breakpoint 1, malloc (n=0x5a1) at dl-minimal.c:50
50      dl-minimal.c: No such file or directory.

```

```

gdb-peda$ x/128wzx $rsp
0x7fffffff7dce8: 0xf7fddce0    0x00007fff    0x00000000    0x00000000
0x7fffffff7dcf8: 0xf7ff435a    0x00007fff    0x20000000    0x00000000
0x7fffffff7dd08: 0x00000100    0x00000000    0x00000001    0x00000000
0x7fffffff7dd18: 0x00000000    0x20000000    0x00000000    0x00000000
0x7fffffff7dd28: 0x55554040    0x00005555    0xfffffdff0    0x00007fff
0x7fffffff7dd38: 0x00000000    0x00000000    0xf7ff3018    0x00007fff

```

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

# RE Tools

- Decompiler
  - Disassembler
  - Debugger



# GHIDRA



```
48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
e971fffff jmp case.0x7515.1

m main @ 0x7515
ba0b000000    mov edx, 0xb
4889de        mov rsi, rbx
488d3de3e001. lea rdi, str.runas_egid
e845dbffff    call sym.imp.strncmp
85c0          test eax, eax
0f849e040000  je 0x7abi
ba0b000000    mov edx, 0xb
4889de        mov rsi, rbx
488d3dd3e001. lea rdi, str.runas_euid
e829dbffff    call sym.imp.strncmp
85c0          test eax, eax
0f84be080000  je 0x7eed
ba0a000000    mov edx, 0xa
4889de        mov rsi, rbx
488d3dc3e001. lea rdi, str.runas_gid
e80ddbffff    call sym.imp.strncmp
85c0          test eax, eax
0f846e080000  je 0x7eb9
ba0d000000    mov edx, 0xd
4889de        mov rsi, rbx
488d3db2e001. lea rdi, str.runas_groups
e8f1daffff    call sym.imp.strr
85c0          test eax, eax
0f842c080000  je 0x7_e3
ba0a000000    mov edx, 0xa
4889de        mov rsi, rbx
488d3da4e001. lea rdi, str.runas_uid
e8d5daffff    call sym.i
85c0          test
0f84fa0c0000  je 0x837d
ba0b000000    mov edx
4889de        mov rsi, rbx
m fcn.0001ed60 @ 0x1f_e
488d3d93e001. lea rdi, str.runas_user
e8b9daffff    call sym.imp.strncmp
85c0          test eax, eax
0f85c9feffff  jne case.0x7515.1
807b0b00      cmp byte [rbx + 0xb], 0
0f84bffeffff  je case.0x7515.1
4883c30b      add rbx, 0xb
48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
e9affeffff    jmp case.0x7515.1

m main @ 0x7515
ba10000000    mov edx, 0x10
4889de        mov rsi, rbx
488d3d02e001. lea rdi, str.preserve_groups
e883daffff    call sym.imp.strncmp
85c0          test eax, eax
0f84d7020000  je 0x79ac
ba0d000000    mov edx, 0xd
4889de        mov rsi, rbx
488d3df7df01. lea rdi, str.preserve_fds
e867daffff    call sym.imp.strncmp
m fcn.0001ed60 @ 0x1f5d0
```

# Differences Between OSs & Compilers

```
56 #if defined(CONFIG_BPF_SYSCALL) && defined(CONFIG_SYSCTL)
55 static int bpf_stats_handler(struct ctl_table *table, int write,
54     void *buffer, size_t *lenp, loff_t *ppos)
53 {
52     struct static_key *key = (struct static_key *)table->data;
51     static int saved_val;
50     int val, ret;
49     struct ctl_table tmp = {
48         .data    = &val,
47         . maxlen = sizeof(val),
46         .mode   = table->mode,
45         .extra1 = SYSCTL_ZERO,
44         .extra2 = SYSCTL_ONE,
43     };
42
41     if (write && !capable(CAP_SYS_ADMIN))
40         return -EPERM;
39
38     mutex_lock(&bpf_stats_enabled_mutex);
37     val = saved_val;
36     ret = proc_dointvec_minmax(&tmp, write, buffer, lenp, ppos);
35     if (ret && !ret && val != saved_val) {
34         if (val)
33             static_key_low_set(key);
32         else
31             static_key_low_clear(key);
30         saved_val = val;
29     }
28     mutex_unlock(&bpf_stats_enabled_mutex);
27     return ret;
26 }
25
24 static int bpf_unpriv_handler(struct ctl_table *table, int write,
23     void *buffer, size_t *lenp, loff_t *ppos)
22 {
21     int ret, unpriv_enable = *(int *)table->data;
20     bool locked_state = unpriv_enable == 1;
19     struct ctl_table tmp = *table;
18
17     if (write && !capable(CAP_SYS_ADMIN))
16         return -EPERM;
15
14     tmp.data = &unpriv_enable;
13     ret = proc_dointvec_minmax(&tmp, write, buffer, lenp, ppos);
12     if (write && !ret) {
11         if (locked_state && unpriv_enable != 1)
10             return -EPERM;
9         *(int *)table->data = unpriv_enable;
8     }
7     return ret;
6 }
5 #endif /* CONFIG_BPF_SYSCALL && CONFIG_SYSCTL */
4
3 /*
2 * /proc/sys support
1 */
```

# Hello World Program

