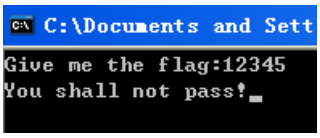
2015_XDCTF_Reverse200_Writeup

转载

weixin_30363981 于 2015-10-12 13:23:00 发布 49 收藏
原文链接：http://www.cnblogs.com/Viwilla/p/4871390.html

OD加载程序，F9运行，提示输入flag。随便输入一个字符串，提示“you shall not pass！”



这道题是输入验证类型，验证的字符串是固定的，因此相对容易分析。验证可分为6个阶段：

① 验证长度

② 验证前6个字符

③ 验证第24位、第13位、第19位

④ 验证7~12位

⑤ 验证14~18位

⑥ 验证20~23位

具体分析步骤如下：

1．验证长度

重新加载，右键“查找”->"所有参考文本字串"。显示字符串（图 2.1），发现很多“you shall not pass”,在汇编中从“you shall not pass ”往回找跳转句，在附近的比较函数（cmp）或test下断，F9运行，提示输入时输入一个字符串“12345”，回车，程序断在004011EB处，此句将堆栈 ss:[0012FE60]的内容和0x19（25）进行比较，大于等于跳转到提示错误，为零也提示错误。数据窗口中查看[0012FE60] 内存（图2.3），很明显此内存+8的地方存放了我们输入的flag，那这个内存存放的是什么呢？数据窗口中此处的值为5，可以联想到是输入flag的长度，可以调试前面的代码得证，换几个不同长度字符串输入也可以验证这一点。

往前可以找到“Give me the flag：”后面可以验证。F8单步，后面的jz和jnz都跳到了00401224，说明此段放行了长度满足0<n<25的输入字符串。

