攻防世界xctf reverse： no-strings-attached

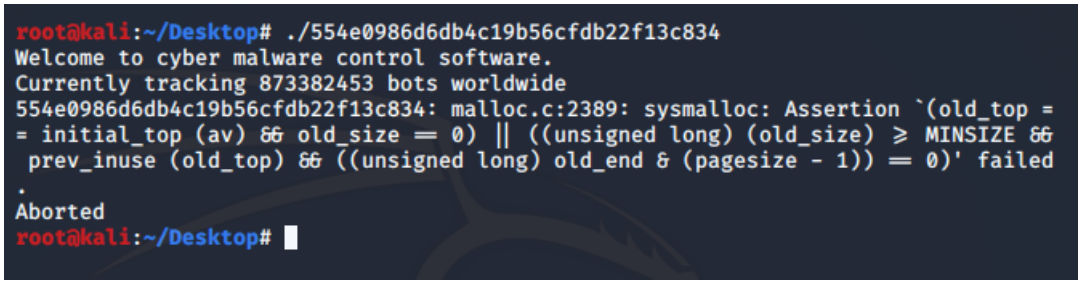
原创

prettyX 于 2020-11-24 15:20:09 发布 479 收藏
分类专栏： reverse

本文链接：https://blog.csdn.net/prettyX/article/details/110045721

先查基本信息，ELF程序，无壳

尝试运行



拽入IDA中，shift F12未发现有价值信息，F5

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   setlocale(6, &locale);
4   banner();
5   prompt_authentication();
6   authenticate();
7   return 0;
8 }
```

经过查看，authenticate()为重要函数，通过下面代码分析，S2即为我们需要的flag

```
1  void authenticate()
2  {
3    int ws[8192]; // [esp+1Ch] [ebp-800Ch]
4    wchar_t *s2; // [esp+801Ch] [ebp-Ch]
5
6    s2 = decrypt(&s, &dword_8048A90);
7    if ( fgetws(ws, 0x2000, stdin) )
8    {
9      ws[wcslen(ws) - 1] = 0;
10     if ( !wcscmp(ws, s2) )
11       wprintf((int)&unk_8048B44); success!
12     else
13       wprintf((int)&unk_8048BA4);
14   }                                Access denied!
15   free(s2);
16 }
```


s2是由decrypt()函数产生的，来看一下decrypt()函数

```
1  wchar_t *__cdecl decrypt(wchar_t *s, wchar_t *a2)
2  {
3    size_t v2; // eax
4    signed int v4; // [esp+1Ch] [ebp-1Ch]
5    signed int i; // [esp+20h] [ebp-18h]
6    signed int v6; // [esp+24h] [ebp-14h]
7    signed int v7; // [esp+28h] [ebp-10h]
8    wchar_t *dest; // [esp+2Ch] [ebp-Ch]
9
10   v6 = wcslen(s);
11   v7 = wcslen(a2);
12   v2 = wcslen(s);
13   dest = (wchar_t *)malloc(v2 + 1);
14   wcscpy(dest, s);
15   while ( v4 < v6 )
16   {
17     for ( i = 0; i < v7 && v4 < v6; ++i )
18       dest[v4++] -= a2[i];
19   }
20   return dest;
21 }
```


查看汇编代码，decrypt的返回值在eax中

```
; Attributes: bp-based frame

public authenticate
authenticate proc near

ws= dword ptr -800Ch
s2= dword ptr -0Ch

; __unwind {
push    ebp
mov     ebp, esp
sub     esp, 8028h
mov     dword ptr [esp+4], offset dword_8048A90 ; wchar_t *
mov     dword ptr [esp], offset s ; s
call    decrypt
mov     [ebp+s2], eax
mov     eax, ds:stdin@@GLIBC_2_0
mov     [esp+8], eax    ; stream
mov     dword ptr [esp+4], 2000h ; n
lea     eax, [ebp+ws]
mov     [esp], eax      ; ws
call    _fgetws
test    eax, eax
jz      short loc_804879C
```


gdb动态调试，在decrypt()函数结尾处下断点，随后读取eax中的值，获取s2

查看.text段的函数

```
objdump -t -j .text 554e0986d6db4c19b56cfdb22f13c834
```

```
smile@ubuntu:~/Desktop/RE/no-string$ objdump -t -j .text 554e0986d6db4c19b56cfdb22f13c834

554e0986d6db4c19b56cfdb22f13c834:     file format elf32-i386

SYMBOL TABLE:
08048550 l    d  .text  00000000              .text
08048580 l     F .text  00000000              __do_global_dtors_aux
080485e0 l     F .text  00000000              frame_dummy
08048860 l     F .text  00000000              __do_global_ctors_aux
08048850 g     F .text  00000002              __libc_csu_fini
08048658 g     F .text  000000b0              decrypt
08048708 g     F .text  000000a1              authenticate
08048852 g     F .text  00000000              .hidden __i686.get_pc_thunk.bx
08048604 g     F .text  0000003f              banner
080487e0 g     F .text  00000061              __libc_csu_init
08048550 g     F .text  00000000              _start
080487a9 g     F .text  00000033              main
08048643 g     F .text  00000015              prompt_authentication
```


使用disass decrypt 查看汇编代码

