



Название:	Крестики
Категория:	Реверс-инжиниринг
Уровень:	Средний
Очки:	350
Описание:	Сможешь решить классический крякм на крестиках и формочках? :)
Теги:	C++, WinForms, Побайтовое шифрование
Автор:	ROP

### Прохождение:

Откроем файл в IDA.

IDA - cross\_cr.exe C:\Users\rop\Pictures\cross\_cr.exe

File Edit Jump Search View Debugger Options Windows Help

Functions

Function name

7 WNDCLASSW WndClass; // [rsp+0Eh] [rbp-20h] BYREF

8

9   \*&v6 = 0LL;

10   \*&v7 = 0LL;

11   \*&v8 = LoadIconW(hInstance, 0x65);

12   \*(&v7 + 1) = hInstance;

13   \*(&v8 + 1) = LoadCursorW(0LL, 0x7F00);

14   \*(&v6 + 1) = sub\_140001180;

15   \*&WndClass.cbClsExtra = v7;

16   \*&WndClass.style = v6;

17   \*&WndClass.hbrBackground = 5uLL;

18   \*&WndClass.hIcon = v8;

19   WndClass.lpszClassName = L"MainWndClass";

20   if ( !RegisterClassW(&WndClass) )

21       return -1;

22   memset(&Msg, 0, sizeof(Msg));

23   CreateWindowExW(0, L"MainWndClass", &WindowName, 0x10000000,

24   while ( GetMessageW(&Msg, 0LL, 0, 0) )

25   {

         TranslateMessage(&Msg);

         DispatchMessage(&Msg);

00000452 WinMain:12 (140001052)

Line 17 of 75, /\_scrt\_is\_nonwritable\_in\_current\_image

Graph overview

Output

Using FLIRT signature: Microsoft VisualC v14 64bit runtime

The screenshot shows the IDA Pro interface with the assembly view open. The assembly code is for the WinMain function, specifically the part where it initializes a window class. A red arrow points from the assembly code at line 17 to the corresponding line in the pseudocode view, which is line 17 of 75. The pseudocode shows the initialization of the WndClass structure, including the registration of the window class and the start of the message loop.

Переходим в обработчик событий.

IDA - cross\_cr.exe C:\Users\rop\Pictures\cross\_cr.exe

```

File Edit Jump Search View Debugger Options Windows Help
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions IDA View-A Pseudocode-B Pseudocode-A Hex View-1
Function name
32     0LL,
33     0LL,
34     0LL);
35     break;
36     case 2u:
37         PostQuitMessage(0);
38         return 0LL;
39     case 0x111u:
40         if ( a4 == qword_140005668 && !WORD1(a3) )
41         {
42             sub_1400013A0();
43             return 0LL;
44         }
45         break;
46     default:
47         return DefWindowProcW(hWndParent, a2, a3, a4);
48     }
49     return 0LL;
50 }

```

Line 17 of 75, /\_scrt\_is\_nonwritable\_in\_current\_image

Graph overview

Output

Затем в основной функционал проверки нашего ввода.

Код выглядит так:

```

int sub_1400013A0()
{
    __int64 v0; // rdi
    __int64 v1; // rax
    __int64 v2; // rsi
    WCHAR *v3; // rbx
    int v4; // esi
    int v5; // ebx
    int v6; // ebp
    int result; // eax
    int v8; // edx
    UINT v9; // r9d

```

```

const WCHAR *v10; // r8
const WCHAR *v11; // rdx
WCHAR String1[256]; // [rsp+20h] [rbp-418h] BYREF
WCHAR String[256]; // [rsp+220h] [rbp-218h] BYREF

memset(String, 0, sizeof(String));
GetWindowTextW(hWnd, String, 256);
memset(String1, 0, sizeof(String1));
GetWindowTextW(qword_140005670, String1, 256);
v0 = -1LL;
v1 = -1LL;
do
    ++v1;
    while ( String1[v1] );
    if ( v1 == 27 && !wcsncmp(String1, L"CODEBY\"", 7uLL) && String1[26] == 125 )
    {
        v2 = 11LL;
        v3 = &String1[13];
        while ( *(v3 - 2) == 45 && iswdigit(*(v3 - 1)) && iswigit(*v3) && iswigit(v3[1]) && iswigit(v3[2]) )
        {
            v2 += 5LL;
            v3 += 5;
            if ( v2 > 24 )
            {
                v4 = atoi(&String1[7]);
                v5 = atoi(&String1[12]);
                v6 = atoi(&String1[17]);
                result = atoi(&String1[22]);
                v8 = result;
                while ( String[v0 + 1] == aMasterOfCodeby[v0 + 1] )
                {
                    v0 += 2LL;
                    if ( v0 == 17 )
                    {

```

```

        if ( (((v4 ^ 0xDFAF7) + 22098798) ^ 0x23B97B) ==
24947582 && (((v5 ^ 0x378) + 1361) ^ 0xB84C) == 40468 )
{
    result = (v6 - 9283) ^ 0xA808;
    if ( result == -47487 && (((v8 ^ 0xFD836) - 131
12) ^ 0xBC3F) == 988548 )
{
    v9 = 0;
    v10 = &unk_1400033D0;
    v11 = &unk_1400033E0;
    return MessageBoxW(0LL, v11, v10, v9);
}
}
return result;
}
result = String[v0];
if ( result != aMasterOfCodeby[v0] )
    goto LABEL_17;
}
break;
}
}
}
LABEL_17:
v9 = 16;
v10 = &Caption;
v11 = &Text;
return MessageBoxW(0LL, v11, v10, v9);
}

```

Имя должно быть `MASTER_OF_CODEBY`.

Флаг должен быть обрамлён `CODEBY{}` в таком виде:

`CODEBY{1234-1234-1234-1234}`.

Каждый блок цифр преобразуется в число из 10-ой системе.

Затем над каждым производятся такие действия:

```
part1 ^= 897123;  
part1 ^= 19092;  
part1 -= 1389123;  
part1 += 23487921;  
part1 ^= 2341243;
```

```
part2 ^= 888;  
part2 += 123;  
part2 += 1238;  
part2 ^= 4231;  
part2 ^= 43211;
```

```
part3 -= 9283;  
part3 ^= 5193;  
part3 ^= 48193;  
part3 -= 444;  
part3 += 183;
```

```
part4 ^= 571982;  
part4 ^= 4291;  
part4 ^= 488123;  
part4 -= 13112;  
part4 ^= 48191;
```

А затем идёт сравнение:

```
if (part1 != 24947582 || part2 != 40468 || part3 != 42949  
19548 || part4 != 988548)  
    passwordMatch = false;
```

Можно применить обратные операции к нужным значениям и получить исходные:

```
part1: ( ( ( (24947582 ^ 2341243) - 23487921) + 1389123) ^  
19092) ^ 897123) = 9312
```

```
part2: ( ( ( (40468 ^ 43211) ^ 4231) - 1238) - 123) ^ 888)
= 8831
part3: ( ( ( (4294919548 - 183) + 444) ^ 48193) ^ 5193) + 9
283) - 1 = 4812
part4: ( ( ( (988548 ^ 48191) + 13112) ^ 488123) ^ 4291) ^
571982) = 1221
```