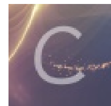
XCTF-Reverse-ExerciseArea-010-writeup

原创

y4ung 于 2019-08-08 14:46:37 发布 4093 收藏
分类专栏： ctf 文章标签： ctf
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/qq_35056292/article/details/98870901
版权
 ctf 专栏收录该内容
35 篇文章 0 订阅
订阅专栏

0x00 介绍

本题是xctf攻防世界中Reverse的新手第十题。题目来源： `SharifCTF 2016`

给了一个二进制文件getit，需要对该二进制文件进行逆向分析，找到flag

实验环境：IDA Pro 7.0，gdb

0x01 解题过程

1.1 文件分析

Linux下的二进制文件，64位系统，符号表未被去掉

```
root@kali:~/hzy/ctf-learning# file getit
getit: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=e389cd7a4b9272ba80f85d7eb604176f6106c61e, not stripped
root@kali:~/hzy/ctf-learning# chmod +x getit
root@kali:~/hzy/ctf-learning# ./getit
root@kali:~/hzy/ctf-learning#
```

1.2 逆向分析

> 用IDA打开，发现该二进制文件未被加壳，因此不需要进行脱壳操作
>
> 基本块0x400775–0x400788是循环的条件判断，地址 `rbp+var_40` 中存储的是循环变量的值，先赋值给eax寄存器，再赋值给ebx，然后调用strlen函数，得到字符串s= `c61b68366edeb7bdce3c6820314b7498` 的长度，保存在eax寄存器中；接下来如果ebx的值，也就是循环变量的值比长度小，就进入循环体，否则退出
>
> 基本块0x40078a–0x4007a6是关键的部分。从字符串s= `c61b68366edeb7bdce3c6820314b7498` 中取s[eax]，并赋值给寄存器eax，然后赋值给寄存器ecx；如果循环变量的值（存在eax中）为偶数，则eax赋值为0xffffffff，否则赋值为0x1；
>
> 基本块0x4007b4–0x4007c5是主要的部分。寄存器ecx的值加上eax的值，保存在ecx中；这里有个edx寄存器的值给了eax，然后将寄存器ecx的值，也就是当前字符串s的字符赋值给字符串t[rax]。往上溯源看下edx就可以发现，在基本块0x40078a–0x4007a6中， `lea edx, [rax+0Ah]` ，edx每次都通过eax来自增1，因此edx是用来偏移字符串t的。

打印一下发现字符串t其实就是flag字符串，格式为`ShariCTF{'?'repet 32 times}`也就是说，每次循环都经过3和4的操作，通过字符串s的字符与eax寄存器相加以后的字符，对flag字符串中的`?`进行填充：

```
应用程序 ▾   位置 ▾   终端 ▾                星期四 09:31
root@kali: ~/hzy/ctf-learning
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
root@kali: ~/hzy/ctf-learning        root@kali: ~/hzy/ctf-learning
RIP: 0x4007bb (<main+101>:      mov    BYTE PTR [rax+0x6010e0],cl)
R8 : 0xffff
R9 : 0x7ffff7facd80 --> 0x0
R10: 0xfffffffffffff449
R11: 0x7ffff7e88410 (<__strlen_sse2>:   pxor   xmm0,xmm0)
R12: 0x400660 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffe190 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x213 (CARRY parity ADJUST zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4007b4 <main+94>:  add    eax,ecx
   0x4007b6 <main+96>:  mov    ecx,eax
   0x4007b8 <main+98>:  movsxd rax,edx
=> 0x4007bb <main+101>: mov    BYTE PTR [rax+0x6010e0],cl
   0x4007c1 <main+107>: add    DWORD PTR [rbp-0x40],0x1
   0x4007c5 <main+111>: jmp    0x400775 <main+31>
   0x4007c7 <main+113>: movabs rax,0x616c662f706d742f
   0x4007d1 <main+123>: mov    QWORD PTR [rbp-0x30],rax
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe070 --> 0x0
0008| 0x7fffffffe078 --> 0x40093d (<__libc_csu_init+77>:        add    rbx,0x1)
0016| 0x7fffffffe080 --> 0x7ffff7fe44c0 (<_dl_fini>:    push   rbp)
0024| 0x7fffffffe088 --> 0x0
0032| 0x7fffffffe090 --> 0x4008f0 (<__libc_csu_init>:   push   r15)
0040| 0x7fffffffe098 --> 0x1563a31ea25da600
0048| 0x7fffffffe0a0 --> 0x7fffffffe190 --> 0x1
0056| 0x7fffffffe0a8 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x00000000004007bb in main ()
gdb-peda$ x/32xb 0x6010ea
0x6010ea <t+10>:        0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f
0x6010f2 <t+18>:        0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f
0x6010fa <t+26>:        0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f
0x601102 <t+34>:        0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f    0x3f
gdb-peda$ x/s 0x6010ea
0x6010ea <t+10>:        '?' <repeats 32 times>, "}"
gdb-peda$ x/s 0x6010e0
0x6010e0 <t>:   "SharifCTF{", '?' <repeats 32 times>, "}"
gdb-peda$
```

https://blog.csdn.net/qq_35056292

用代码解出flag填充的内容：

```python
s = "c61b68366edeb7bdce3c6820314b7498"

res = ""

for i, each in enumerate(s):
    each_ascii = ord(each)

    if i % 2 == 0:
        eax = 0xffffffff
        tmp = eax + each_ascii
        tmp = int("0x" + hex(tmp)[3:], 16)
    else:
        eax = 1
        tmp = eax + each_ascii


    tmp = chr(tmp)
    res = res + tmp


print(res)
```

那么flag为：SharifCTF{b70c59275fcfa8aebf2d5911223c6589}