cryptohack GENERAL 部分wp by crumbling

目录

GENERAL

ENCODING

ASCII

ascii编码

chr()用于将ASCII转为字符，ord()功能相反

```
a=[99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110,
116, 52, 98, 108, 51, 125]
b=''
for i in a:
    b+=chr(i)
print(b)
#crypto{ASCII_pr1nt4bl3}
```

Hex

十六进制字符串与字节之间的转换

bytes.fromhex(a)用于将十六进制字符串转为字节,a.hex()功能相反

```
a='63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d'
b=bytes.fromhex(a)
print(b)
```

Base64

base64编码

*需要先对16进制字符串解码

```
import base64
a='72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf'
b=bytes.fromhex(a)
c=base64.b64encode(b)
print(c)
```

附:

魔改表base64

注：填充用的"="有的时候需要替换，有的时候不需要。

```
import base64
str1 = "j2rXjx8yjd=YRZWyTIuwRdbyQdbqR3R9iZmsScutj2iqj3/tidj1jd=D"
string1 = "GHI3KLMNJOPQRSTUb=cdefghijklmnopWXYZ/12+406789VaqrstuvwxyzABCDEF5"
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="
print(base64.b64decode(str1.translate(str.maketrans(string1,string2))))
```

Bytes and Big Integers

字节与大整数之间的转换。

bytes_to_long()字节转大整数，long_to_bytes()功能相反

```
from Crypto.Util.number import *
a=11515195063862318899931685488813747395775516287289682636499965282714637259206269
c=long_to_bytes(a)
print(c)
```

Encoding Challenge

100个题目，手搓显然不现实，可以用pwntool连接靶机然后解题（附件有example，还是比较好理解的），配置了linux下的python环境，然后简单研究了一下几个函数的作用。

关于解密的题目内容，只是在前面四题的基础上增加了个rot13（凯撒13位），用maketrans函数建立映射表，然后str.translate替换。

```
from pwn import *
import json

r = remote('socket.cryptohack.org', 13377, level = 'debug')

def json_recv():
    line = r.recvline()
    return json.loads(line.decode())

def json_send(hsh):
    request = json.dumps(hsh).encode()
    r.sendline(request)

import base64
from Crypto.Util.number import *
import string

def decryptHtoB(enc):
    return (bytes.fromhex(enc)).decode()

def decryptbase64(enc):
    enc=enc.encode()
    ans=base64.b64decode(enc)
    return ans.decode()

def decryptItoB(enc):
    enc=int(enc,16)
    return (long_to_bytes(enc)).decode()

def decryptrot13(enc):
    string1= 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
    string2 = 'NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm'
    return enc.translate(str.maketrans(string1,string2))

def decryptutf(enc):
    return "".join(chr(i) for i in enc)

def solve():
    received = json_recv()
    enc=received["encoded"]
    enctype=received["type"]
    if enctype=="base64":
        dec=decryptbase64(enc)
    if enctype=="hex":
        dec=decryptHtoB(enc)
    if enctype=="rot13":
        dec=decryptrot13(enc)
    if enctype=="bigint":
        dec=decryptItoB(enc)
    if enctype=="utf-8":
        dec=decryptutf(enc)
    to_send = {"decoded": dec}
```

```
    json_send(to_send)
    return None

for _ in range(100):
    solve()
```

XOR

XOR Starter

简单直接的异或

```
a=b'label'
def solution(a):
     return b''.join((i^13).to_bytes(1,'big') for i in a)
print(solution(a))
```

XOR Properties

附件中有介绍异或的相关性质：

Commutative: A ⊕ B = B ⊕ A

Associative: A ⊕ (B ⊕ C) = (A ⊕ B) ⊕ C

Identity: A ⊕ 0 = A

Self-Inverse: A ⊕ A = 0

这里主要利用：A^A^B=0^B=B（也是比较常用的）

```
key1=bytes.fromhex('a6c8b6733c9b22de7bc0253266a3867df55acde8635e19c73313')
t2=bytes.fromhex('37dcb292030faa90d07eec17e3b1c6d8daf94c35d4c9191a5e1e')
t3=bytes.fromhex('c1545756687e7573db23aa1c3452a098b71a7fbf0fdddde5fc1')
t4=bytes.fromhex('04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf')
key2=b''
key=b''
flag=b''
for i in range(len(t3)):
     key += (t3[i] ^ key1[i]).to_bytes(1, 'big')
for i in range(len(t2)):
     flag +=(t4[i]^key[i]).to_bytes(1,'big')
print(flag)
```

Favourite byte

信息异或了某个长度为1字节的key，枚举一下求解。

```
enc=bytes.fromhex('73626960647f6b206821204f21254f7d694f7624662065622127234f726927756d')
for j in range(128):
   flag=b''
   for key in range(len(key1)):
      flag +=(enc[i]^key).to_bytes(1,'big')
   if b"crypto" in flag:
      print(flag)
```