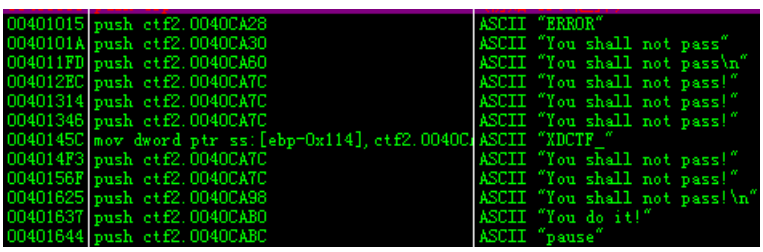


图 2.1

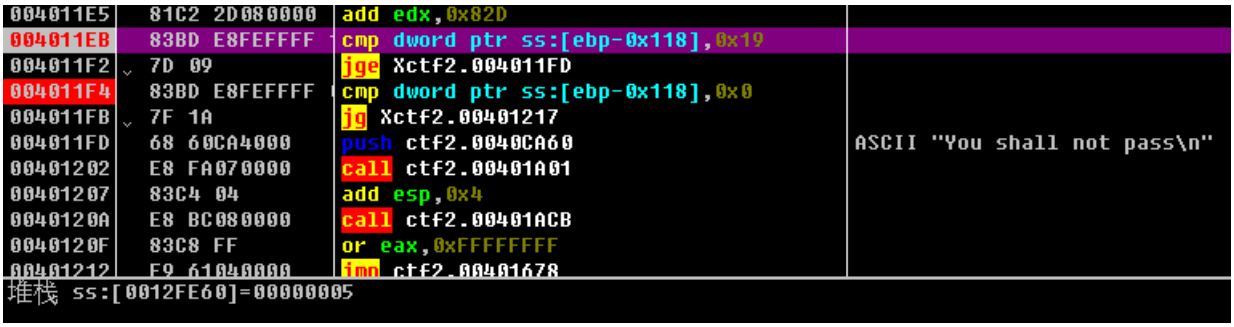


图 2.2

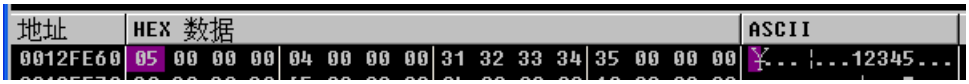


图 2.3

图 2.4

2. 验证前6个字符

F8继续单步运行，下面的代码将输入的前6个字符和“XDCTF{”比较，相等则跳到0040130F处。（代码2.1）

```
; 代码2.1
004012AC    8B8D D0FEFFFF   mov ecx,dword ptr ss:[ebp-0x130]
004012B2    83C1 01         add ecx,0x1
004012B5    898D D0FEFFFF   mov dword ptr ss:[ebp-0x130],ecx
004012BB    8B95 E0FEFFFF   mov edx,dword ptr ss:[ebp-0x120]
004012C1    83EA 01         sub edx,0x1
004012C4    3995 D0FEFFFF   cmp dword ptr ss:[ebp-0x130],edx ;  控制比较次数为6次
004012CA    7F 3C           jg Xctf2.00401308
004012CC    8B85 D0FEFFFF   mov eax,dword ptr ss:[ebp-0x130]
004012D2    0FBE8C05 70FFFF>movsx ecx,byte ptr ss:[ebp+eax-0x90]
004012DA    8B95 D0FEFFFF   mov edx,dword ptr ss:[ebp-0x130]
004012E0    0FBE8415 F0FEFF>movsx eax,byte ptr ss:[ebp+edx-0x110]
004012E8    3BC8            cmp ecx,eax         ;将输入的前六个字符和”XDCTF_”比较
004012EA    74 1A           je Xctf2.00401306
004012EC    68 7CCA4000     push ctf2.0040CA7C                      ; ASCII "You shall not pass!"
004012F1    E8 0B070000     call ctf2.00401A01
004012F6    83C4 04         add esp,0x4
004012F9    E8 CD070000     call ctf2.00401ACB
004012FE    83C8 FF         or eax,0xFFFFFFFF
00401301    E9 72030000     jmp ctf2.00401678
00401306  ^ EB A4           jmp Xctf2.004012AC
```

3. 验证第24位、第13位、第19位

分析到这里我们不妨换一个字符串输入，就取它最大允许的长度24，输入“XDCTF{123456789ABCDEFGH}”，果然程序跳到0040130F处（代码3.1）。这段代码发现将第24位和“}”比较，第13位和“_”比较，第19位和“$”比较，相等跳到0040136B。至此我们可以修改输入字符为“XDCTF{123456_ABCDE$abcd}”。

```
；代码3.1
0040130F    83F9 7D         cmp ecx,0x7D    ; 第24位和“}”比较
00401312    74 1A           je Xctf2.0040132E
00401314    68 7CCA4000     push ctf2.0040CA7C                      ; ASCII "You shall not pass!"
00401319    E8 E3060000     call ctf2.00401A01
0040131E    83C4 04         add esp,0x4
00401321    E8 A5070000     call ctf2.00401ACB
00401326    83C8 FF         or eax,0xFFFFFFFF
00401329    E9 4A030000     jmp ctf2.00401678
0040132E    0FBE95 FCFEFFFF movsx edx,byte ptr ss:[ebp-0x104]
00401335    83FA 5F         cmp edx,0x5F   ; 第13位和“_”比较
00401338    75 0C           jnz Xctf2.00401346
0040133A    0FBE85 02FFFFFF movsx eax,byte ptr ss:[ebp-0xFE]
00401341    83F8 24         cmp eax,0x24    ; 第19位和“$”比较
00401344    74 1A           je Xctf2.00401360
00401346    68 7CCA4000     push ctf2.0040CA7C                      ; ASCII "You shall not pass!"
0040134B    E8 B1060000     call ctf2.00401A01
00401350    83C4 04         add esp,0x4
00401353    E8 73070000     call ctf2.00401ACB
00401358    83C8 FF         or eax,0xFFFFFFFF
0040135B    E9 18030000     jmp ctf2.00401678
00401360    74 09           je Xctf2.0040136B  ; 通过
00401362    75 12           jnz Xctf2.00401376   ; 不通过
```

4．验证7~12位

重新运行程序，输入“XDCTF{123456_ABCDE$abcd}”，程序如预期跳到0040136B，再跳到00401376（代码4.1），F8到0040145C处，程序将”XDCTF_”放入内存0040CA90。

