NSSCTF_SWPU新生赛Crypto_wp

原创

M3ng@L 于 2021-10-23 21:26:43 发布 199 收藏 1

文章标签： 1024程序员节 密码学 python

本文链接：https://blog.csdn.net/qq_51999772/article/details/120926547

这是复现的题目的wp及个人心得，有很多借鉴了大佬的文章及wp，这里就不一一粘贴链接了

Crypto 8

题目：

> 73E-30U1&>V-H965S95]I<UJP;W=E<GT`

UUencode:

> Uuencode是二进制信息和文字信息之间的转换编码
> Uuencode将输入文字以每三个字节为单位进行编码，如此重复进行。如果最后剩下的文字少于三个字节，不够的部份用零补齐。这三个字节共有24个Bit，以6-bit为单位分为4个群组，每个群组以十进制来表示所出现的数值只会落在0到63之间。将每个数加上32，所产生的结果刚好落在ASCII字符集中可打印字符（32-空白...95-底线）的范围之中

Crypto 7

题目：

> 69f7906323b4f7d1e4e972acf4abfbfc,得到的结果用NSSCTF{}包裹。

由0-9；a-f组成的不一定是十六进制转换

也有可能是md5加密后的密文；md5加密又分为16位加密或者32位加密；这里的密文是16位的（可以当作判断条件）

Crypto 5

题目：

> flag=
> 25166751653530941364839663846806543387720865339263370907985655775152187319464715737116599171477207047430065345882626259880756839094179627032623895330242655333
> n=
> 13410948148270371321483802303541805256700087058716079693570858469413250739421136365242016093118533228040643729021051209066397763473086403237097740717973194006863453607928452802073998866571320081502134270036992251840696835645573639373894612801397364323522832797117071197968393196485456390498066985066062856141 9

试着分解n，在线网站和yafu都不可以

但是这里的c相对n很小；

由 $c = m \ (mod n)$

这里有两种情况：

> $m > n$.
> $c + k*$
> 这样就爆破k；当c+k*n能够被e整除时即为m
> $m < n$.
> c开e次方即为m

由于这里e也未知；但是常见的e一般是3，代进去试一试，或者爆破e也可

代码实现：

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

c = 25166751653530941364839663846806543387720865339263370907985655775152187319464715737116599171477207047430065345882626259880756839094179627032623895330242655333
n = 13410948148270371321483802303541805256700087058716079693570858469413250739421136365242016093118533228040643729021051209066397763473086403237097740717973194006863453607928452802073998866571320081502134270036992251840696835645573639373894612801397364323522832797117071197968393196485456390498066985066062856141 9


e = 3 # 假设
k = 0 #注意k是从0开始的；因为还有可能是m^e小于n
while True:
    if gmpy2.iroot(c + k * n,e)[1]:
        m = gmpy2.iroot(c + k * n,e)[0]
        break
    else:
        k += 1
flag = long_to_bytes(m)
print(flag)
# if m**e < n:
#     print("s")
```

Crypto 3

题目：

```
from gmpy2 import *
from Crypto.Util.number import *
flag  = '*******************'

p = getPrime(512)
q = getPrime(512)
m1 = bytes_to_long(bytes(flag.encode()))

n = p*q

flag1 = pow(m1,p,n)
flag2 = pow(m1,q,n)
print('flag1= '+str(flag1))
print('flag2= '+str(flag2))
print('n= '+str(n))


#flag1= 17893542812755845772427795161304049467610774531005620109503081344099161906017295486868699578946474114607624347167976713200068059018517606363517478396368430072890681401898145302336139240273132723451063402106360810413024642916851746118524166947301681245568333254648265529408446609050354235727237078987509705857
#flag2= 95580409405085606847879727622943874726633827220524165744517624606566789614499137069562997931972825651309707390763700301965277040876322904891716953565845966918293178547100704981251056401939781365264616997055296773593435626490578886752446381493929807909671245959154990639046333135728431707979143972145708806954
#n= 14045732358382416033898931768969810273834106196776815387964650542235854472060747614097706405362900576455133908212033722367233097929837365376678262097345409550748411856588488562332875164866037989459206343692490389498699474639450853972145935520018408947097777207572031948283992385697916631970047434904232689897l
```

观察题目

$c1= m \ \% \ n= m \ \% \ (n* \ )$

$c2= m \ \% \ n= m \ \% \ (n* \ )$

所以：

$c1 \ \% \ p= m \ \ (mod \ p)$

$c2 \ \% \ q= m \ \ (mod \ q)$

又由费马小定理：

$$a \quad \equiv 1 \ (mod \ p)$$

可知：$p)$

联立可得：

$c1 \equiv m \ (mod \ p)$

$c2 \equiv m \ (mod \ q)$

也即：

$c1= m + k \ \cdot$

$c2= m + k \ \cdot$

所以

$c1*c2= m +$

$m*(c1+ \quad 2)$

两式抵消：

$m - m*(c1+ \quad 2)$

所以这时可以使用sagemath进行同余式方程的求解

