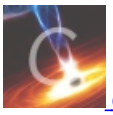
YOU NEED PYTHON writeup

原创

cien_anos 于 2017-04-13 18:09:56 发布 2304 收藏 3

分类专栏： ctf记录 文章标签： ctf

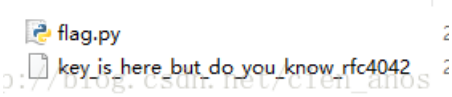
本文链接：https://blog.csdn.net/cien_anos/article/details/70160614



第一次独立做出来的ctf题目，记录一下。以后也会把ctf记录和总结写出来

题目地址：https://dn.jarvisoj.com/challengefiles/%E9%A2%98%E7%9B%AE%EF%BC%9Ayou_need_python.zip.74d515955b9aa607b488a48437591a14

下载附件解压缩后发现有两个文件



先看第一个

```
import marshal, zlib, base64

exec(marshal.loads(zlib.decompress(base64.b64decode('eJxtVP9r21YQvyd/ieWm66Cd03QM1B8C3pggUuzYCSWstHSFQi
```

不妨先运行一下

```
[*] Please input key: 1
[*] Please input flag: 1
[!] Key or flag is wrong, try again:)
```


输入key和flag，暂且不知道key和flag的值是多少。

返回到代码:

exec()函数可用来执行执行储存在字符串或文件中的Python语句（应该可以是.pyo文件）。

marshal.loads()将二进制流反序列化为对象

zlib.decompress()对字符串进行解压缩

base64.b64decode()为base64解码

分析可知，代码中marshal.loads()处理后的字符串是pyo格式的编译程序。利用uncompyle2.uncompyle()将其反编译为py文件，得到程序的源码

```
import hashlib

def sha1(string):
    return hashlib.sha1(string).hexdigest()


def calc(strSHA1):
    r = 0
    for i in strSHA1:
        r += int('0x%s' % i, 16)

    return r


def encrypt(plain, key):
    keySHA1 = sha1(key)
    intSHA1 = calc(keySHA1)
    r = []
    for i in range(len(plain)):
        r.append(ord(plain[i]) + int('0x%s' % keySHA1[i % 40], 16) - intSHA1)
        intSHA1 = calc(sha1(plain[:i + 1])[:20] + sha1(str(intSHA1))[:20])

    return ''.join(map(lambda x: str(x), r))


if __name__ == '__main__':
    key = raw_input('[*] Please input key:')
    plain = raw_input('[*] Please input flag:')
    encryptText = encrypt(plain, key)
    cipherText = '-185-147-211-221-164-217-188-169-205-174-211-225-191-234-148-199-198-253-175-157-222-
    if encryptText == cipherText:
        print '[>] Congratulations! Flag is: %s' % plain
        exit()
    else:
        print '[!] Key or flag is wrong, try again:)'
        exit()
```

分析代码可知，程序接受两个参数key和plain，encrypt(key,plain)的返回值与代码给出的字符串相等的话则说明找到了正确的flag。

key参数可从题目给的第二个文件中找到。文件名为key_is_here_but_do_you_know_rfc4042，rfc4042中定义了utf-9编码。github上找到了utf-9的编解码库，通过解码，获取了文件中的内容

```
a = open('C:\\Users\\lenovo\\Desktop\\txt_a', 'r')
b = a.read()
print utf9.utf9decode(b)
```

```
输出结果为：#输出为：  _____*((__//__+___+______-____%____)**((___%(___-_))+________+(___%___+_____+______
```

看样子是一个很长的表达式，很轻松的想到“_”的个数表示数字，于是通过以下代码对其进行解析

```
a = open('C:\\Users\\lenovo\\Desktop\\txt_a', 'r')
b = a.read()
c = utf9.utf9decode(b)

num = 0
data = ''
for i in c:
    if i == '_':
        num += 1
    else:
        if num != 0:
            data += repr(num)
            num = 0
        data += i
data += repr(num)
print eval(data)
```

```
结果为52870021310743315l3
```

第一次做的时候我直接把这个当作key进行暴力破解，发现得出的结果是一堆乱码。于是又返回这里，想到这个可能是一个字符串的ascii编码的10进制表示，尝试解码

```
str_data = hex(d)[2:-1]
key = ''

for i in range(len(str_data)/2):
    key += chr(eval('0x'+str_data[2*i:2*i+2]))

print key
```

得到key为l_4m-k3y

最后，直接利用反编译的代码，爆破出flag值

```python
import hashlib
def sha1(string):
    return hashlib.sha1(string).hexdigest()


def calc(strSHA1):
    r = 0
    for i in strSHA1:
        r += int('0x%s' % i, 16)

    return r


def encrypt(plain, key):
    keySHA1 = sha1(key)
    intSHA1 = calc(keySHA1)
    r = []
    for i in range(len(plain)):
        r.append(ord(plain[i]) + int('0x%s' % keySHA1[i % 40], 16) - intSHA1)
        intSHA1 = calc(sha1(plain[:i + 1])[:20] + sha1(str(intSHA1))[:20])

    return ''.join(map(lambda x: str(x), r))

string = '-185-147-211-221-164-217-188-169-205-174-211-225-191-234-148-199-198-253-175-157-222-135-240-
flag = ''
flag_list = []

for i in range(len(string)/4):
    for j in range(200):
        if encrypt(flag + chr(j), 'I_4m-k3y') == string[0:4*(i+1)]:
            flag += chr(j)
            flag_list.append(j)
            break
print flag
```

结果为flag{Lif3_i5_5h0r7_U_n33d_Py7h0n}