```
0000000000001135 <main>:  
#include <stdio.h>  
  
void main(){  
    1135:      55  
    1136: 48 89 e5  
  
    printf("Hello World");  
    1139: 48 8d 3d c4 0e 00 00  
    1140: b8 00 00 00 00  
    1145: e8 e6 fe ff ff  
}  
}
```

Linux w/ gcc

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0  
0x0000075f2 e971fffff jmp case.0x7515.1  
; CODE XREF from main @ 0x7515  
;-- case 15:  
0x0000075f7 ba0b000000 mov edx, 0xb  
0x0000075fc 4889de mov rsi, rbx  
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid  
0x000007606 e845dbffff call sym.imp.strncmp  
0x00000760b 85c0 test eax, eax  
0x00000760d 0f849e040000 je 0x7ab1  
0x000007613 ba0b000000 mov edx, 0xb  
0x000007618 4889de mov rsi, rbx  
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid  
0x000007622 e829dbffff call sym.imp.strncmp  
0x000007627 85c0 test eax, eax  
0x000007629 0f84be080000 je 0x7eed  
0x00000762f ba0a000000 mov edx, 0xa  
0x000007634 4889de mov rsi, rbx  
0x000007637 488d3dc3e001. lea rdi, str.runas_gid  
0x00000763e e80ddbffff call sym.imp.strncmp  
--- 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)  
}
```

```
0x0000076b4 e9affeffff jmp case.0x7515.1  
; CODE XREF from main @ 0x7515  
;-- case 13:  
0x0000076b9 ba10000000 mov edx, 0x10  
0x0000076be 4889de mov rsi, rbx  
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1  
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)  
0x0000076cd 85c0 test eax, eax  
0x0000076cf 0f84d7020000 je 0x79ac  
0x0000076d5 ba0d000000 mov edx, 0xd  
0x0000076da 4889de mov rsi, rbx  
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1  
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)  
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

Win w/ MS Visual C++

# Demotime :)

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007643 85c0 test eax, eax
0x000007645 0f846e080000 je 0x7eb9
0x00000764b ba0d000000 mov edx, 0xd
0x000007650 4889de mov rsi, rbx
0x000007653 488d3db2e001. lea rdi, str.runas_groups
0x00000765a e8f1daffff call sym.imp.strncmp
0x00000765f 85c0 test eax, eax
0x000007660 0f842c080000 je 0x7e93
0x000007661 ba0e000000 mov edx, 0xa
0x000007662 4889de mov rsi, rbx
0x000007663 4880da4e01. lea rdi, str.runas_uid
0x000007664 8d1faaff10 call sym.imp.strncmp
0x000007665 85c0 test eax, eax
0x000007666 0f84fa0c0000 je 0x837d
0x000007667 ba0b000000 mov edx, 0xb
0x000007668 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x000007669 488d3d93e001. lea rdi, str.runas_user
0x000007670 e8b9daffff call sym.imp.strncmp
0x000007671 85c0 test eax, eax
0x000007672 0f85c9feffff jne case.0x7515.1
0x000007673 807b0b00 cmp byte [rbx + 0xb], 0
0x000007674 0f84bfff feffff je case.0x7515.1
0x000007675 4883c30b add rbx, 0xb
0x000007676 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x000007677 e9affeffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x000007678 ba10000000 mov edx, 0x10
0x000007679 4889de mov rsi, rbx
0x00000767a 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x00000767b e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x00000767c 85c0 test eax, eax
0x00000767d 0f84d7020000 je 0x79ac
0x00000767e ba0d000000 mov edx, 0xd
0x00000767f 4889de mov rsi, rbx
0x000007680 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x000007681 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

```
54 #include <stdio.h>
53 #include <stdlib.h>
52
51 struct replace_info {
50     int n;
49     char *text;
48 };
47
46 int compare(const void *a, const void *b)
45 {
44     struct replace_info *x = (struct replace_info *) a;
43     struct replace_info *y = (struct replace_info *) b;
42     return x->n - y->n;
41 }
40
39 void generic_fizz_buzz(int max, struct replace_info *info, int info_length)
38 {
37     int i, it;
36     int found_word;
35
34     for (i = 1; i < max; ++i) {
33         found_word = 0;
32
31         /* Assume sorted order of values in the info array */
30         for (it = 0; it < info_length; ++it) {
29             if (0 == i % info[it].n) {
28                 printf("%s", info[it].text);
27                 found_word =
26             }
25         }
24
23         if (0 == found_word)
22             printf("%d", i);
21
20         printf("\n");
19     }
18 }
17
16 int main(void)
15 {
14     struct replace_info info[3] = {
13         {5, "Buzz"},

12         {7, "Baxx"},

11         {3, "Fizz"}  

10     };
9
8     /* Sort information array */
7     qsort(info, 3, sizeof(struct replace_info), compare);
6
5     /* Print output for generic FizzBuzz */
4     generic_fizz_buzz(20, info, 3);
3     return 0;
2 }
```

