alading

原创

zh_explorer 于 2015-04-27 22:29:25 发布 756 收藏

分类专栏： hduisa内部平台writeup 文章标签： Reverse ctf writeup

本文链接：https://blog.csdn.net/zh_explorer/article/details/45317959

阿拉灯的神丁 (ノ｀□′)ノ ┻━┻

(ノ｀□′)ノ ┻━┻再掀一次，坑爹啊！！

某人放的风骚逆向题目，简直作死。直接脱去分析 =ヽ(˚▽ ˚)ー C<(/;◇;)/~[拖走]

```
push    ecx             ; lpflOldProtect
push    40h             ; flNewProtect <-翻翻头文件，是可读可写可执行
push    140h            ; dwSize
push    offset loc_401000 ; lpAddress
call    ds:VirtualProtect
```

风骚的把代码段起始的0x140字节改成了可写的属性。然后输入密码。检测是否是8位，不是则提示重新输入。

下面注意这一段代码。

```
.text:004011B8 loc_4011B8:                             ; CODE XREF: _main+87j
.text:004011B8                 jmp     short loc_4011C3
.text:004011BA ; ---------------------------------------------------------------------------
.text:004011BA
.text:004011BA loc_4011BA:                             ; CODE XREF: _main+96j
.text:004011BA                 mov     edx, 1
.text:004011BF                 test    edx, edx
.text:004011C1                 jnz     short loc_401185
.text:004011C3
.text:004011C3 loc_4011C3:                             ; CODE XREF: _main:loc_401188j
.text:004011C3                 mov     [ebp+var_10], 0
.text:004011CA                 mov     [ebp+var_14], 0
.text:004011D1                 jmp     short loc_4011F8
.text:004011D3 ; ---------------------------------------------------------------------------
.text:004011D3
.text:004011D3 loc_4011D3:                             ; CODE XREF: _main+102j
.text:004011D3                 mov     eax, [ebp+var_10]
.text:004011D6                 add     eax, 1
.text:004011D9                 mov     [ebp+var_10], eax
.text:004011DC                 mov     ecx, [ebp+var_14]
.text:004011DF                 add     ecx, 1
.text:004011E2                 mov     [ebp+var_14], ecx
.text:004011E5                 mov     edx, [ebp+var_14]
.text:004011E8                 and     edx, 80000007h
.text:004011EE                 jns     short loc_4011F5
.text:004011F0                 dec     edx
.text:004011F1                 or      edx, 0FFFFFFF8h
.text:004011F4                 inc     edx
.text:004011F5
.text:004011F5 loc_4011F5:                             ; CODE XREF: _main+CEj
.text:004011F5                 mov     [ebp+var_14], edx
.text:004011F8
.text:004011F8 loc_4011F8:                             ; CODE XREF: _main+81j
.text:004011F8                 cmp     [ebp+var_10], 100h
.text:004011FF                 jge     short loc_401224
.text:00401201                 mov     eax, [ebp+var_10]
.text:00401204                 xor     ecx, ecx
.text:00401206                 mov     cl, byte_41204C[eax]
.text:0040120C                 mov     edx, [ebp+var_14]
.text:0040120F                 xor     eax, eax
.text:00401211                 mov     al, byte_414B88[edx]
.text:00401217                 xor     ecx, eax
.text:00401219                 mov     edx, [ebp+var_10]
.text:0040121C                 mov     byte_41204C[edx], cl
.text:00401222                 jmp     short loc_4011D3
.text:00401224 ; ---------------------------------------------------------------------------
.text:00401224
.text:00401224 loc_401224:                             ; CODE XREF: _main+DFj
.text:00401224                 jmp     short loc_40125D
.text:00401224 ; ---------------------------------------------------------------------------
```

