实验吧逆向catalyst-system Writeup

转载

weixin_30799995 于 2018-08-25 22:03:00 发布 24 收藏
原文链接：http://www.cnblogs.com/zUotTe0/p/9535556.html


```
.text:0000000000400D11                add     rax, 8
```

下载之后查看知道为ELF文件，linux中执行之后发现很慢；

拖入ida中查看发现有循环调用 sleep 函数:

```
.text:0000000000400EA5 loc_400EA5:                             ; CODE XREF: main+D2↑j
.text:0000000000400EA5                cmp     [rbp+var_4], 0
.text:0000000000400EA9                jle     short loc_400E67
```

这是已经改过了，edit -> patch program -> change byte 修改一下比较参数可以去除等待时间，总共去除两处；

运行之后发现是输入 username 和 passsword 的方式；

那么问题就在两者上面了。

反编译之后查看源码，v8对应username, v7为password.

```
50  putchar(10);
51  sub_400C9A(v8);
52  sub_400CDD(v8);
53  sub_4008F7(v8);
54  sub_400977(v8, v7);
55  sub_400876(v8, v7);
56  return 0LL;
57}
```

点进第一个函数，在 return 中是对用户名的判断，易推断出用户名为 12 位长度(0b1100);

再看第二个函数，也是对用户名进行判断：

```
8   v4 = *a1;
9   v3 = a1[1];
10  v2 = a1[2];
11  if ( v4 - v3 + v2 != 1550207830
12    || v3 + 3 * (v2 + v4) != 12465522610LL
13    || (result = 0x32AC30689A6AD314LL, v2 * v3 != 0x32AC30689A6AD314LL) )
14  {
15    puts("invalid username or password");
16    exit(0);
17  }
```

```
.text:0000000000400D01                add     rax, 4
```

```
.text:0000000000400D11                add     rax, 8
```

关于v4, v3, v2 的赋值，切到对应汇编代码处看出以4个子节为一个 Int 进行赋值；之后就是一个三元一次方程，可以得到用户名；

要注意用户名的顺序因为文件是小端序的，所以为 "catalyst_ceo"；

第三个函数也是验证用户名正确性；

第四个函数，是对密码的验证，由此可以反推出密码；

```
27  srand(a1[1] + *a1 + a1[2]);
28  v2 = *a2;
29  if ( v2 - rand() != 1441465642 )
30  {
31    puts("invalid username or password");
32    exit(0);
33  }
34  v3 = a2[1];
35  if ( v3 - rand() != 251096121 )
```

srand((unsigned int) seed)是C语言中的一个随机数发生器初始化函数，而种子则为用户名所对应的三个 int 值之和，因为种子为定值，所以rand()产生的随机数序列是不变的；之后写CPP文件得出随机值，linux中运行得到10个随机序列（Linux中返回最大32位随机值，而 Windows 最大为16位），还要注意右边比较值有正负之分，得到总共10个与密码相关的数字，这时再注意到每一个 vx 值都为 Int 型，相当于 4 个char型；

在这里由用户名和密码就可以获得 flag 了，密码每四个逆序排列；

Welcome to Catalyst systems
Loading.
Username: catalyst_ceo
Password: sLSVpQ4vK3cGWyW86AiZhggwLHBjmx9CRspVGggj
Logging in.
your flag is: ALEXCTF{1_t41d_y0u_y0u_ar3__gr34t__reverser__s33}

最后一个函数是flag生成：

```
6  printf("your flag is: ALEXCTF{", a2, a1);
7  for ( i = 0; i < strlen(s); ++i )
8    putchar((unsigned __int8)byte_6020A0[i] ^ s[i]);
9  return puts("}");
10}
```

s为密码串，总共为10*4 = 40 个字符；

双击byte_6020A0 数据处，刚好40个字符与得到的字符进行异或；

```
.data:00000000006020A0 byte_6020A0     db 42h                  ; DATA XREF: sub_400876+2E↑r
.data:00000000006020A1                 db  13h
.data:00000000006020A2                 db  27h ; '
.data:00000000006020A3                 db  62h ; b
.data:00000000006020A4                 db  41h ; A
.data:00000000006020A5                 db  35h ; 5
.data:00000000006020A6                 db  6Bh ; k
.data:00000000006020A7                 db  0Fh
.data:00000000006020A8                 db  7Bh ; {
.data:00000000006020A9                 db  46h ; F
.data:00000000006020AA                 db  3Ch ; <
.data:00000000006020AB                 db  3Eh ; >
.data:00000000006020AC                 db  67h ; g
.data:00000000006020AD                 db  0Ch
.data:00000000006020AE                 db    8
.data:00000000006020AF                 db  59h ; Y
.data:00000000006020B0                 db  44h ; D
```

选中数据，edit -> extractdata 得到数组；

注意因为程序为小端序，所以最后每四个字符逆序异或；

```
_cmp = [
    1441465642,
    251096121,
    -870437532,
    -944322827,
    647240698,
    638382323,
    282381039,
    -966334428,
    -58112612,
    605226810
]
_rand = [
    0x684749,
    0x673ce537,
    0x7b4505e7,
    0x70a0b262,
    0x33d5253c,
    0x515a7675,
    0x596d7d5d,
    0x7cd29049,
    0x59e72db6,
    0x4654600d
]

_xor = [
    0x42, 0x13, 0x27, 0x62, 0x41, 0x35, 0x6B, 0x0F, 0x7B, 0x46,
    0x3C, 0x3E, 0x67, 0x0C, 0x08, 0x59, 0x44, 0x72, 0x36, 0x05,
    0x0F, 0x15, 0x54, 0x43, 0x38, 0x17, 0x1D, 0x18, 0x08, 0x0E,
    0x5C, 0x31, 0x21, 0x16, 0x02, 0x09, 0x18, 0x14, 0x54, 0x59
]

p = 0
L = b''

for i in range(10):
    tmp = hex(_cmp[i] + _rand[i])[2:]
    # print(tmp)
    while tmp:
        L += bytes([int(tmp[:2], base=16) ^ _xor[p//4*4 + 3-p%4]])
        tmp = tmp[2:]
        p += 1

LL = b''
while L:
    LL += L[:4][::-1]
    L = L[4:]
print(LL)
```

得到flag：ALEXCTF{1_t41d_y0u_y0u_ar3__gr34t__reverser__s33}

转载于:https://www.cnblogs.com/zUotTe0/p/9535556.html