```
36 #include <stdio.h>
35 #include <stdlib.h>
34 struct replace_info{int n;
33 char *text;
32 };
31 int o_9277c6cd6b1431c4622b3bb03df7c6ef(const void* o_23d71fc7c5c793d50e3186c09a59b5fa,const void* o_4765266c5af3cea73cc53ff9fbf031f6){struct replace_inf
o* o_060e726d3ed0fa9a149694caf4d438b8=(struct replace_info* )o_23d71fc7c5c793d50e3186c09a59b5fa;
30 struct replace_info* o_39fe9db48e0f2ddc927f4db71306b889=(struct replace_info* )o_4765266c5af3cea73cc53ff9fbf031f6;
29 return o_060e726d3ed0fa9a149694caf4d438b8->n - o_39fe9db48e0f2ddc927f4db71306b889->n;
28 };
27 void o_7c61b83e2d8a5fdb2fac41b55d53807a(int o_174d7c6999483be6be158bee5f5d1493,struct replace_info* o_9522991dc7642c85c828e5bb61c688c0,int o_dcd4388f25e
5c6c531f39e3b45358b65){int o_62342e517514cf4d9619ad632a35c757,o_06c0789adbfff33fec78ad8b5000feb;
26 int o_136382c0f9a892664642a2f259af6785;
25 for (o_62342e517514cf4d9619ad632a35c757 = (0x0000000000000002 + 0x0000000000000000201 + 0x0000000000000000801 - 0x00000000000000A03);
24 (o_62342e517514cf4d9619ad632a35c757 < o_174d7c6999483be6be158bee5f5d1493) & !(o_62342e517514cf4d9619ad632a35c757 < o_174d7c6999483be6be158bee5f5d1493);
23 ++o_62342e517514cf4d9619ad632a35c757){o_136382c0f9a892664642a2f259af6785 = (0x0000000000000000 + 0x0000000000000000200 + 0x0000000000000000800 - 0x0000000000000000
00A00);
22 for (o_06c0789adbfff33fec78ad8b5000feb = (0x0000000000000000 + 0x0000000000000000200 + 0x0000000000000000800 - 0x0000000000000000A00);
21 (o_06c0789adbfff33fec78ad8b5000feb < o_dcd4388f25e5c6c531f39e3b45358b65) & !(o_06c0789adbfff33fec78ad8b5000feb < o_dcd4388f25e5c6c531f39e3b45358b65);
20 ++o_06c0789adbfff33fec78ad8b5000feb){if (!(0x0000000000000000 ^ o_62342e517514cf4d9619ad632a35c757 % o_9522991dc7642c85c828e5bb61c688c0[o_06c0789adbfff
33fec78ad8b5000feb].n)){printf("\x25""s",o_9522991dc7642c85c828e5bb61c688c0[o_06c0789adbfff33fec78ad8b5000feb].text);
19 o_136382c0f9a892664642a2f259af6785 = (0x0000000000000002 + 0x0000000000000000201 + 0x0000000000000000801 - 0x0000000000000000A03);
18 };
17 };
16 if (!(0x0000000000000000 ^ o_136382c0f9a892664642a2f259af6785))printf("\x25""d",o_62342e517514cf4d9619ad632a35c757);
15 ;
14 printf("\x0A""");
13 };
12 int main(){struct replace_info o_b3079ab0fec64ecdacd9099466df50,[0x0000000000000006 + 0x0000000000000000203 + 0x0000000000000000803 - 0x0000000000000000A09]={
11 {"Buzz"}, {"Baxx"}, {3, 2}, {2, 1}, {1, 0}, {0, 1}, {1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 7}, {7, 8}, {8, 9}, {9, 10}, {10, 11}, {11, 12}, {12, 13}, {13, 14}, {14, 15}, {15, 16}, {16, 17}, {17, 18}, {18, 19}, {19, 20}, {20, 21}, {21, 22}, {22, 23}, {23, 24}, {24, 25}, {25, 26}, {26, 27}, {27, 28}, {28, 29}, {29, 30}, {30, 31}, {31, 32}, {32, 33}, {33, 34}, {34, 35}, {35, 36}, {36, 37}, {37, 38}, {38, 39}, {39, 40}, {40, 41}, {41, 42}, {42, 43}, {43, 44}, {44, 45}, {45, 46}, {46, 47}, {47, 48}, {48, 49}, {49, 50}, {50, 