```
#sagemath代码
c1 = 17893542812755845772427795161304049467610774531005620109503081344099161906017295486868699578946474114607624347167976713200068059018517606363517478396368430072890681401898145302336139240273132723451063402106360810413024642916851746118524166947301681245568333254648265529408446609050354235727237078987509705857
c2 = 55804094050856068478797276229438747266338272205241657445176246065667896144991370695629979319728256513097073907637003019652770408763229048917169535658459669182931785471007049812510564019397813652646169970552967735934356264905788867524463814939298079096712459591549906390463331357284317079791439721457088 06954
n = 14045732358382416033898931768969810273834106196776815387964650542235854472060747614097706405362900576455133908212033722367233097929837365376678262097345409550748411856588488562332875164866037989459206343692490389498699474639450853972145935520018408947097777207572031948283992385697916631970047434904232 6898971

PR.<m> = PolynomialRing(Zmod(n))
f = m^2-(c1+c2)*m+c1*c2
x0= f.small_roots(X=2^400) #经计算m的范围是在2^400以内
for i in x0:
    print(i)
```

得到i也即是m；转换为bytes类型即可

Crypto 1

题目：

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



flag  = '**************************'
flag = {"asfajgfbiagbwe"}
p = getPrime(2048)
q = getPrime(2048)
m1 = bytes_to_long(bytes(flag.encode()))

e1e2 = 3087
n = p*q
print()

flag1 = pow(m1,e1,n)
flag2 = pow(m1,e2,n)
print('flag1= '+str(flag1))
print('flag2= '+str(flag2))
print('n= '+str(n))


#flag1= 463634070971821449698012827631572665302589213868521491855038966879005784397309389922926838028598122795187
584361359142761652619958273094398420314927073008031088375892957173280915904309949716842152249806486027920136603248454946737961650252641668562626310035983343018705370077783879047584582817271215517599531278507300104564011142229942160380563527291388260832749808727470291331902902518196932928128107067117198707209620169906575791373793854773799564060536121390593687449884988936522369331738199522700261116496965863870682295858957952661531894477603953742494526632841396338388879198270913523572980574440793543571757278020533565628285714358815083303489096524318164071888139412436112963845619981511061231001617406815056986634680975142352197476024575809514978857034477688443230263761729039797859679474548105510091080314572941648406111575247191733432594858810892529386644566376733736242444315001396118161944126792698184800910746657631468596147874835238845211404211589224327251424508160460779824381758673754666305973734468713088186135742308444802795989340244530329908960608193104121703595514393956745678210720344789834528473103815037772244732920207837587054152953984005141575943608338440820365961331353509434377223869139344747536480617159 4
#flag2= 1309595342757044532162823348150346472658756327817987509016277738268126573392743624062462979254112918221931914834098473233151103937290207005269467127867933809916750081285618636310810952222262857884129703625183987574237052161123135331553903152048755166454593706297062778762116567532479842823797318507704479785378550703793249352827893274286252599452500667740496509514650437000889589657620544186158380493407246393733512489334943555919342363605067784967410510641567710927980051125341620501650954300650008279160968934085697510855503796205582829422546069788190338855392214163358483190820548061488594277131442867775162517244743196139603277996437232782059692536365146847574090590033482291513412004517852883955964845634802612129631140710649795598123275824746748122252606167570998908969003400079905855014704847627523627349682975325336548461909005710176359593858839458583349958843417679056195675053417520475897318158684892956905741097588250213866984406706113611271708966890151084324084907637235946732994723360655753016810555830845478477331688010301912621221303696874972369597603668741060438015424933922274248909255957341504875867574843046099458279257623828895927437096824852292676047719445354695578601208784913299847924485971072563257833469044 8
#n= 609305637099654478882754880905638123124918364116173050874864700996165096776233155524277418132679727857702738043786588380577485490575591029930152718828075976000078971987922107645530323356525126496562423491563365836491753476840795804040219013880969539154444387313029522565456897962200817021423704204077133003361140660038327458057898764857872645377236870759691588009666047187685654297678987435769051762120388537868493789773766688347724903911796741124237476823452505450704989455260077833828660552130714794889208291939055406292476845194489525212129635173284301782141617878483740788532998492403101324795726865866661786740345862631916793208037250277376942046905892342213663197755010315060990871143919384283302925469309777769989798197913048813940747488087191697903624669415774198027063997058701217124640082074789591591494106726857376728759663074734040755438623372683762856958888826373151815914621262862750497078245369680378038995425628467728412953392359090775734440671874387905724083226246587924716226512631671786591611586774947156657178654343092123117255372954798131265566301316033414311712092913492774989048057650627801991277862963173961355088082419091848569675686058581383542877982979697235829206442087786927939745804017455244315305118437
```

