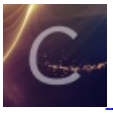
DDCTF 2019 逆向题writeup re1-3

原创

Siphre 于 2019-04-19 11:06:21 发布 614 收藏
分类专栏： write up 逆向 CTF python 文章标签： DDCTF 逆向 CTF write up


write up 同时被 3 个专栏收录
5 篇文章 0 订阅
订阅专栏


逆向
12 篇文章 0 订阅
订阅专栏


CTF
7 篇文章 0 订阅
订阅专栏

Re1 windows reverse 1

Upx壳，用工具脱壳。静态分析：

主流程很简单，用户输入input经过sub_401000函数处理后与DDCTF{reverseME}进行比较。

Sub_401000:

看汇编，函数逻辑就是，根据用户输入的字符的ascii码，读取byte_402FF8对应下标的字符。

byte_402FF8是一个倒序的Ascii码可见字符表。00403018-00402FF8=32，32就是可见字符表的第一个字符的ascii码。

```
.data:00403018     aZyxwvutsrqponm db '~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>'
.data:00403018                     db '=<;:9876543210/.-,+*)(',27h,'&%$#"! ',0
```

写出脚本:

```
a=[]
flag="DDCTF{reverseME}"
for i in range(32,127):
    a.append(chr(i))
a.reverse()

for i in range(len(flag)):
    for j in range(len(a)):
        if a[j]==flag[i]:
            print(chr(j+32),end="")
```

脱壳以后无法动调、无法运行，怀疑是需要修复什么的，还不太会，所以只用了静态分析。

Re2 windows reverse 2

Aspack壳，XVolk通用脱壳机可以完美脱，可调试、可运行。

IDA+F5静态分析：

sub_4011F0：输入合法性校验，长度为非零偶数，字符范围0-9、A-Z

sub_401240：将输入2位为一组，视为十六进制，转换为十进制数，传给sub_401000。

sub_401000：base64

最后跟reverse+比较

Py脚本：

```
import base64
a="reverse+"
byte=base64.b64decode(a.encode('utf-8'))
print(byte)
for i in range(len(byte)):
    print(str(hex(byte[i])[-2:]).upper(),end="")
```

Re3 confused

IOS逆向，第一次逆IOS，解压confused.app\Contents\MacOS目录下有个xia0Crakeme,IDA打开。

checkCode：主校验函数，限制输入前缀”DDCTF{“，后缀”}”,中间18位

```
if ( objc_msgSend_ptr(v17_input, selRef_hasPrefix_, &cfstr_Ddctf) )// limit the prefix
{
  v8 = objc_msgSend_ptr(v17_input, selRef_length);// check the final char
  v9 = objc_msgSend_ptr(v17_input, selRef_substringFromIndex_, v8 - 1);// get the final char
  LODWORD(v10_tailChar) = objc_retainAutoreleasedReturnValue(v9);
  v16 = v10_tailChar;
  if ( objc_msgSend_ptr(v10_tailChar, selRef_isEqualToString_, &stru_100003138) )// limit the tail
  {
    v11_wholeLength = objc_msgSend_ptr(v17_input, selRef_length);// whole length
    v23 = 6LL;
    v22 = v11_wholeLength - 7;
    v24 = 6LL;
    v25 = v11_wholeLength - 7;
    v12 = objc_msgSend_ptr(v17_input, selRef_substringWithRange_, 6LL, v11_wholeLength - 7);// from 6 sub v11 - 7 char str[6:-7]
    LODWORD(v13) = objc_retainAutoreleasedReturnValue(v12);
    v15 = v13;
    if ( objc_msgSend_ptr(v13, selRef_length) == 18 )// str[6:-7] must has 18 bytes
    {
      LODWORD(v14) = objc_retainAutorelease(v15);
      v18 = objc_msgSend_ptr(v14, selRef_UTF8String);
    }
    objc_storeStrong(&v15, 0LL);
  }
  objc_storeStrong(&v16, 0LL);
}
if ( v18 )
{
  if ( sub_1000011D0(v18) == 1 )              // check str[6:-7]
```


