

XCTF Windows_Reverse1

原创

[weixin_44164182](#) 于 2021-02-11 13:37:19 发布 48 收藏

分类专栏: [ctf 网络安全](#) 文章标签: [安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_44164182/article/details/113789821

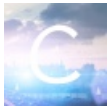
版权



ctf 同时被 2 个专栏收录

8 篇文章 0 订阅

订阅专栏



网络安全

3 篇文章 0 订阅

订阅专栏

```
mov     esi, ds:printf
push   offset Format      ; "please input code:"
call   esi ; printf
lea    edx, [esp+824h+var_404]
push   edx
push   offset aS         ; "%s"
call   ds:scanf
lea    eax, [esp+82Ch+var_404]
push   eax
lea    ecx, [esp+830h+var_804]
call   sub_401000
add    esp, 28h
mov    ecx, offset aDdctfReverseme ; "DDCTF{reverseME}"
lea    eax, [esp+808h+var_804]
```

https://blog.csdn.net/weixin_44164182

主函数中接收输入后, 调用sub_401000后进行判断, 即sub_401000执行了关键逻辑, 根据用户输入生成待判断的字符串。调用时分别使用堆栈和寄存器传入了两个main函数堆栈的地址, 分别是输入字符串地址(堆栈)和执行sub_401000后生成的字符串的保存地址(寄存器ecx)。

```

unsigned int __cdecl sub_401000(const char *raw_input)
{
    char *ptr_output; // ecx
    unsigned int v2; // edi
    unsigned int result; // eax
    const char *v4; // ebx

    v2 = 0;
    result = strlen(raw_input);
    if ( result )
    {
        v4 = (const char *)(raw_input - ptr_output);
        do
        {
            *ptr_output = template_string[ptr_output[(DWORD)v4]];
            ++v2;
            ++ptr_output;
            result = strlen(raw_input);
        }
        while ( v2 < result );
    }
    return result;
}

```

https://blog.csdn.net/weixin_44164182

生成的字符串来自template_string，并以某种方式按照用户输入进行索引生成，其中template_string如下

IDA - windows_reverse1.idb (windows_reverse1.exe) Z:\home\han\pen\xtcf\reverse\windows_reverse1.idb

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol

Functions window

Function name

- sub_401000
- _main
- __security_check_cookie(x)
- __tmainCRTStartup
- start
- __report_gsfailure
- __CxxUnhandledExceptionFilter(_EXCEPT)
- _amsg_exit
- _onexit
- _atexit
- sub_40161A
- _XcptFilter
- __ValidateImageBase
- __FindPESection
- __IsNonwritableInCurrentImage
- __initterm
- __initterm_e
- __SEH_prolog4
- __SEH_epilog4
- __except_handler4
- __setdefaultprecision
- sub_401875
- __security_init_cookie
- __crt_debugger_hook
- terminate(void)
- _unlock
- _dllonexit
- _lock
- __except_handler4_common
- _invoke_watson
- __controlfp_s
- memset

Line 1 of 32

Output window

8
Python>print(0x402ff8-0x403018)
-32
Python

AU: idle Down Disk: 192GB

template_string

```

.rdata:00402FF5 db ? ;
.rdata:00402FF6 db ? ;
.rdata:00402FF7 db ? ;
.rdata:00402FF8 char template_string[] ; DATA XREF: sub_401000+24+r
.rdata:00402FF9 align 10h
.rdata:00402FF9 _rdata ends
.rdata:00402FF9
.data:00403000 ; Section 3. (virtual address 00003000)
.data:00403000 ; Virtual size : 000003E4 ( 996.)
.data:00403000 ; Section size in file : 00000200 ( 512.)
.data:00403000 ; Offset to raw data for section: 00001600
.data:00403000 ; Flags C0000040: Data Readable Writable
.data:00403000 ; Alignment : default
.data:00403000 ;
.data:00403000 ; Segment type: Pure data
.data:00403000 ; Segment permissions: Read/Write
.data:00403000 _data segment para public 'DATA' use32
.data:00403000 assume cs: data
.data:00403000 ;org 403000h
.data:00403000 __security_cookie dd 0BB40E64Eh ; DATA XREF: _main+61r
.data:00403000 ; __security_check_cookie(x)+r ...
.data:00403004 dword_403004 dd 44BF19B1h ; DATA XREF: __report_gsfailure+B0+r
.data:00403004 ; __security_init_cookie+2B1w ...
.data:00403008 db 0FFh
.data:00403009 db 0FFh
.data:0040300A db 0FFh
.data:0040300B db 0FFh
.data:0040300C db 0FFh
.data:0040300D db 0FFh
.data:0040300E db 0FFh
.data:0040300F db 0FFh
.data:00403010 dword_403010 dd 0FFFFFFEh ; DATA XREF: .text:004013E2+r
.data:00403014 dword_403014 dd 1 ; DATA XREF: .text:004013C8+r
.data:00403018 aZyxwvutsrqponmlkjihgfedcba`^}\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>'
.data:00403018 db '<:;9876543210/.-,+*('&#39;','!','0

```

生成字符串的索引为

```
template_string[ptr_output[v4]]
v4=raw_input-ptr_output
```

等价于

```
template_string[ptr_output+v4]
template_string[ptr_output+raw_input-ptr_output]
template_string[raw_input]
```

即以用户输入的字符的ASCII码为索引在template_string中索引生成了字符串，并在主函数中与字符串DDCTF{reverseMe}对比。由于ASCII码中可打印字符在32之后，实际参与字符串生成的模板字符串指针和template_string指针的偏移恰为32。

脚本如下

```
target_string = 'DDCTF{reverseME}'
template = [0x7E, 0x7D, 0x7C, 0x7B, 0x7A, 0x79, 0x78, 0x77, 0x76, 0x75,
            0x74, 0x73, 0x72, 0x71, 0x70, 0x6F, 0x6E, 0x6D, 0x6C, 0x6B,
            0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61,
            0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57,
            0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D,
            0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43,
            0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39,
            0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F,
            0x2E, 0x2D, 0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25,
            0x24, 0x23, 0x22, 0x21, 0x20, 0x00]

flag = ''
for i in target_string:
    flag += chr(template.index(ord(i))+32)
print(flag)
```

1. IDA反编译最容易出错的部分在于参数类型的识别和函数调用返回传参的识别，对于函数本身逻辑识别不容易出错
2. 数组的索引array[ind]相当于*(array+ind)其中array为指针，ind为偏移