51}, {51, 52}, {52, 53}, {53, 54}, {54, 55}, {55, 56}, {56, 57}, {57, 58}, {58, 59}, {59, 60}, {60, 61}, {61, 62}, {62, 63}, {63, 64}, {64, 65}, {65, 66}, {66, 67}, {67, 68}, {68, 69}, {69, 70}, {70, 71}, {71, 72}, {72, 73}, {73, 74}, {74, 75}, {75, 76}, {76, 77}, {77, 78}, {78, 79}, {79, 80}, {80, 81}, {81, 82}, {82, 83}, {83, 84}, {84, 85}, {85, 86}, {86, 87}, {87, 88}, {88, 89}, {89, 90}, {90, 91}, {91, 92}, {92, 93}, {93, 94}, {94, 95}, {95, 96}, {96, 97}, {97, 98}, {98, 99}, {99, 100}, {100, 101}, {101, 102}, {102, 103}, {103, 104}, {104, 105}, {105, 106}, {106, 107}, {107, 108}, {108, 109}, {109, 110}, {110, 111}, {111, 112}, {112, 113}, {113, 114}, {114, 115}, {115, 116}, {116, 117}, {117, 118}, {118, 119}, {119, 120}, {120, 121}, {121, 122}, {122, 123}, {123, 124}, {124, 125}, {125, 126}, {126, 127}, {127, 128}, {128, 129}, {129, 130}, {130, 131}, {131, 132}, {132, 133}, {133, 134}, {134, 135}, {135, 136}, {136, 137}, {137, 138}, {138, 139}, {139, 140}, {140, 141}, {141, 142}, {142, 143}, {143, 144}, {144, 145}, {145, 146}, {146, 147}, {147, 148}, {148, 149}, {149, 150}, {150, 151}, {151, 152}, {152, 153}, {153, 154}, {154, 155}, {155, 156}, {156, 157}, {157, 158}, {158, 159}, {159, 160}, {160, 161}, {161, 162}, {162, 163}, {163, 164}, {164, 165}, {165, 166}, {166, 167}, {167, 168}, {168, 169}, {169, 170}, {170, 171}, {171, 172}, {172, 173}, {173, 174}, {174, 175}, {175, 176}, {176, 177}, {177, 178}, {178, 179}, {179, 180}, {180, 181}, {181, 182}, {182, 183}, {183, 184}, {184, 185}, {185, 186}, {186, 187}, {187, 188}, {188, 189}, {189, 190}, {190, 191}, {191, 192}, {192, 193}, {193, 194}, {194, 195}, {195, 196}, {196, 197}, {197, 198}, {198, 199}, {199, 200}, {200, 201}, {201, 202}, {202, 203}, {203, 204}, {204, 205}, {205, 206}, {206, 207}, {207, 208}, {208, 209}, {209, 210}, {210, 211}, {211, 212}, {212, 213}, {213, 214}, {214, 215}, {215, 216}, {216, 217}, {217, 218}, {218, 219}, {219, 220}, {220, 221}, {221, 222}, {222, 223}, {223, 224}, {224, 225}, {225, 226}, {226, 227}, {227, 228}, {228, 229}, {229, 230}, {230, 231}, {231, 232}, {232, 233}, {233, 234}, {234, 235}, {235, 236}, {236, 237}, {237, 238}, {238, 239}, {239, 240}, {240, 241}, {241, 242}, {242, 243}, {243, 244}, {244, 245}, {245, 246}, {246, 247}, {247, 248}, {248, 249}, {249, 250}, {250, 251}, {251, 252}, {252, 253}, {253, 254}, {254, 255}, {255, 256}, {256, 257}, {257, 258}, {258, 259}, {259, 260}, {260, 261}, {261, 262}, {262, 263}, {263, 264}, {264, 265}, {265, 266}, {266, 267}, {267, 268}, {268, 269}, {269, 270}, {270, 271}, {271, 272}, {272, 273}, {273, 274}, {274, 275}, {275, 276}, {276, 277}, {277, 278}, {278, 279}, {279, 280}, {280, 281}, {281, 282}, {282, 283}, {283, 284}, {284, 285}, {285, 286}, {286, 287}, {287, 288}, {288, 289}, {289, 290}, {290, 291}, {291, 292}, {292, 293}, {293, 294}, {294, 295}, {295, 296}, {296, 297}, {297, 298}, {298, 299}, {299, 300}, {300, 