

BUU Crypto解析

原创

LOI哦糯米糕 于 2020-10-20 14:24:34 发布 420 收藏
版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_43404314/article/details/106172173
版权

[GXCTF2019]CheckIn

打开题目所提供的文件，发现内容有点像base64解密文件，解密后得到

Base64.us Base64 在线编码解码 (最好的 Base64 在线工具)

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

```
dIkqTCpRjA8fUBIMD5GNDkwMjNARKUwl0BFTg==
```

编码 (Encode)

解码 (Decode)

交换

(编码快捷键: **Ctrl** + **Enter**)

Base64 编码或解码的结果:

编/解码后自动全选

```
v)*L* F0<>@H0>F49023@FE0#@EN
```

https://blog.csdn.net/weixin_43404314

看到解码出来的结果既有数字又有字母符号，这让我想到了rot，

ROT5、**ROT13**、**ROT18**、**ROT47** 编码是一种简单的码元位置顺序替换暗码。此类编码具有可逆性，可以自我解密，主要用于应对快速浏览，或者是机器的读取，而不让其理解其意。

ROT5 是 rotate by 5 places 的简写，意思是旋转5个位置，其它皆同。下面分别说说它们的编码方式：

ROT5：只对数字进行编码，用当前数字往前数的第5个数字替换当前数字，例如当前为0，编码后变成5，当前为1，编码后变成6，以此类推顺序循环。

ROT13：只对字母进行编码，用当前字母往前数的第13个字母替换当前字母，例如当前为A，编码后变成N，当前为B，编码后变成O，以此类推顺序循环。

ROT18：这是一个异类，本来没有，它是将ROT5和ROT13组合在一起，为了好称呼，将其命名为ROT18。

ROT47：对数字、字母、常用符号进行编码，按照它们的ASCII值进行位置替换，用当前字符ASCII值往前数的第47位对应字符替换当前字符，例如当前为小写字母z，编码后变成大写字母K，当前为数字0，编码后变成符号_。用于ROT47编码的字符其ASCII值范围是33-126，具体可参考[ASCII编码](#)。

这个跟rot47有点像，于是rot47解码 (<https://www.qqxiuzi.cn/bianma/ROT5-13-18-47.php>) 得到

ROT5/13/18/47编码转换

ROT47 编码: (字母、数字、标点)

GXY{Y0u_kNow_much_about_Rot}

ROT47 ROT18 ROT13 ROT5 复位

(点击第一次加密 点击第二次解密)

ROT5、ROT13、ROT18、ROT47 编码是一种简单的码元位置顺序替换暗码。此类编码具有可逆性，可以自我解密，主要用于应对快速浏览，或者是机器的读取，而不让其理解其意。

ROT5 是 rotate by 5 places 的简写，意思是旋转5个位置，其它皆同。下面分别说说它们的编码方式：

ROT5: 只对数字进行编码，用当前数字往前数的第5个数字替换当前数字，例如当前为0，编码后变成5，当前为1，编码后变成6，以此类推顺序循环。

ROT13: 只对字母进行编码，用当前字母往前数的第13个字母替换当前字母，例如当前为A，编码后变成N，当前为B，编码后变成O，以此类推顺序循环。

ROT18: 这是一个异类，本来没有，它是将ROT5和ROT13组合在一起，为了好称呼，将其命名为ROT18。

ROT47: 对数字、字母、常用符号进行编码，按照它们的ASCII值进行位置替换，用当前字符ASCII值往前数的第47位对应字符替换当前字符，例如当前为小写字母z，编码后变成大写字母K，当前为数字0，编码后变成符号_。用于ROT47编码的字符其ASCII值范围是33 - 126，具体可参考ASCII编码。

https://blog.csdn.net/weixin_43404314

即可提交flag.

[AFCTF2018]Morse

打开题目，发现是一段摩斯密码，进行解密，得到一串十六进制数，将其进行字符转换即可得到flag.

[BJDCTF2020]signin

打开题目，内容是16进制数，于是进行字符转换，得到flag.