继续往下分析，从00401090开始，进入第四阶段的验证(代码 4.2)。代码中标注了三行关键代码，004010AA处第一次验证指向的地址为0040CA90，此处的内存存放了“XDCTF_”(图 4.1)；004010B3处第一次验证指向的地址为004010B3，存放了“123456”。004010B6处将“XDCTF_”和“123456”对应依次相减（ascii码），得到的结果在004010BE处与ds:[edx+ecx*4+0x8]中存储的数据比较，相等则通过验证。通过强制跳转分析，每一次相减得到的正确数值应为：

堆栈 ds:[0012FEF8]=00000015

堆栈 ds:[0012FEFC]=FFFFFFD5

堆栈 ds:[0012FF00]=FFFFFFD5

堆栈 ds:[0012FF04]=FFFFFFED

堆栈 ds:[0012FF08]=FFFFFFD4

堆栈 ds:[0012FF0C]=FFFFFFFE

即：7~12位正确的输入应为(需转换类型)：

“X” — 0x00000015 = “C”

“D” — 0xFFFFFFD5 = “o”

“C” — 0xFFFFFFD5 = “n”

“T” — 0xFFFFFFED = “g”

“F” — 0xFFFFFFD4 = “r”

“_” — 0xFFFFFFFE = “a”

```
; 代码4.1
0040145C    C785 ECFEFFFF 9>mov dword ptr ss:[ebp-0x114],ctf2.0040CA>; ASCII "XDCTF_"
00401466    6A 06           push 0x6
00401468    E8 12020000     call ctf2.0040167F


; 代码4.2
00401090    8B45 F8         mov eax,dword ptr ss:[ebp-0x8]
00401093    83C0 01         add eax,0x1
00401096    8945 F8         mov dword ptr ss:[ebp-0x8],eax
00401099    8B4D FC         mov ecx,dword ptr ss:[ebp-0x4]
0040109C    83E9 01         sub ecx,0x1
0040109F    394D F8         cmp dword ptr ss:[ebp-0x8],ecx
004010A2    7F 26           jg Xctf2.004010CA
004010A4    8B55 08         mov edx,dword ptr ss:[ebp+0x8]
004010A7    0355 F8         add edx,dword ptr ss:[ebp-0x8]
004010AA    0FBE02       movsx eax,byte ptr ds:[edx] ; [0040CA90]-->”XDCTF_”
004010AD    8B4D 0C         mov ecx,dword ptr ss:[ebp+0xC]
004010B0    034D F8         add ecx,dword ptr ss:[ebp-0x8]
004010B3    0FBE11       movsx edx,byte ptr ds:[ecx] ;   [004010B3]-->”123456”
004010B6    2BC2            sub eax,edx              ;对应依次相减-->eax
004010B8    8B4D F8         mov ecx,dword ptr ss:[ebp-0x8]
004010BB    8B55 10         mov edx,dword ptr ss:[ebp+0x10]
004010BE    3B448A 08       cmp eax,dword ptr ds:[edx+ecx*4+0x8] ;eax是否与ds:[edx+ecx*4+0x8]中存储的数据是否相
等
004010C2    74 04           je Xctf2.004010C8
004010C4    32C0            xor al,al
004010C6    EB 04           jmp Xctf2.004010CC
004010C8  ^ EB C6           jmp Xctf2.00401090
```