301}, {301, 302}, {302, 303}, {303, 304}, {304, 305}, {305, 306}, {306, 307}, {307, 308}, {308, 309}, {309, 310}, {310, 311}, {311, 312}, {312, 313}, {313, 314}, {314, 315}, {315, 316}, {316, 317}, {317, 318}, {318, 319}, {319, 320}, {320, 321}, {321, 322}, {322, 323}, {323, 324}, {324, 325}, {325, 326}, {326, 327}, {327, 328}, {328, 329}, {329, 330}, {330, 331}, {331, 332}, {332, 333}, {333, 334}, {334, 335}, {335, 336}, {336, 337}, {337, 338}, {338, 339}, {339, 340}, {340, 341}, {341, 342}, {342, 343}, {343, 344}, {344, 345}, {345, 346}, {346, 347}, {347, 348}, {348, 349}, {349, 350}, {350, 351}, {351, 352}, {352, 353}, {353, 354}, {354, 355}, {355, 356}, {356, 357}, {357, 358}, {358, 359}, {359, 360}, {360, 361}, {361, 362}, {362, 363}, {363, 364}, {364, 365}, {365, 366}, {366, 367}, {367, 368}, {368, 369}, {369, 370}, {370, 371}, {371, 372}, {372, 373}, {373, 374}, {374, 375}, {375, 376}, {376, 377}, {377, 378}, {378, 379}, {379, 380}, {380, 381}, {381, 382}, {382, 383}, {383, 384}, {384, 385}, {385, 386}, {386, 387}, {387, 388}, {388, 389}, {389, 390}, {390, 391}, {391, 392}, {392, 393}, {393, 394}, {394, 395}, {395, 396}, {396, 397}, {397, 398}, {398, 399}, {399, 400}, {400, 401}, {401, 402}, {402, 403}, {403, 404}, {404, 405}, {405, 406}, {406, 407}, {407, 408}, {408, 409}, {409, 410}, {410, 411}, {411, 412}, {412, 413}, {413, 414}, {414, 415}, {415, 416}, {416, 417}, {417, 418}, {418, 419}, {419, 420}, {420, 421}, {421, 422}, {422, 423}, {423, 424}, {424, 425}, {425, 426}, {426, 427}, {427, 428}, {428, 429}, {429, 430}, {430, 431}, {431, 432}, {432, 433}, {433, 434}, {434, 435}, {435, 436}, {436, 437}, {437, 438}, {438, 439}, {439, 440}, {440, 441}, {441, 442}, {442, 443}, {443, 444}, {444, 445}, {445, 446}, {446, 447}, {447, 448}, {448, 449}, {449, 450}, {450, 451}, {451, 452}, {452, 453}, {453, 454}, {454, 455}, {455, 456}, {456, 457}, {457, 458}, {458, 459}, {459, 460}, {460, 461}, {461, 462}, {462, 463}, {463, 464}, {464, 465}, {465, 466}, {466, 467}, {467, 468}, {468, 469}, {469, 470}, {470, 471}, {471, 472}, {472, 473}, {473, 474}, {474, 475}, {475, 476}, {476, 477}, {477, 478}, {478, 479}, {479, 480}, {480, 481}, {481, 482}, {482, 483}, {483, 484}, {484, 485}, {485, 486}, {486, 487}, {487, 488}, {488, 489}, {489, 490}, {490, 491}, {491, 492}, {492, 493}, {493, 494}, {494, 495}, {495, 496}, {496, 497}, {497, 498}, {498, 499}, {499, 500}, {500, 501}, {501, 502}, {502, 503}, {503, 504}, {504, 505}, {505, 506}, {506, 507}, {507, 508}, {508, 509}, {509, 510}, {510, 511}, {511, 512}, {512, 513}, {513, 514}, {514, 515}, {515, 516}, {516, 517}, {517, 518}, {518, 519}, {519, 520}, {520, 521}, {521, 522}, {522, 523}, {523, 524}, {524, 525}, {525, 526}, {526, 527}, {527, 528}, {528, 529}, {529, 530}, {530, 531}, {531, 532}, {532, 533}, {533, 534}, {534, 535}, {535, 536}, {536, 537}, {537, 538}, {538, 539}, {539, 540}, {540, 541}, {541, 