[AFCTF2018]Single

文件中有密文，还有.c文件，查看代码，发现数组arr是打乱顺序的，无法逆向，也不能爆破，于是直接将密文中的内容进行词频分析

<https://quipqiup.com/>

Puzzle:

```
Oeakay jxoqarerexn u omc tmdc qxuebf lxdomru. Er omc ba uxoarvenz fesa wmdzmoa werv uqajemf reoa lxd rmus-bmuay afaoanru (a.z. IJUB eJRL).
JRL zmoau xlrax rxijv xn omnc xrvad muqajru xl enlxdomrexn uajiderc: jdcqrxzmqvc, uraxz, benmdc mnmfcueu, datadua anzanaadenz, oxbefa uajiderc mny xrvadu. Zxyx ramou zanadmffc
vmta urdxnz useffu mny akqadeanja en mff rvaua euiaiu.
Iuimffc, lfmz eu uxoa urdenz xl dmyxo ymrm xd rakra en uxoa lxdomr. Akmoqfa mljrl{Xv_I_lxiny_er_neja_rDc}
```

Clues: For example G=R QVW=THE

mljrl=afctf

dictionary

Solve

```
0 -1.448 Capture the Flag (CTF) is a special kind of information security competitions. There are three common types of CTFs: Jeopardy, Attack-Defence and mixed. Jeopardy-style CTFs has a couple of questions (tasks) in range of categories. For example, Web, Forensic, Crypto, Binary or something else. Team can gain some points for every solved task. More points for more complicated tasks usually. The next task in chain can be opened only after some team solve previous task. Then the game time is over sum of points shows you a CTF winner. Famous example of such CTF is Defcon CTF quals. Well, attack-defence is another interesting kind of competitions. Here every team has own network(or only one host) with vulnerable services. Your team has time for patching your services and developing exploits usually. So, then organizers connects participants of competition and the wargame starts! You should protect own services for defence points and hack opponents for attack points. Historically this is a first type of CTFs, everybody knows about DEF CON CTF - something like a World Cup of all other competitions. Mixed competitions may vary possible formats. It may be something like wargame with special time for task-based elements (e.g. UCSB iCTF). CTF games often touch on many other aspects of information security: cryptography, stego, binary analysis, reverse engineering, mobile security and others. Good teams generally have strong skills and experience in all these issues. Usually, flag is some string of random data or text in some format. Example
afctf{Oh_U_found_it_nice_tRy}
```

https://blog.csdn.net/weixin_43404314

[ACTF新生赛2020]crypto-classic0

打开文件发现需要密码，于是爆破，得到密码为19990306

输入打开文件是一个.c文件，对代码进行稍微的修改

```
#include<stdio.h>

char flag[25] = "Ygvd#mq[lYate[eIghqvakl]";

int main()
{
    int i;
    for(i=0;i<25;i++)
    {
        flag[i] ^= 0x7;
        flag[i] += 3;
        printf("%c",flag[i]);
    }
    return 0;
}
```

https://blog.csdn.net/weixin_43404314

得到flag{my_naive_encrytion}

[MRCTF2020]keyboard

文件是这样的



得到的flag用
MRCTF{xxxxxx}形式上叫
都为小写字母

6
666
22
444
555
33
7
44
666
66
3

https://blog.csdn.net/weixin_43404314

刚看到这个时并没有什么思路，后来百度看了一下解析，思路一下就通了，就是手机中九键英文格，根据提供的信息，有几个数字，就代表着是九键格中每个格子的英文的第几个。最后得到flag:flag{mobilephone}

[MRCTF2020]天干地支+甲子

，有题目可以知道此题目是一个天干地支60轮回，下载文件，提示了：

得到得字符串用MRCTF{}包裹

一天Eki收到了一封来自Sndav的信，但是他有点迷希望您来解决一下

甲戌
甲寅
甲寅
癸卯
己酉
甲寅
辛丑

找出他们对应的数字，如图：

01 甲子	11 甲戌	21 甲申	31 甲午	41 甲辰	51 甲寅
02 乙丑	12 乙亥	22 乙酉	32 乙未	42 乙巳	52 乙卯
03 丙寅	13 丙子	23 丙戌	33 丙申	43 丙午	53 丙辰
04 丁卯	14 丁丑	24 丁亥	34 丁酉	44 丁未	54 丁巳
05 戊辰	15 戊寅	25 戊子	35 戊戌	45 戊申	55 戊午
06 己巳	16 己卯	26 己丑	36 己亥	46 己酉	56 己未
07 庚午	17 庚辰	27 庚寅	37 庚子	47 庚戌	57 庚申
08 辛未	18 辛巳	28 辛卯	38 辛丑	48 辛亥	58 辛酉
09 壬申	19 壬午	29 壬辰	39 壬寅	49 壬子	59 壬戌
10 癸酉	20 癸未	30 癸巳	40 癸卯	50 癸丑	60 癸亥