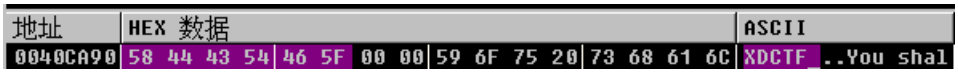


图 4.1

图 4.2

5. 验证14~18位

F8继续往下，来到004010F1处（代码5.1），从这里开始将第14~18位依次与“t”“U”、“l”、“a”、“t”比较，相等则验证通过。

```
;代码5.1
004010F1    8955 F4         mov dword ptr ss:[ebp-0xC],edx
004010F4    8B45 08         mov eax,dword ptr ss:[ebp+0x8]
004010F7    8945 F0         mov dword ptr ss:[ebp-0x10],eax
004010FA    8B4D F0         mov ecx,dword ptr ss:[ebp-0x10]
004010FD    8A11            mov dl,byte ptr ds:[ecx]
004010FF    8855 EF         mov byte ptr ss:[ebp-0x11],dl
00401102    8B45 F4         mov eax,dword ptr ss:[ebp-0xC]
00401105    3A10            cmp dl,byte ptr ds:[eax]
00401107    75 2E           jnz Xctf2.00401137
00401109    807D EF 00      cmp byte ptr ss:[ebp-0x11],0x0
0040110D    74 1F           je Xctf2.0040112E
0040110F    8B4D F0         mov ecx,dword ptr ss:[ebp-0x10]
00401112    8A51 01         mov dl,byte ptr ds:[ecx+0x1]
00401115    8855 EE         mov byte ptr ss:[ebp-0x12],dl
00401118    8B45 F4         mov eax,dword ptr ss:[ebp-0xC]
0040111B    3A50 01         cmp dl,byte ptr ds:[eax+0x1]
0040111E    75 17           jnz Xctf2.00401137
00401120    8345 F0 02      add dword ptr ss:[ebp-0x10],0x2
00401124    8345 F4 02      add dword ptr ss:[ebp-0xC],0x2
00401128    807D EE 00      cmp byte ptr ss:[ebp-0x12],0x0
0040112C  ^ 75 CC           jnz Xctf2.004010FA
```

6. 验证20~23位

继续往下跟踪，到004015D1处进入第6阶段验证。(代码 6.1)

“T”与第20个字符异或结果为0x31，该字符为“e”；

“C”与第21个字符异或结果为0x3A,该字符为“y”；

“D”与第22个字符异或结果为0xB,该字符为“W”

“X”与第23个字符异或结果为0x2D，该字符为“u”

```
;代码6.1
004015D1    0FBE8D DCFEFFFF movsx ecx,byte ptr ss:[ebp-0x124]   ; “e”
004015D8    0FBE95 F3FEFFFF movsx edx,byte ptr ss:[ebp-0x10D]   ;“T”
004015DF    33CA            xor ecx,edx
004015E1    83F9 31         cmp ecx,0x31          ;ecx=“e”⊕ “T”=0x31
004015E4    75 3F           jnz Xctf2.00401625
004015E6    0FBE85 DDFEFFFF movsx eax,byte ptr ss:[ebp-0x123] ;  ”y”
004015ED    0FBE8D F2FEFFFF movsx ecx,byte ptr ss:[ebp-0x10E]  ;”C”
004015F4    33C1            xor eax,ecx
004015F6    83F8 3A         cmp eax,0x3A         ;eax=“y”⊕ “C”=0x3A
004015F9    75 2A           jnz Xctf2.00401625
004015FB    0FBE95 DEFEFFFF movsx edx,byte ptr ss:[ebp-0x122]
00401602    0FBE85 F1FEFFFF movsx eax,byte ptr ss:[ebp-0x10F]
00401609    33D0            xor edx,eax
0040160B    83FA 0B         cmp edx,0xB        ; edx=“O”⊕ “D”=0xB
0040160E    75 15           jnz Xctf2.00401625
00401610    0FBE8D DFFEFFFF movsx ecx,byte ptr ss:[ebp-0x121]
00401617    0FBE95 F0FEFFFF movsx edx,byte ptr ss:[ebp-0x110]
0040161E    33CA            xor ecx,edx
00401620    83F9 2D         cmp ecx,0x2D        ;ecx=“u”⊕ “X”=0x2D
00401623    74 12           je Xctf2.00401637
00401625    68 98CA4000     push ctf2.0040CA98                      ; ASCII "You shall not pass!\n"
0040162A    E8 D2030000     call ctf2.00401A01
0040162F    83C4 04         add esp,0x4
00401632    E8 94040000     call ctf2.00401ACB
00401637    68 B0CA4000     push ctf2.0040CAB0                      ; ASCII "You do it!"
0040163C    E8 C0030000     call ctf2.00401A01
```

到这里，flag已浮出水面，赶紧输入一下，”you do it!”~~

Flag: XDCTF{Congra_tUlat$eyOu}

Ps: 欢迎大家指正~

By. Vi

转载于:https://www.cnblogs.com/Viwilla/p/4871390.html