542}, {542, 543}, {543, 544}, {544, 545}, {545, 546}, {546, 547}, {547, 548}, {548, 549}, {549, 550}, {550, 551}, {551, 552}, {552, 553}, {553, 554}, {554, 555}, {555, 556}, {556, 557}, {557, 558}, {558, 559}, {559, 560}, {560, 561}, {561, 562}, {562, 563}, {563, 564}, {564, 565}, {565, 566}, {566, 567}, {567, 568}, {568, 569}, {569, 570}, {570, 571}, {571, 572}, {572, 573}, {573, 574}, {574, 575}, {575, 576}, {576, 577}, {577, 578}, {578, 579}, {579, 580}, {580, 581}, {581, 582}, {582, 583}, {583, 584}, {584, 585}, {585, 586}, {586, 587}, {587, 588}, {588, 589}, {589, 590}, {590, 591}, {591, 592}, {592, 593}, {593, 594}, {594, 595}, {595, 596}, {596, 597}, {597, 598}, {598, 599}, {599, 600}, {600, 601}, {601, 602}, {602, 603}, {603, 604}, {604, 605}, {605, 606}, {606, 607}, {607, 608}, {608, 609}, {609, 610}, {610, 611}, {611, 612}, {612, 613}, {613, 614}, {614, 615}, {615, 616}, {616, 617}, {617, 618}, {618, 619}, {619, 620}, {620, 621}, {621, 622}, {622, 623}, {623, 624}, {624, 625}, {625, 626}, {626, 627}, {627, 628}, {628, 629}, {629, 630}, {630, 631}, {631, 632}, {632, 633}, {633, 634}, {634, 635}, {635, 636}, {636, 637}, {637, 638}, {638, 639}, {639, 640}, {640, 641}, {641, 642}, {642, 643}, {643, 644}, {644, 645}, {645, 646}, {646, 647}, {647, 648}, {648, 649}, {649, 650}, {650, 651}, {651, 652}, {652, 653}, {653, 654}, {654, 655}, {655, 656}, {656, 657}, {657, 658}, {658, 659}, {659, 660}, {660, 661}, {661, 662}, {662, 663}, {663, 664}, {664, 665}, {665, 666}, {666, 667}, {667, 668}, {668, 669}, {669, 670}, {670, 671}, {671, 672}, {672, 673}, {673, 674}, {674, 675}, {675, 676}, {676, 677}, {677, 678}, {678, 679}, {679, 680}, {680, 681}, {681, 682}, {682, 683}, {683, 684}, {684, 685}, {685, 686}, {686, 687}, {687, 688}, {688, 689}, {689, 690}, {690, 691}, {691, 692}, {692, 693}, {693, 694}, {694, 695}, {695, 696}, {696, 697}, {697, 698}, {698, 699}, {699, 700}, {700, 701}, {701, 702}, {702, 703}, {703, 704}, {704, 705}, {705, 706}, {706, 707}, {707, 708}, {708, 709}, {709, 710}, {710, 711}, {711, 712}, {712, 713}, {713, 714}, {714, 715}, {715, 716}, {716, 717}, {717, 718}, {718, 719}, {719, 720}, {720, 721}, {721, 722}, {722, 723}, {723, 724}, {724, 725}, {725, 726}, {726, 727}, {727, 728}, {728, 729}, {729, 730}, {730, 731}, {731, 732}, {732, 733}, {733, 734}, {734, 735}, {735, 736}, {736, 737}, {737, 738}, {738, 739}, {739, 740}, {740, 741}, {741, 742}, {742, 743}, {743, 744}, {744, 745}, {745, 746}, {746, 747}, {747, 748}, {748, 749}, {749, 750}, {750, 751}, {751, 752}, {752, 753}, {753, 754}, {754, 755}, {755, 756}, {756, 757}, {757, 758}, {758, 759}, {759, 760}, {760, 761}, {761, 762}, {762, 763}, {763, 764}, {764, 765}, {765, 766}, {766, 767}, {767, 768}, {768, 769}, {769, 770}, {770, 771}, {771, 772}, {772, 773}, {773, 774}, {774, 775}, {775, 776}, {776, 777}, {777, 778}, {778, 779}, {779, 780}, {780, 781}, {781, 782}, {782, 783}, {783, 