写代码：

```
a=[11,51,51,40,46,51,38] sum='' for i in a:
sum+=chr(i+60) print(sum)
```

得到flag

[WUSTCTF2020]佛说：只能四天

打开题目有几个文件，看了一下，没有丝毫思路，只有去看看大佬的Writeup,突然茅塞顿开，

先要对文件的题目进行新约佛论禅解码 [新约佛论禅](#)



然后再进行社会主义核心价值观编码 [核心价值观](#)

核心价值观编码

社会主义核心价值观：富强、民主、文明、和谐；自由、平等、公正、法治；爱国、敬业、诚信、友善

RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3S040S0WG3LynarAVGRZSJRAEYEZ_ooe_doyouknowfence

编码

解码

平等文明自由友善公正自由诚信富强自由自由平等民主平等自由自由友善敬业平等公正平等富强平等自由平等民主和谐公正自由诚信平等和谐公正自由法治平等法治法治法治和谐和谐平等自由和谐自由自由和谐公正自由敬业自由文明和谐平等自由文明和谐平等和谐文明自由和谐自由和谐平等和谐法治公正诚信平等公正诚信民主自由和谐公正民主平等平等平等平等自由和谐和谐和谐平等和谐自由诚信平等和谐自由自由友善敬业平等和谐自由友善敬业平等法治自由法治和谐和谐自由友善公正法治敬业公正友善爱国公正民主法治文明自由民主平等公正自由法治平等文明平等友善自由平等和谐自由友善自由平等文明自由民主自由平等平等敬业自由平等平等诚信富强平等友善敬业公正诚信平等公正友善敬业公正平等平等诚信平等公正自由公正诚信平等法治敬业公正诚信平等法治平等公正友善平等公正诚信自由公正友善敬业法治法治公正公正公正平等公正诚信自由公正和谐公正平等

工具介绍

核心价值观编码 (Core Values Encoder)，经爱党爱国青年sym同意移植本站，旨在通过编程学习党的十八大提出的“社会主义核心价值观”。

联系作者：[github地址](#)

https://blog.csdn.net/weixin_43404314

查看解码的内容，发现最后有一段提示doyouknowfence，查阅了一下，发现是栅栏密码，于是在进行栅栏解密

[BJDCTF 2nd]rsa1

在虚拟机上nc 题目提供的信息

打开后看见题目提供了两个公式

p^*p+q^*q =数值和 $p-q$ =数值 以及 c 和 e

很显然是一个求m得题。首先我们先解出q和p，解出后后面的步骤就简单了。写了求解代码

```

from z3 import * s = Solver() p = Int('p') q = Int('q')
s.add(p*p+q*q==14577767896439231717951214151751334895523
10581059610082098848446975246360741820759182392279195852
08644108611626846193547420096576371805786622869202584453
95189879163161424078110556497247976881839121608594927590
45013451067791630397174120040562740522084419060718444311
98359257996810241446926750473781120669405938) s.add(p-
q==10791789672039627266125263390803996393204014458697550
29828581897555189226591977086789123235078229286549292580
077134182794605403808650547216021817752827174)
s.add(p>0) if s.check() == sat: print s.model() import
libnum import gmpy2 from Crypto.Util.number import * q =
79808378924973129692729824659358893488757671984956548795
54374645126472973151566117482365015476194443329466111649
547173320327698324242697674006938644862993 p =
90600168597012756958855088050162889881961686443654099093
8295654268166219974354320427148825055442372987858691726
681356114933102132893244890028756397690167 c =
69859658614948079531941815132551219020540784247532447081
09589557027094278911919053771192943639345960952388130163
4267221446548074633607624231798135937071294443968074947
97448168112367643242512197583181892589997384422507068688
34395865427340554307275322906672556672110471407205750280
1392452853242422804839755300 e = 10058063 phin = (p-1)*
(q-1) n=p*q d=gmpy2.invert(e,phin) # print d m =
pow(c,d,n) print long_to_bytes(m)

```

flag[b6d27a35-7ef8-49ef-9a4a-c5f365b34976]