sub_1000011D0：包含sub_100001F60、sub_100001F00

sub_100001F60：一个函数表，通过十六进制数F1-F8标识八个函数，实际只用到F0 F8 F2 F6 F7 F3。

```
int __fastcall sub_100001F60(__int64 a1, __int64 input)
{
  *a1 = 0;
  *(a1 + 4) = 0;                              // 关键是要看懂0 和6
  *(a1 + 8) = 0;
  *(a1 + 12) = 0;
  *(a1 + 16) = 0;
  *(a1 + 176) = 0;                            // return value. 1=success  0=failed
  *(a1 + 32) = 0xF0u;
  *(a1 + 40) = sub_100001D70;                 // *(a1+24) +=6看一下汇编,根据qword_100001E38确定偏移量
  *(a1 + 48) = 0xF1u;
  *(a1 + 56) = xor_100001A60;                 // *(a1+24) + 1
  *(a1 + 64) = 0xF2u;
  *(a1 + 72) = chkinput_100001AA0;            // *(a1+24) + 2
                                              // *(a1+16) 赋值 【check input】
  *(a1 + 80) = 0xF4u;
  *(a1 + 88) = add_100001CB0;                 // *(a1+24) + 1
  *(a1 + 96) = 0xF5u;
  *(a1 + 104) = jian_100001CF0;               // *(a1+24) + 1
  *(a1 + 112) = 0xF3u;                        // F3 is control bit of end?
  *(a1 + 120) = sub_100001B70;                // decompile failed. Need to read the Assembly Language
  *(a1 + 128) = 0xF6u;
  *(a1 + 136) = sub_100001B10;                // *(a1+24) + 2
                                              // when *(a1+16) == 1 置零
                                              // when *(a1+16) != 1 *(a1+24)+=*(*(a1+24)+1)
  *(a1 + 144) = 0xF7u;
  *(a1 + 152) = sub_100001D30;                // 最终赋值
                                              // return result =*(a1 + 176) = *(*(a1 + 24) + 1)
                                              // *(a1+24) + 5
  *(a1 + 160) = 0xF8u;
  *(a1 + 168) = sub_100001C60;                // *(a1+24) + 1  offset the alphbet
  input_100003F58 = malloc(1024uLL);
  return __memcpy_chk(input_100003F58 + 48, input, 18LL, -1LL);
```


F0：10001DC5的jump需要读汇编，作用是F0后两位赋给a1(见后文函数表的绿框内数据)，指针后移6

```
__text:0000000100001DB3                 lea     rax, qword_100001E38
__text:0000000100001DBA                 mov     rcx, [rbp+var_20]
__text:0000000100001DBE                 movsxd  rdx, dword ptr [rax+rcx*4]
__text:0000000100001DC2                 add     rdx, rax
__text:0000000100001DC5                 jmp     rdx
__text:0000000100001DC7 ; ---------------------------------------------------------------------------
__text:0000000100001DC7                 mov     rax, [rbp+var_18] ; -71
__text:0000000100001DCB                 mov     ecx, [rax]      ; *(*(a1+24))
__text:0000000100001DCD                 mov     rax, [rbp+var_8]
__text:0000000100001DD1                 mov     [rax], ecx
__text:0000000100001DD3                 jmp     loc_100001E26
__text:0000000100001DD8 ; ---------------------------------------------------------------------------
__text:0000000100001DD8                 mov     rax, [rbp+var_18] ; -60
__text:0000000100001DDC                 mov     ecx, [rax]
__text:0000000100001DDE                 mov     rax, [rbp+var_8]
__text:0000000100001DE2                 mov     [rax+4], ecx    ; *(a1+4)=*(a1+24)
__text:0000000100001DE5                 jmp     loc_100001E26
__text:0000000100001DEA ; ---------------------------------------------------------------------------
__text:0000000100001DEA                 mov     rax, [rbp+var_18] ; -4E
__text:0000000100001DEE                 mov     ecx, [rax]
__text:0000000100001DF0                 mov     rax, [rbp+var_8]
__text:0000000100001DF4                 mov     [rax+8], ecx    ; *(a1+8)
__text:0000000100001DF7                 jmp     loc_100001E26
__text:0000000100001DFC ; ---------------------------------------------------------------------------
__text:0000000100001DFC                 mov     rax, [rbp+var_18] ; -3C
__text:0000000100001E00                 mov     ecx, [rax]
__text:0000000100001E02                 mov     rax, [rbp+var_8] ; first three row are same
__text:0000000100001E06                 mov     [rax+0Ch], ecx  ; *(a1+12)
__text:0000000100001E09                 jmp     loc_100001E26
__text:0000000100001E0E ; ---------------------------------------------------------------------------
__text:0000000100001E0E                 mov     rax, cs:input_100003F58 ; -2A
__text:0000000100001E15                 mov     rcx, [rbp+var_18] ; *(a1+24)+2
__text:0000000100001E19                 movsxd  rcx, dword ptr [rcx] ; *(*(a1+24)+2)
__text:0000000100001E1C                 movsx   edx, byte ptr [rax+rcx] ; *(*(a1+24)+input)
__text:0000000100001E20                 mov     rax, [rbp+var_8]
```