观察题目

相同的m，不同的e，很像共模攻击；

但是e1，e2的具体值不确定只有乘积知道；

分解e1,e2的乘积得到3^2*7^3

由于这个有多种组合方式，导致e1，e2不一定是互质的

而共模攻击的原理：

> $c1 * c2 = m \quad (mod n)$
> 其中$e1 * x +$

在原来的应用过程中，一般e1，e2互素；所以$gcd(e1,e2) = 1$

而这里e1，e2不一定互素；就直接带入最大公约数即可

使得等式左侧加上$k*n$直接开方得到m；其中k是需要爆破的

代码实现：

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

flag1= 463634070971821449698012827631572665302589213868521491855038966879005784397309389922926838028598122795187584361359142761652619958273094398420314927073008031088375892957173280915904309949716842152249806486027920136603248454946737961650252641668562626310035983343018705370077783879047584582817271215517599531278507300104564011142229942160380563527291388260832749808727470291331902902518196932928128107067117198707209620169906575791373793854773799564060536121390593687449884988936522369331738199522700261116496965863870682295858957952661531894477603953742494526632841396338388879198270913523572980574440793543571757278020533565628285714358815083303489096524318164071888139412436112963845619981511061231001617406815056986634680975142352197476024575809514978857034477688443230263761729039797859697947454810551009108031457294164840611157524719173343259485881089252938664456637673337362424443150013961181619441267926981848009107466576314685961478748352388452114042115892243272514245081604607798243817586737546663059737344687130881861357423084448027959893402445303299089606081931041217035955143939567456782107203447898345284731038150377722447329202078375870541529539840051415759436083384408203659613313535094343772238691393447475364806171594
flag2= 130959534275704453216282334815034647265875632781798750901627773826812657339274362406246297925411291822193191483409847323315110393729020700526946712786793380991675008128561863631081095222226285788412970362518398757423705216112313533155390315204875516645459370629706277876211656753247984282379731850770447978537855070379324935282789327428625259945250066774049650951465043700088958965762054418615838049340724639373351248933494355591934236360506778496741051064156771092798005112534162050165095430065000827916096893408569751085550379620558282942254606978819033885539221416335848319082054806148859427713144286777516251724474319613960327799643723278205969253636514684757409059003348229151341200451785288395596484563480261212963114071064979559812327582474674812225260616757099890896900340007990585501470484762752362734968297532533654846190900571017635959385883945858334995884341767905619567505341752047589731815868489295690574109758825021386698440670611361127170896689015108432408490763723594673299472336065575301681055583084547847733168801030191262122130369687497236959760366874106043801542493392227424890925595734150487586757484304609945827925762382889592743709682485229267604771944535469557860120878491329984792448597107256325783346904408
n= 609305637099654478882754880905638123124918364116173050874864700996165096776233155524277418132679727857702738043786588380577485490575591029930152718828075976000078971987922107645530323356525126496562423491563366583649175347684079580404021901388096953915444438731302952256545689796220081702142370420407713300336114066003832745805789876485787264537723687075969158800966604718768565429767898743576905176212038853786849378977376668834772490391179674112423747682345250545070498945526007783382866055213071479488920829193905540629247684519448952521212963517328430178214161787848374078853299849240310132479572686586666178674034586263191679320803725027737694204690589234221366319775501031506099087114391938428330292546930977776998979819791304881394074748808719169790362466941577419802706399705870121712464008207478959159149410672685737672875966307473404075543862337268376285695888882637315181591462126286275049707824536968037803899542562846772841295339235909077573444067187438790572408322624658792471622651263167178659161158677494715665717865434309212311725537295479813126556630131603341431171209291349277498904805765062780199127786296317396135508808241909184856967568605858138354287798297969723582920644208778692793974580401745524431530511843 7
e1e2 = 3087

fac = [3,3,7,7,7]
factor = [3,7,3*7,3*3,7*7,3*7*7,3*3*7,7*7*7,3*7*7*7,3*3*7*7]
for e1 in factor:
    if e1e2 % e1 == 0:
        e2 = e1e2 // e1
        _,s,t = gmpy2.gcdext(e1,e2)
        gcd = gmpy2.gcd(e1,e2)
        temp = pow(flag1,s,n) * pow(flag2,t,n) % n
        for k in range(2**20):
            temp2 = gmpy2.iroot(temp + k*n,gcd)
            if temp2[1]:
                m = temp2[0]
                print(long_to_bytes(m))
                break
```