有关于python中z3的详细介绍，建议去z3了解，讲得很通俗易懂

[MRCTF2020]vigenere

是一个维吉尼亚题，下载文件后打开，发现是一堆乱序的单词，还提供了一个py文件，看了py文件，发现写不出代码，只得在网上找在线解密工具，功夫不负有心人，也是找到了vigenere解密，在不知道密钥的前提下，猜测一下密钥的大概范围，解密，在文章的最后也得到了flag



[GWCTF 2019]BabyRSA

文件提供的内容为提个txt文本文件，内容如下：

```

N=636585149594574746909030160182690866229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487
m1=9000974341452243216986938028371257528604943208941176518717463554774967878152694586469377652961131656594987260127122886704588843739714198427509292876586402662196866469569298
m2=487443985757405173426628188375657117604235507936967522993257972108872283698305238454465723214226871414276788912058186197039821242912736742824080627680971802511206914394672159

```

还提供了一个加密.py脚本；因此我们可以根据题目提供的信息分析其解题思路。

```
import hashlib
import sympy
from Crypto.Util.number import *
flag = 'GWH{T*****}'
secret = '*****'
assert(len(flag) == 38)
half = len(flag) / 2
flag1 = flag[:half]
flag2 = flag[half:]
secret_num = getPrime(1024) * bytes_to_long(secret)
p = sympy.nextprime(secret_num)
q = sympy.nextprime(p)
N = p * q
e = 0x10001
F1 = bytes_to_long(flag1)
F2 = bytes_to_long(flag2)
c1 = F1 + F2
c2 = pow(F1, 3) + pow(F2, 3)
assert(c2 < N)
m1 = pow(c1, e, N)
m2 = pow(c2, e, N)
output = open('secret', 'w')
output.write('N=' + str(N) + '\n')
output.write('m1=' + str(m1) + '\n')
output.write('m2=' + str(m2) + '\n')
output.close()
```

代码加密过程说明了flag的字符长度为38位，将flag分为左右两部分，脚本提到了nextprime，指的是下一个质数，说明p,q是两个相邻的质数，那么可以对N进行开根号，有 $q = \text{sympy.nextprime}(p)$ ，可知q是p的下一个质数。再可根据两个表达式求解c1,c2.对c1,c2进行加密即可得m1,m2。代码清楚了，就很清楚思路了，要求出F1,F2首先就得先求出c1,c2,而求解c1,c2的关键是求解p,q.前面已经介绍了该如何求得q,p后面的步骤就可想而知了。

```

import gmpy2 import sympy e = 0x10001
N=gmpy2.mpz(63658514959457474690903016018269086622290925
6464847291783000651837227921337237899651287943597732709
44384034858925295744880727101606841413640006527614873110
65141015589377654873782315294379788472913014975827912743
00447392540004266109228345730949570825895394456108282794
28814524313491262061930512829074466232633130599104490893
5720939438327403018096308475415925489212002882243278920
86509499376383034294564688891001926138590737529238124542
12239908948930178355331390933536771065791817643978763045
03083371232616288381063812002937833709293866217411974768
78994846036283440794935566014224984053607319581627192961
60584042671057160241284852522913676264596201906163)
m1=90009974341452243216986938028371257528604943208941176
5187174635547749678781526945864693776529611316565949872
60127122886704588843739714198427509292876586402662196866
46956929872115782173093979742958745121671928568709468526
09871592718982960049728311805164110730512885269703205336
81151812160696266061655034651257252048755787012377892929
6621182400276148181527666236869005129138862782476859103
08672609186049761488328294995502322241433324319326856478
16216998704125578224043812138040266858312214307282907555
97819259339616650158674713248841654338515199405532003173
73252045781390117026471308510707700147808334133900206987
0585378257051150217511755761491021553239
m2=48744398575740517342662818837565711760423550793696752
29932579721088722836983052384544657232142268714142767889
12058186197039821242912736742824080627680971802511206914
39467215924020691073585065199931610001469106729570813863
93632035962446939955627802866371163947382507741297590210
80197323724805414668042318806010652814405078769738548913
67546618155100552706530951536495061013720639325714835765
96666870916627498485602254538263622717042926928475963395
33229088038820532086109421158575841077601268713175097874
08353624900601894878941323878392284563349402360886525607
19628565812298900438969390256136005642833913293314521990
62858930374565991634191495137939574539546
n2=gmpy2.iroot(N,2)[0] p=sympy.nextprime(n2) q=N//p phi=
(p-1)*(q-1) d=gmpy2.invert(e,phi) c1=pow(m1,d,N)
c2=pow(m2,d,N) a=3*c1 b=-3*pow(c1,2) c=pow(c1,3)-c2
delta=gmpy2.iroot(pow(b,2)-4*a*c,2)[0] F2=(-
b+delta)//(2*a) F1=c1-F2 print(hex(F2)
[2:].decode('hex')+hex(F1)[2:].decode('hex'))