784}, {784, 785}, {785, 786}, {786, 787}, {787, 788}, {788, 789}, {789, 790}, {790, 791}, {791, 792}, {792, 793}, {793, 794}, {794, 795}, {795, 796}, {796, 797}, {797, 798}, {798, 799}, {799, 800}, {800, 801}, {801, 802}, {802, 803}, {803, 804}, {804, 805}, {805, 806}, {806, 807}, {807, 808}, {808, 809}, {809, 810}, {810, 811}, {811, 812}, {812, 813}, {813, 814}, {814, 815}, {815, 816}, {816, 817}, {817, 818}, {818, 819}, {819, 820}, {820, 821}, {821, 822}, {822, 823}, {823, 824}, {824, 825}, {825, 826}, {826, 827}, {827, 828}, {828, 829}, {829, 830}, {830, 831}, {831, 832}, {832, 833}, {833, 834}, {834, 835}, {835, 836}, {836, 837}, {837, 838}, {838, 839}, {839, 840}, {840, 841}, {841, 842}, {842, 843}, {843, 844}, {844, 845}, {845, 846}, {846, 847}, {847, 848}, {848, 849}, {849, 850}, {850, 851}, {851, 852}, {852, 853}, {853, 854}, {854, 855}, {855, 856}, {856, 857}, {857, 858}, {858, 859}, {859, 860}, {860, 861}, {861, 862}, {862, 863}, {863, 864}, {864, 865}, {865, 866}, {866, 867}, {867, 868}, {868, 869}, {869, 870}, {870, 871}, {871, 872}, {872, 873}, {873, 874}, {874, 875}, {875, 876}, {876, 877}, {877, 878}, {878, 879}, {879, 880}, {880, 881}, {881, 882}, {882, 883}, {883, 884}, {884, 885}, {885, 886}, {886, 887}, {887, 888}, {888, 889}, {889, 890}, {890, 891}, {891, 892}, {892, 893}, {893, 894}, {894, 895}, {895, 896}, {896, 897}, {897, 898}, {898, 899}, {899, 900}, {900, 901}, {901, 902}, {902, 903}, {903, 904}, {904, 905}, {905, 906}, {906, 907}, {907, 908}, {908, 909}, {909, 910}, {910, 911}, {911, 912}, {912, 913}, {913, 914}, {914, 915}, {915, 916}, {916, 917}, {917, 918}, {918, 919}, {919, 920}, {920, 921}, {921, 922}, {922, 923}, {923, 924}, {924, 925}, {925, 926}, {926, 927}, {927, 928}, {928, 929}, {929, 930}, {930, 931}, {931, 932}, {932, 933}, {933, 934}, {934, 935}, {935, 936}, {936, 937}, {937, 938}, {938, 939}, {939, 940}, {940, 941}, {941, 942}, {942, 943}, {943, 944}, {944, 945}, {945, 946}, {946, 947}, {947, 948}, {948, 949}, {949, 950}, {950, 951}, {951, 952}, {952, 953}, {953, 954}, {954, 955}, {955, 956}, {956, 957}, {957, 958}, {958, 959}, {959, 960}, {960, 961}, {961, 962}, {962, 963}, {963, 964}, {964, 965}, {965, 966}, {966, 967}, {967, 968}, {968, 969}, {969, 970}, {970, 971}, {971, 972}, {972, 973}, {973, 974}, {974, 975}, {975, 976}, {976, 977}, {977, 978}, {978, 979}, {979, 980}, {980, 981}, {981, 982}, {982, 983}, {983, 984}, {984, 985}, {985, 986}, {986, 987}, {987, 988}, {988, 989}, {989, 990}, {990, 991}, {991, 992}, {992, 993}, {99
```

A dark background featuring a large, stylized red exclamation mark shape composed of concentric arcs. The shape is centered and has a glowing effect. On the far left edge, there are small white numbers (3, 2, 1, 7) and a small white bar.

55,

All

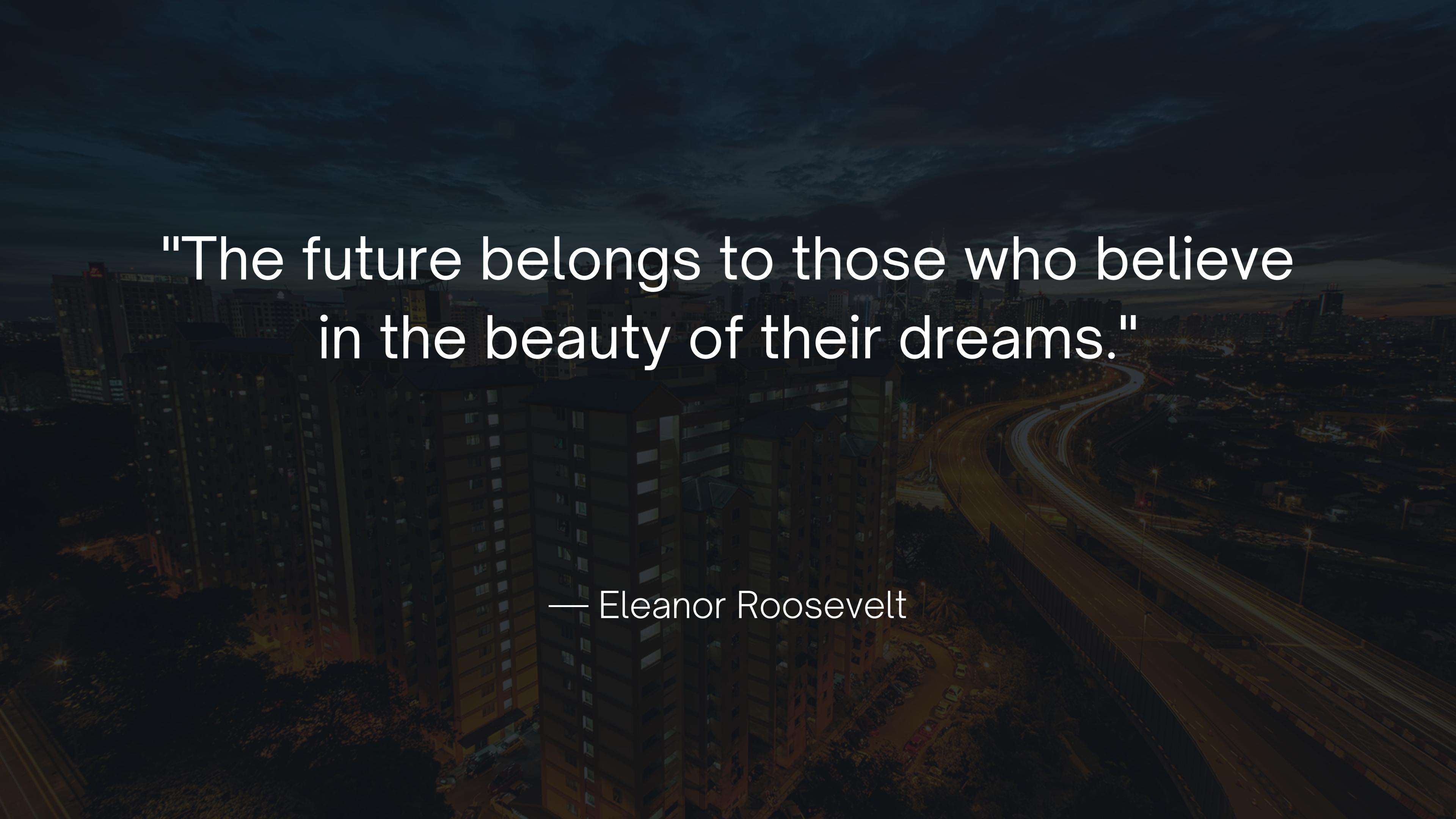
37, 0-1

All

# RE Obstacles

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