主要的功能是把输入的8位的密码和0x41204C处的0x100字节的数据进行亦或加密。

然后程序跳转到这里

```
.text:0040125D ; ---------------------------------------------------------------------------
.text:0040125D
.text:0040125D loc_40125D:                             ; CODE XREF: _main:loc_401224j
.text:0040125D                 mov     [ebp+var_10], 0
.text:00401264                 jmp     short loc_40126F
.text:00401266 ; ---------------------------------------------------------------------------
.text:00401266
.text:00401266 loc_401266:                             ; CODE XREF: _main+174j
.text:00401266                 mov     eax, [ebp+var_10]
.text:00401269                 add     eax, 1
.text:0040126C                 mov     [ebp+var_10], eax
.text:0040126F
.text:0040126F loc_40126F:                             ; CODE XREF: _main+144j
.text:0040126F                 cmp     [ebp+var_10], 100h
.text:00401276                 jge     short loc_401296
.text:00401278                 mov     ecx, [ebp+var_10]
.text:0040127B                 xor     edx, edx
.text:0040127D                 mov     dl, byte_41204C[ecx]
.text:00401283                 mov     eax, offset loc_401000
.text:00401288                 mov     ecx, [ebp+var_10]
.text:0040128B                 mov     cl, byte_41214C[ecx]
.text:00401291                 mov     [eax+edx], cl
.text:00401294                 jmp     short loc_401266
.text:00401296 ; ---------------------------------------------------------------------------
.text:00401296
.text:00401296 loc_401296:                             ; CODE XREF: _main+156j
.text:00401296                 call    loc_401000
```

然后，已刚才亦或的字符串为指针，或者说是偏移地址吧。把0x41204C按照这个偏移地址表patch到0x401000处。最后再call 0x401000

=，=自解密和修改的代码(ノ｀□′)ノ ┻━┻

然后，应该怎么办？

先看看0x401000处代码，在最后发现这个

```
00401101   .  5F            pop edi
00401102   .  5E            pop esi
00401103   .  5B            pop ebx
00401104   .  83C4 58       add esp,0x58
00401107   .  3BEC          cmp ebp,esp
00401109   .  E8 42020000   call alaaladi.00401350
0040110E   .  8BE5          mov esp,ebp
00401110   .  5D            pop ebp
00401111   .  C3            retn
```

典型的函数结尾。可以猜估计patch的代码应该是个函数（~~废话~~），而且应该是标准的编译器编译函数。

```
00401120   $  55            push ebp
00401121   .  8BEC          mov ebp,esp
00401123   .  83EC 54       sub esp,0x54
00401126   .  53            push ebx
00401127   .  56            push esi
00401128   .  57            push edi
00401129   .  8D7D AC       lea edi,dword ptr ss:[ebp-0x54]
0040112C   .  B9 15000000   mov ecx,0x15
00401131   .  B8 CCCCCCCC   mov eax,0xCCCCCCCC
00401136   .  F3:AB         rep stos dword ptr es:[edi]
```

随便一看，函数开头都长的一样。就是栈帧的大小不一样。不过函数结尾处已经告诉你栈帧的大小。那么，函数开头24字节的内容可以确定了.

```
0x55 0x8B 0xEC 0x83 0xEC 0x58
0x53 0x56 0x57 0x8D 0x7D 0xAC
0xB9 0x15 0x00 0x00 0x00 0xB8
0xCC 0xCC 0xCC 0xCC 0xF3 0xAB
```