```

得flag: GWHT{f709e0e2cfe7e530ca8972959a1033b2}

[WUSTCTF2020]babysrsa

打开题目，给我们提供的内容为c,n,e

可行而知这是要求m，即解密

要求m首先得求出d，而要求得d得先求出p和q，由于题目提供的n不是很大，可以用大数分解(<http://www.factordb.com/>)直接求出p和q

脚本如下：

```

import gmpy2 from Crypto.Util.number import
long_to_bytes c =
28767758880940662779934612526152562406674613203406706867
456395986985664083182 e = 65537
p=189239861511125143212536989589123569301 q=
386123125371923651191219869811293586459
d=gmpy2.invert(e,(p-1)*(q-1))
n=730698867716256428074357836610140626042647684817351458
73508846925735521695159 m=pow(c,d,n) print
long_to_bytes(m)

```

[BJDCTF2020]easysrsa

提供的信息为一个脚本，如下：

```
from Crypto.Util.number import getPrime,bytes_to_long
from sympy import Derivative
from fractions import Fraction
from secret import flag
p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
m=bytes_to_long(flag)
c=pow(m,e,n)
print(c,z,n)
"""
output:
792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380351099487742013553726854941065265447962085869132411036718202564878840
321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764345875193333001424730103446948038851685575488012024959332262154377633:
153107451613368954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012490941618672076865376070474479687087589909501363809247473590525705495:
"""
```

思路：

根据代码可以知道几个已知信息

$$n=p*q$$

$$z=p^2+q^2$$

难点就是要把z的这个公式推导出来，不熟悉的，百度搜索查找找到

Fraction(a,b)=a/b;

Derivative(f(x),x):这是用来求导的

同时又知道arctan和arth的导函数分别为 $(1/(1+x^2))$ 、 $(1/(1-x^2))$

$$\text{反正弦函数的导数公式: } (\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$$

$$\text{反余弦函数的导数公式: } (\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$$

$$\text{反正切函数的导数公式: } (\arctan x)' = \frac{1}{1+x^2}$$

$$\text{反余切函数的导数公式: } (\operatorname{arc} \cot x)' = -\frac{1}{1+x^2}$$

https://blog.csdn.net/weixin_43404314

所以，很容易推导出 $z=p^2+q^2$

然后我们根据 $n=p*q$

联系上面两个公式，推导出p和q

首先设两个参数a,b

$$\text{其中 } a=p^2+q^2+2*p*q=z+2*n$$

$$b=p^2+q^2-2*p*q=z-2*n$$

然后将a,b开平方，

$$p=(\sqrt{a}+\sqrt{b})//2$$

$$q=(\sqrt{b}-\sqrt{a})//2$$

后面就很容易了

$$\text{phi}=(p-1)*(q-1)$$

已知了e

我们可以求得d

$$ed=1 \pmod{\text{phi}}$$

$$m=\text{pow}(c,d,n)$$

根据以上思路可写出py脚本：


```
from Crypto.Util.number import *
from sympy import Derivative
from fractions import Fraction
from gmpy2 import *

c=mpz(79225478668577614598074915026542162830127761777895115493506729581018102813484022840983101477965494306892538035109948774201355372685494106526544796208586913241103671820256
z=mpz(32115748677623209667471622872185275070257924766015020072805267359839059393284316595882933372289732127274076434587519333300142473010344694803885168557548801202495933226215
n=mpz(15310745161336895413406690009324766200789179248896951942047235448901612351128459309145825547569298479821101249094161867207686537607047447968708758990950136380924747359052!
pqplus=iroot(z+2*n,2)[0]
pqminus=iroot(z-2*n,2)[0]
p=(pqminus+pqplus)//2
q=(pqplus-pqminus)//2
# p=getPrime(1024)
# q=getPrime(1024)
e=65537
phi=(p-1)*(q-1)
d=invert(e,phi)
m=pow(c,d,n)
#z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
flag=long_to_bytes(m)
print flag
```

持续更新中