You either know, XOR you don't

主要利用在A^key=C的情况下有C^key=A，并且已知flag中包含'crypto{}'和全部密文，和上题一样枚举获得key

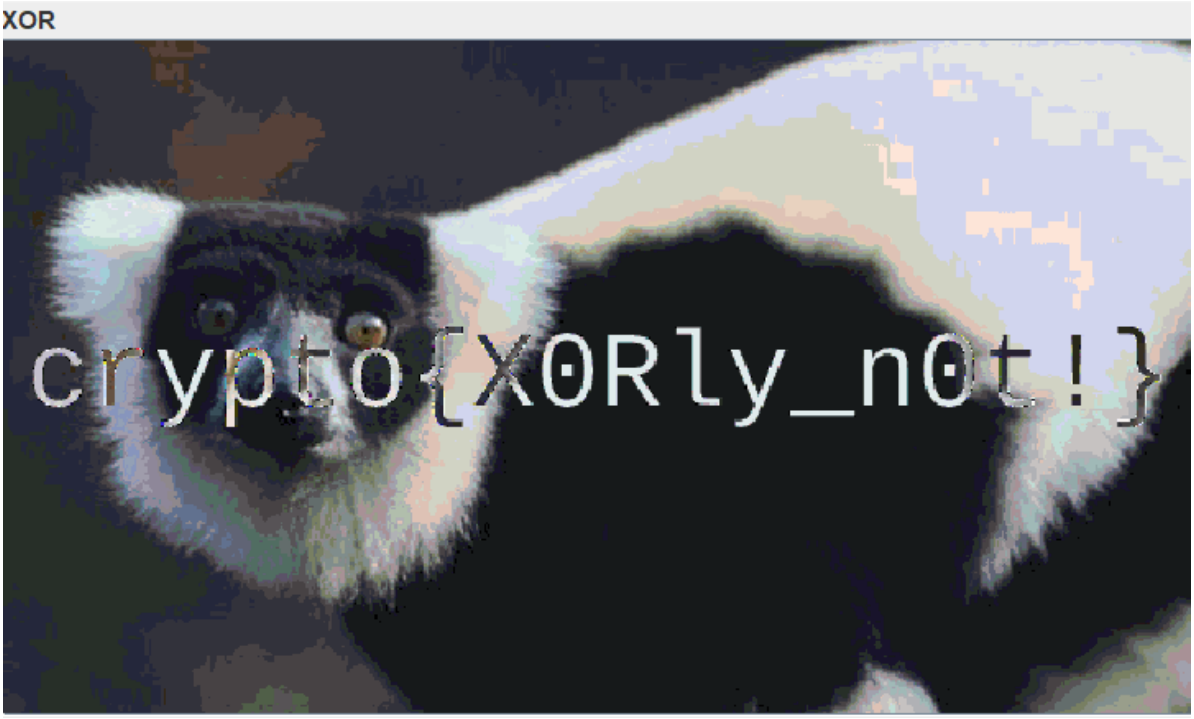
```
enc=bytes.fromhex('0e0b213f26041e480b26217f27342e175d0e070a3c5b103e2526217f27342e175d0e077e263451150104')
key1=b'crypto{}'
key=[]
flag=b""
for j in range(7):
     for i in range(127):
          if enc[j]^i==key1[j]:
               key.append(i)
               break
j=len(enc)-1
for i in range(127):
     if enc[j]^i==key1[-1]:
          key.append(i)
          break

for i in range(len(enc)):
     flag +=(enc[i] ^ key[i % 8]).to_bytes(1,'big')
print(flag)
```

*有一点点猜的部分，不过不是关键。

Lemur XOR

图片的异或，即每个对应像素点进行异或，而Stegsolve有相应的异或功能。(其实是当时用PIL出了点问题采用了stegsolve)



MATHEMATICS

Greatest Common Divisor

gcd，最大公因数。

~~哇，一年前得到超朴素代码~~

```
x = 66528
y = 52920
while(x%y!=0):
    m=x%y
    x=y
    y=m
print(y)
```

可以直接调用math库的函数

```
from math import *
print(gcd(66528,52920))
```

Extended GCD

扩展欧几里得算法。

```
def extended_gcd(a,b):
    if a==0:
        return b,0,1
    else:
        gcd, x, y = extended_gcd(b%a, a)
        return gcd, y-(b // a) * x, x
print(extended_gcd(26513, 32321))
```

Modular Arithmetic 1

求余数

`print(max(8146798528947%17,11%6))`

Modular Arithmetic 2

费马小定理可以直接获得答案

> 如果p是一个质数，而整数a不是p的倍数，则有a^（p-1）≡1（mod p）

当然了解快速幂以及pow函数的使用也很有必要。

Modular Inverting

求乘法逆元

根据提示利用费马小定理：当a与p**互质**时，有a*a^（p-2）≡1（mod p），也就是说a的逆元是a^(p-2)，而该值可以利用快速幂求解。

也可以了解一下gmpy2这个库

```
from gmpy2 import*
print(gmpy2.invert(3,13))
```