然后，加密算法有了。密文有了，明文有了。写段代码把密码跑出来吧。

```
#include <srdio.h>
int main(void)
{   //要被加密的偏移地址表
    int pass[64]={  0xE8D24C6B,    0xC9BB168E,    0x9D25F51A,    0xD48693D4,
                    0xEB50BEEE,    0x43F44463,    0x6A706816,    0x75B33DD8,
                    0xE72FAD05,    0x1A377970,    0x31495D41,    0xF191B1BB,
                    0x7F5720BD,    0x475F70E6,    0x2C8235D2,    0x91904C17,
                    0x23C1B9D6,    0x0129887B,    0x5CA80E98,    0xFAFC58CE,
                    0x6EA6D592,    0xE74108D6,    0xDEB9A036,    0x34A1B385,
                    0xE24102F7,    0x063A95EC,    0xA1E30CB7,    0x71F164BE,
                    0x269C78AE,    0x5E85AD18,    0xF398117B,    0xA85823CB,
                    0xB5192D9B,    0x90AD9F45,    0x3362E463,    0x6216F76F,
                    0x74EE625D,    0x3309E81F,    0x94E9D486,    0x14691029,
                    0x961A0D67,    0x9914F2AE,    0x0C674D90,    0xED4AED52,
                    0x7DC96D8A,    0xFCF8D3E3,    0xD9EAC2E0,    0x16232F88,
                    0x1609D1AA,    0x0AB25C0C,    0x5358852A,    0x7C623B8C,
                    0xC613C5BB,    0x7757C234,    0x2E5D8904,    0xF44C9E13,
                    0x7B6D0AE7,    0x5094BC39,    0xCF300BF5,    0xD1AAE2AC,
                    0xD0427C19,    0xB53531FB,    0x81CE7FE4,    0xCCD7A794,};
    //被patch的code
    int code[64]={
                    0xF8003320,    0x8BF8F80C,    0x0001F817,    0x53EB4145,
                    0x8A4530C2,    0x41F400B9,    0x53001683,    0xCC008333,
                    0x83090183,    0x7D8883C7,    0x0000FC8D,    0xDC83F88B,
                    0xCCC74541,    0x45407D15,    0x00300800,    0x00C48901,
                    0xF850208B,    0x058B68CC,    0x74554545,    0x00883099,
                    0xEC55F491,    0x7D1A0958,    0xC04DF8F8,    0x8B8B8900,
                    0x8B80E800,    0x8BD22055,    0x454DDA33,    0xF30000FC,
                    0x89410090,    0x502033C1,    0x45004D42,    0x3BF59107,
                    0x8AF8C019,    0xE80088C0,    0xC200158A,    0x58F86455,
                    0xC78B56CC,    0x4D884400,    0xD43303F8,    0x8BECC7F8,
                    0x0083EB00,    0x088944EB,    0x01A8F0D2,    0xE855C7F8,
                    0xB8035709,    0x00EB03F0,    0xD8C0C1F8,    0xEB7DF830,
                    0x4D548B8B,    0x00838BF8,    0x00EBEA45,    0x00000045,
                    0x8908FCFC,    0x0058C04B,    0xEB04D6F8,    0x0DF0F883,
                    0xAB8B0045,    0x45018333,    0xE2410000,    0x5BC2458B,
                    0xF8FB457D,    0x7D8A45E8,    0x6883FCF4,    0x45C74100,};
    int i,j,k,num=0;
    //这是函数开头24字节的的内容
    char x[24]={
                    0x55, 0x8B, 0xEC, 0x83, 0xEC, 0x58,
                    0x53, 0x56, 0x57, 0x8D, 0x7D, 0xAC,
                    0xB9, 0x15, 0x00, 0x00, 0x00, 0xB8,
                    0xCC, 0xCC, 0xCC, 0xCC, 0xF3, 0xAB,};
    char *a;
    char *p;
    char flag;
    for(k=0;k<24;k++)
```

```
    {
        for(i=0;i<64;i++)
        {
            p=(char *)(pass+i);
            a=(char *)(code+i);
            for(j=0;j<4;j++)
            {
                if(x[k]==*(a+j))
                {
                    flag=*(p+j)^num;
                    if(flag>32&&flag<127)
                        printf("%c %d\t",flag,(i*4+j+1)%8);
                }
            }
        }
        printf("\n");
        num++;
    }
    return 0;
}
```

代码的思路是。在code表中找到与函数开头相同的8位机器码，计算索引。然后在pass表的相应位置取出偏移地址，和函数机器码的偏移地址亦或，就是密码。然后pass表索引除以8的余数就是密码字符串的索引。

```
 0:o 5  J 7
 1:( 7  5 0 ] 6 l 3
 2:l 4  k 7
 3:> 6  z 6
 4:j 4  m 7
 5:g 0  R 7
 6:l 4
 7:e 2
 8:e 2
 9:H 1
10:z 6  ) 7
11:
12:o 5
13:
14:J 6  ~ 3    G 3 ? 4 " 4 j 6 v 2 l 3 i 1 ] 4 5 6 l 7 r 0 y 0
15:K 6  F 3 > 4 # 4 k 6 w 2 m 3 h 1 \ 4 4 6 m 7 s 0 x 0
16:T 6  ` 3 Y 3 ! 4 < 4 t 6 h 2 r 3 w 1 C 4 + 6 r 7 l 0 g 0
17:l 4
18:g 0  m 4 i 5 O 1
19:f 0  l 4 h 5 N 1
20:a 0  k 4 o 5 I 1
21:` 0  j 4 n 5 H 1
22:g 0
23:l 4
```

因为代码的机器码可能有重复，所以密码会有多种情况。不过只有一种情况的话，那一定是密码了。
喜闻乐见的填字游戏=,=
最后密码是“Hellozmg”
(╯‵□′)╯ ﹀┻━┻再掀一次