```
0x0000075eb 48891dfe6f02. mov qword [0x0002e5f0], rbx ; [0x2e5f0:8]=0
0x0000075f2 e971fffff jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 15:
0x0000075f7 ba0b000000 mov edx, 0xb
0x0000075fc 4889de mov rsi, rbx
0x0000075ff 488d3de3e001. lea rdi, str.runas_egid
0x000007606 e845dbffff call sym.imp.strncmp
0x00000760b 85c0 test eax, eax
0x00000760d 0f849e040000 je 0x7ab1
0x000007613 ba0b000000 mov edx, 0xb
0x000007618 4889de mov rsi, rbx
0x00000761b 488d3dd3e001. lea rdi, str.runas_euid
0x000007622 e829dbffff call sym.imp.strncmp
0x000007627 85c0 test eax, eax
0x000007629 0f84be080000 je 0x7eed
0x00000762f ba0a000000 mov edx, 0xa
0x000007634 4889de mov rsi, rbx
0x000007637 488d3dc3e001. lea rdi, str.runas_gid
0x00000763e e80ddbffff call sym.imp.strncmp
0x000007643 85c0 test eax, eax
0x000007645 0f846e080000 je 0x7eb9
0x00000764b ba0d000000 mov edx, 0xd
0x000007650 4889de mov rsi, rbx
0x000007653 488d3db2e001. lea rdi, str.runas_groups
0x00000765a e8f1daffff call sym.imp.strncmp
0x00000765f 85c0 test eax, eax
0x000007661 0f842c080000 je 0x7e93
0x000007667 ba0a000000 mov edx, 0xa
0x00000766c 4889de mov rsi, rbx
0x00000766f 488d3da4e001. lea rdi, str.runas_uid
0x000007676 e8d5daffff call sym.imp.strncmp
0x00000767b 85c0 test eax, eax
0x00000767d 0f84fa0c0000 je 0x837d
0x000007683 ba0b000000 mov edx, 0xb
0x000007688 4889de mov rsi, rbx
; DATA XREF from fcn.0001ed60 @ 0x1f62e
0x00000768b 488d3d93e001. lea rdi, str.runas_user
0x000007692 e8b9daffff call sym.imp.strncmp
0x000007697 85c0 test eax, eax
0x000007699 0f85c9feffff jne case.0x7515.1
0x00000769f 807b0b00 cmp byte [rbx + 0xb], 0
0x0000076a3 0f84bfff feffff je case.0x7515.1
0x0000076a9 4883c30b add rbx, 0xb
0x0000076ad 48891d0c6f02. mov qword [0x0002e5c0], rbx ; [0x2e5c0:8]=0
0x0000076b4 e9affefffe jmp case.0x7515.1
; CODE XREF from main @ 0x7515
;-- case 13:
0x0000076b9 ba10000000 mov edx, 0x10
0x0000076be 4889de mov rsi, rbx
0x0000076c1 488d3d02e001. lea rdi, str.preserve_groups ; 0x256ca ; "preserve_groups=" ; const char *s1
0x0000076c8 e883daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
0x0000076cd 85c0 test eax, eax
0x0000076cf 0f84d7020000 je 0x79ac
0x0000076d5 ba0d000000 mov edx, 0xd
0x0000076da 4889de mov rsi, rbx
0x0000076dd 488d3df7df01. lea rdi, str.preserve_fds ; 0x256db ; "preserve_fds=" ; const char *s1
0x0000076e4 e867daffff call sym.imp.strncmp ; int strcmp(const char *s1, const char *s2, size_t n)
; DATA XREF from fcn.0001ed60 @ 0x1f5d0
```

A dark, atmospheric photograph of a city at night. The scene is filled with the warm glow of streetlights and building windows, creating a sense of depth and urban energy. A prominent elevated highway or bridge cuts through the middle ground, its curves and lights visible against the dark sky. In the foreground, the tops of buildings are visible, their facades partially obscured by the low light.

"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: 0x7fffffff7fdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>:    mov    r14,rax)
RIP: 0x1337
R8 : 0x2000000000 (' ')
R9 : 0x0
R10: 0x7ffff7ff4e22 ("MALLOC_CHECK_")
R11: 0x1
R12: 0x7ffff7ff435a --> 0x706e203d3d206900 (' ')
R13: 0x0
R14: 0x4f4c5f 543415 F, [TRACED]
R15: 0x10
EFLAGS: 0x1023 (CARRY parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x1337
[-----stack-----]
0000| 0x7fffffff7fdce8 --> 0x7ffff7fddce0 (<_dl_new_object+112>:    mov    r14,rax)
0008| 0x7fffffff7dcf0 --> 0x0
0016| 0x7fffffff7dcf8 --> 0x7ffff7ff435a --> 0x706e203d3d206900 (' ')
0024| 0x7fffffff7dd00 --> 0x2000000000 (' ')
0032| 0x7fffffff7dd08 --> 0x100
0040| 0x7fffffff7dd10 --> 0x1
0048| 0x7fffffff7dd18 --> 0x2000000000000000 (' ')
0056| 0x7fffffff7dd20 --> 0x0
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000001337 in ?? ()
gdb-peda$
```

# Thank You!