F8：字母向后偏移2位，其他字符不变，指针后移1

F2：校验输入，标志位*(a1+16)置1，指针后移1

F6：输入校验成功，标志位*(a1+16)置0，无事发生；输入校验失败，跳转到F7 00处，打印”失败”字样。指针后移2

F7：将后一位的值传给*(a1+176)并return,成功就是1，失败就是0，指针后移5

F3：终止符

sub_100001F00：逐位读取loc_100001984的函数表，遇到F3之前，循环执行sub_100001E50

sub_100001E50：根据当前取值字符F0-F8，执行相应函数，作用类似switch case语句。

loc_100001984的函数表：（绿框表示字符校验值，黑框表示终止符，红框表示成功失败位）

```
F0 10 66 00 00 00 F8 F2  30 F6 C1 F0 10 "63 00 00
00 F8 F2 31 F6 |B6 F0 10  6A 00 00 00 F8 F2 32 F6 |
AB F0 10 6A 00 00 00 F8  F2 33 F6 |A0 F0 10 6D 00
00 00 F8 F2 34 F6 |95 F0  10 57 00 00 00 F8 F2 35
F6 |8A F0 10 6D 00 00 00  F8 F2 36 F6| 7F F0 10 73
00 00 00 F8 F2 37 F6 |74  F0 10 45 00 00 00 F8 F2
38 F6| 69 F0 10 6D 00 00  00 F8 F2 39 F6 |5E F0 10
72 00 00 00 F8 F2 3A F6 | 53 F0 10 52 00 00 00 F8
F2 3B F6| 48 F0 10 66 00  00 00 F8 F2 3C F6 |3D F0
10 63 00 00 00 F8 F2 3D  F6| 32 F0 10 44 00 00 00
F8 F2 3E F6 |27 F0 10 6A  00 00 00 F8 F2 3F F6 |1C
F0 10 79 00 00 00 F8 F2  40 F6 |11 F0 10 65 00 00
00 F8 F2 41 F6 |06 F7 √01√  00 00 00 F3 F7 00 00 00
00 F3 5D C3 0F 1F 84 00  00 00 00 00 55 48 89 E5

66 63 6A  6A 6D 57 6D 73 45 6D 72 52 66 63 44 6A 79 65
```

1-成功
0-失败

执行顺序是：(F0 -> F8 -> F2 -> F6) x18次 -> F7 -> F3

作用是：将绿框字母字符向后偏移两位，非字母不变，再与用户输入比较。

Python脚本:

```
a=[0x66,0x63,0x6A,0x6A,0x6D,0x57,0x6D,0x73,0x45,0x6D,0x72,0x52,0x66,0x63,0x44,0x6A,0x5F,0x65]
for i in range(len(a)):
    print(chr(a[i]+2),end="")
```

Re4 obf

没做出来。看CFG图有点像ollvm混淆，之前研究的时候跑通了TSRC-bird的脚本Deflat.py，这次想用但是环境没搭好，搭环境调脚本到比赛结束。。。。