网络与信息安全领域专项赛 writeup

原创

逃课的小学生 于 2019-08-19 13:56:04 发布 1029 收藏 1
分类专栏： crypto wechall


crypto 同时被 2 个专栏收录
20 篇文章 1 订阅
订阅专栏


wechall
2 篇文章 0 订阅
订阅专栏

crypto sm4：

题目提示这是一个sm4加密的题目，给出了key和密文，于是在网上找了一个sm4加密的代码(https://blog.csdn.net/songdawww/article/details/79112548)，稍加修改即可得到正确答案(将源代码主函数的ecb解密留下，将密钥和密文放入即可)

```
key_data = [13, 204, 99, 177, 254, 41, 198, 163, 201, 226, 56, 214, 192, 194, 98, 104]
    iv_data = [0x5a]*16
en_data=[46, 48, 220, 156, 184, 218, 57, 13, 246, 91, 1, 63, 60, 67, 105, 64, 149, 240, 217, 77, 107, 49, 2
sm4_d.sm4_set_key(key_data, DECRYPT)
de_data = sm4_d.sm4_crypt_ecb(en_data)
de_data_str = "".join([chr(x) for x in de_data])
print de_data_str
```

```
SM4:   flag{1caa96be-4266-4a8e-bd2c-ece977495497}
```

crypto dp：

题目中我们可以拿到e,n,dp,c,这里dp=e^(-1)mod(p-1)。于是我们有m^(e*dp) mod p=m，即m^(e*dp)=k*p+m，于是我们有(m^(e*dp)) mod n=(k*p+m) mod n= k*p+m，这时我们有gcd(((m^(e*dp)) mod n)-m,n)=gcd(k*p,n)=p，即可正常将n分解

```
e=65537
n=963757146665289974184814265445141340580197683432866741850921714950323851383087098535391831463316027758059
dp=813394057049025176760221889085475436896278294537998655500914948427254395705713100713377290385165255391580
c=597137277657470690515854669815717809870618759720498166203631053436957591577695096289379080927483346254567
```

代码如下

```
import binascii
import gmpy2
e=65537
n=963757146665289974184814265445141340580197683432866741850921714950323851383087098535391831463316027758059
dp=813394057049025176760221889085475436896278294537998655500914948427254395705713100713377290385165255391158
c=597137277657470690515854669815717809870618759720498166203631053436957591577695096289379080927483346254567

# Random R
r = 2
# n = pq
p = gmpy2.gcd(n, pow(r, (e*dp), n) - r)
q = gmpy2.div(n, p)
print("p: %d" % p)
print("q: %d" % q)
# calculate d
phi = (p-1) * (q-1)
d = gmpy2.invert(e, phi)
print("phi: %d" % phi)
print("d: %d" % d)
# Calculate message
m = int(pow(c, d, n))
print("m: %d" % m)
# Convert int message to string
mHex = format(m, 'x')
print(mHex)
message = binascii.unhexlify(mHex).decode("utf-8")
print(message)
```

re src_leak:

我们查看源码发现我们需要符合func3< func2<x1>为1且_func1<x1>::result为文件中output中的数值。我们会发现_func1<i>::result为1的时候，i=1，2，3，_func1<i>::result为2的时候，i=4，5，6，7，8。这时我们会发现_func1<i>::result为k的时候，i有2k+1个，且顺序从1往后排。所以我们可以编程计算出_func1<i>::result为k的i的范围

```
sum=0;
for i in xrange(1,963):
    sum=sum+(2*i+1)

print sum
```

我们将这范围的数字代入 func3< func2<x1>，查找其中最后结果为1的最小值即为flag的一部分

接下来我们需要知道在0<i<10000中，有多少fun4<i> ::value==1。我们发现当i为素数时，fun4<i> ::value==1。这时我们只需找到0<i<10000中有多少素数即可。

re flat

题目提示uuid，所以我们输入flag{uuid},即可到uuid的检查，逐步执行uuid检查程序，我们发现在程序中做了如下的更改，他将原先的uuid中的数字“0123456789”转为“ABCDEFGHIJ",将字母abcdef转为数字”123456“，然后我们可以在执行的时候观察栈我们会发现我们的uuid加密后得到的密文和最后要检验的密文，所以我们只需要将密文解密即可得到答案


```
dict1={"A":"0","B":"1","C":"2","D":"3","E":"4","F":"5","G":"6","H":"7","I":"8","J":"9","1":"a","2":"b","3":
cipher="J2261C63-3I2I-EGE4-IBCC-IE41A5I5F4HB"
message=""
for i in cipher:
    if dict1.has_key(i):
        message=message+dict1[i]
    else:
        message=message+i

print message
```