```
   0x080486d5 <+125>:   mov    edx,DWORD PTR [edx]
   0x080486d7 <+127>:   mov    ebx,ecx
   0x080486d9 <+129>:   sub    ebx,edx
   0x080486db <+131>:   mov    edx,ebx
   0x080486dd <+133>:   mov    DWORD PTR [eax],edx
   0x080486df <+135>:   add    DWORD PTR [ebp-0x1c],0x1
   0x080486e3 <+139>:   add    DWORD PTR [ebp-0x18],0x1
   0x080486e7 <+143>:   mov    eax,DWORD PTR [ebp-0x18]
   0x080486ea <+146>:   cmp    eax,DWORD PTR [ebp-0x10]
   0x080486ed <+149>:   jge    0x80486f7 <decrypt+159>
   0x080486ef <+151>:   mov    eax,DWORD PTR [ebp-0x1c]
   0x080486f2 <+154>:   cmp    eax,DWORD PTR [ebp-0x14]
   0x080486f5 <+157>:   jl     0x80486b8 <decrypt+96>
   0x080486f7 <+159>:   mov    eax,DWORD PTR [ebp-0x1c]
   0x080486fa <+162>:   cmp    eax,DWORD PTR [ebp-0x14]
   0x080486fd <+165>:   jl     0x80486af <decrypt+87>
   0x080486ff <+167>:   mov    eax,DWORD PTR [ebp-0xc]
   0x08048702 <+170>:   add    esp,0x34
   0x08048705 <+173>:   pop    ebx
   0x08048706 <+174>:   pop    ebp
   0x08048707 <+175>:   ret                  decrypt()
End of assembler dump.
pwndbg>
```


我们把断点，下在decrypt()函数的最后一条指令上

使用“i b”命令，查看断点信息

```
   0x08048706 <+174>:   pop    ebp
   0x08048707 <+175>:   ret
End of assembler dump.
pwndbg> b *0x08048707
Breakpoint 1 at 0x8048707
pwndbg> i b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08048707 <decrypt+175>
pwndbg>
```

r，运行到断点

n，单步执行，到decrypt()函数结束的下一行

```
0x08048725 in authenticate ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
 EAX  0x804d030 ◂— 0x39 /* '9' */
 EBX  0x0
 ECX  0x1480
 EDX  0x7d
 EDI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e8d6c
 ESI  0xf7fb7000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e8d6c
 EBP  0xffffd0c8 —▸ 0xffffd0e8 ◂— 0x0
*ESP  0xffff50a0 —▸ 0x8048aa8 ◂— cmp    dl, byte ptr [eax + eax]
*EIP  0x8048725 (authenticate+29) ◂— mov    dword ptr [ebp - 0xc], eax
[ DISASM ]
   0x8048707 <decrypt+175>        ret
    ↓
 ► 0x8048725 <authenticate+29>    mov    dword ptr [ebp - 0xc], eax
   0x8048728 <authenticate+32>    mov    eax, dword ptr [stdin@@GLIBC_2.0] <0x804a03c>
   0x804872d <authenticate+37>    mov    dword ptr [esp + 8], eax
   0x8048731 <authenticate+41>    mov    dword ptr [esp + 4], 0x2000
   0x8048739 <authenticate+49>    lea    eax, [ebp - 0x800c]
   0x804873f <authenticate+55>    mov    dword ptr [esp], eax
   0x8048742 <authenticate+58>    call   fgetws@plt <fgetws@plt>

   0x8048747 <authenticate+63>    test   eax, eax
   0x8048749 <authenticate+65>    je     authenticate+148 <authenticate+148>

   0x804874b <authenticate+67>    lea    eax, [ebp - 0x800c]
[ STACK ]
00:0000│ esp  0xffff50a0 —▸ 0x8048aa8 ◂— cmp    dl, byte ptr [eax + eax]
01:0004│      0xffff50a4 —▸ 0x8048a90 ◂— add    dword ptr [eax + eax], edx
02:0008│      0xffff50a8 ◂— 0x0
... ↓
[ BACKTRACE ]
 ► f 0  8048725 authenticate+29
   f 1  80487d5 main+44
   f 2 f7decfb9 __libc_start_main+249
pwndbg>                                                  https://blog.csdn.net/prettyX
```

i r，查看寄存器的值

```
pwndbg> i r
eax            0x804d030           134533168
ecx            0x1480              5248
edx            0x7d                125
ebx            0x0                 0
esp            0xffff50a0          0xffff50a0
ebp            0xffffd0c8          0xffffd0c8
esi            0xf7fb7000          -134516736
edi            0xf7fb7000          -134516736
eip            0x8048725           0x8048725 <authenticate+29>
eflags         0x282               [ SF IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x63                99
pwndbg>                                          https://blog.csdn.net/prettyX
```

查看eax的值：

```
x/6sw $eax
```

6：显示6行数据

s：字符串形式

w：word（4字节）形式

```
pwndbg> x/6sw $eax
0x804d030:      U"9447{you_are_an_international_mystery}"
0x804d0cc:      U""
0x804d0d0:      U""
0x804d0d4:      U""
0x804d0d8:      U""
0x804d0dc:      U""
pwndbg>
```

flag get