

ctf-crypto1

原创

零零柒c 于 2021-09-26 23:10:50 发布 1448 收藏 1

分类专栏: [网络安全](#) 文章标签: [python](#) [web安全](#) [安全](#)

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。

本文链接: https://blog.csdn.net/qq_45448359/article/details/120497870

版权



[网络安全](#) 专栏收录该内容

6 篇文章 0 订阅

订阅专栏

前言

一次巧合的情况下点开了一个ctf比赛 一直没玩过ctf 第一次 0基础分析 签到题 crypto的

题目:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Util.number import *
import random
flag=b'flag{*****}'
n = 2 ** 256
flaglong=bytes_to_long(flag)
m = random.randint(2, n-1) | 1
c = pow(m, flaglong, n)
print('m = ' + str(m))
print('c = ' + str(c))

# m = 73964803637492582853353338913523546944627084372081477892312545091623069227301
# c = 21572244511100216966799370397791432119463715616349800194229377843045443048821
```

开始啥都不会

先通过找师傅帮忙 给出了解密的脚本

```
m = 73964803637492582853353338913523546944627084372081477892312545091623069227301
c = 21572244511100216966799370397791432119463715616349800194229377843045443048821
n = 2 ** 256
import sympy
flag=sympy.discrete_log(n,c,m)
print(flag)
import binascii
print(hex(flag))
print(hex(flag)[2:])
print(binascii.unhexlify(hex(flag)[2:]))#将答案的十六进制转出来就行
print(binascii.hexlify(b'flag{DASCTF_zjut}'))
```

运行结果

```
flag.py C:\Users\ASUS\AppData\Local\Temp\flag.p 1 m = 73964803637492582853353338913523546944627084372081477892312545091623069227301
2 c = 21572244511100216966799370397791432119463715616349800194229377843045443048821
3 n = 2 ** 256
4 import sympy
5 flag=sympy.discrete_log(n,c,m)
6 print(flag)
7 import binascii
8 print(hex(flag))
9 print(hex(flag)[2:])
10 print(binascii.unhexlify(hex(flag)[2:]))#将答案的十六进制转出来就行
11 print(binascii.hexlify(b'flag{DASCTF_zjut}'))
```

```
Run: flag x
C:\Users\ASUS\AppData\Local\Programs\Python\Python37\python.exe "D:/WeChat Files/wxid_rp15fs136mgp22/FileStorage/File/2021-09/flag.py"
34852863801130149185238904762083023615101
0x666c61677b4441534354465f7a6a75747d
666c61677b4441534354465f7a6a75747d
b'flag{DASCTF_zjut}'
b'666c61677b4441534354465f7a6a75747d'
```

CSDN @零零柒c

先分析题目

主要的地方在

$c = \text{pow}(m, \text{flaglong}, n)$ 这一段代码

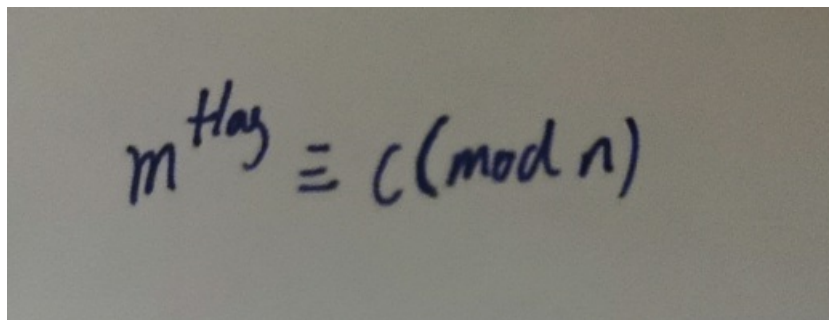
这段代码可以看出只有 flaglong 是我们不知道的

所以需要反解出 flaglong

$c = \text{pow}(x, y, z)$: 这个是表示 x 的 y 次幂后除以 z 的余数。

所以要求 y 就相当于求 flaglong

根据师傅的帮助 这道题就可以转化为


$$m^{\text{flag}} \equiv c \pmod{n}$$

求满足 m 的 flag 次方等于 c 对模 n 取同余

“ \equiv ”是数论中表示同余的符号

discrete_log 这个函数就是用来求这种类型的

$\text{discrete_log}(x, y, z)$, x 是模数, y 是余数, z 是底数

$\text{sympy.discrete_log}(n, c, m)$ 就相对于上面的 $c = \text{pow}(m, \text{flaglong}, n)$

c 是余数 m 是底数 n 是模数

`**flag=sympy.discrete_log(n,c,m)**`这样解出来的flag就是flaglong 是十进制数 因为开始是通过bytes_to_long函数转化的 所以要转化为字符

十进制转化为字符 要先转化为16进制 然后转化为字符
所以用hex先转化为16进制 然后截取掉 进制标志位
然后在通过binascii.unhexlify()函数转化为字符

也可通过在线的进制转化

2进制 8进制 10进制 16进制 32进制 36进制 58进制 62进制 | 更多:

转换

进制	结果	解释
2	1100110011011000110000101100111011110110100010	
8	631543026357321040523206521062767515235272175	
10	34852863801130149185238904762083023615101	
16	666c61677b4441534354465f7a6a75747d	
26	ijtmzfljqzpomjxsvhloviydvayv	小写字母
32	SKCC5KQPH21AD1N8HJZF9N7AX3X	不包含 ILOU 字符
36	174b5qlzhzowz62jnh59dxn5p	数字 + 小写字母

CSDN@零零柒c

16进制转换文本 / 文本转16进制

666c61677b4441534354465f7a6a75747d

字符串转16进制 >>

16进制转字符串 >>

结果互换

全部清空

flag{DASCTF_zjut}

CSDN @零零柒c

最后也可以通过Crypto 库的另一个函数直接转化flag字符

```
Type: help, copyright, credits, license, for more
>>> import sympy 请输入要进行 Base64 编码或解码的字符
>>> from Crypto.Util.number import *
>>> n = 2 ** 256  hjZX1pcnVmdmRzZWZ/bGlg==
>>> m = 7396480363749258285335333891352354694462708437208
>>> c = 2157224451110021696679937039779143211946371561634
>>> flag=sympy.discrete_log(n, c, m)
>>>
>>> print(long_to_bytes(flag))
b'flag{DASCTF_zjut}'
>>>
```

CSDN @零零柒c

但是这里我本地不知道什么原因用不起Crypto 就没有在本地复现了

成功解出flag