

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](#)

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/zhang14916/article/details/99713802>

版权



[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 $dp=e^{-1} \pmod{p-1}$ 。于是我们有 $m^{(e*dp)} \pmod p = m$, 即 $m^{(e*dp)} = k*p + m$, 于是我们有 $(m^{(e*dp)}) \pmod n = (k*p + m) \pmod n = k*p + m$, 这时我们有 $\gcd(((m^{(e*dp)}) \pmod n) - m, n) = \gcd(k*p, n) = p$, 即可正常将n分解

```
e=65537
n=963757146665289974184814265445141340580197683432866741850921714950323851383087098535391831463316027758059:
dp=81339405704902517676022188908547543689627829453799865550091494842725439570571310071337729038516525539158:
c=597137277657470690515854669815717809870618759720498166203631053436957591577695096289379080927483346254567:
```

代码如下

```

import binascii
import gmpy2
e=65537
n=963757146665289974184814265445141340580197683432866741850921714950323851383087098535391831463316027758059
dp=81339405704902517676022188908547543689627829453799865550091494842725439570571310071337729038516525539158
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“ABCDEFGHJI”,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

```

gdb-peda$ x /20sw 0x7fffffffde50
0x7fffffffde50: U"HBIGHGE5-3CEA-BB5J-2I65-AAACJ4JAD\x40073e"
0x7fffffffdee0: U"\xffffe1c0翻\xffffe070"
0x7fffffffdef0: U""
0x7fffffffdef4: U"\x126f478\x2febe0ac"
0x7fffffffdf00: U"\x41f865c6"
0x7fffffffdf08: U"\x61278e\x12e5a95e"
0x7fffffffdf14: U"\x6a28d80b\xffffdf90翻\xffffe030翻\x1ffe190-\xffffe0e0翻\x401342"
0x7fffffffdf40: U""
0x7fffffffdf44: U""
0x7fffffffdf48: U"\xffffe070翻"
0x7fffffffdf54: U""
0x7fffffffdf58: U"\x18db46cc\x54762a38\x5eebab89"
0x7fffffffdf68: U"\xc1f1738\x128d0181\x2de607e4\x3c57f144\x45bda74a\xcdc4b0d0\xffffe070翻@25\xcdc4b0d0J2261C63-3I2I-EGE4-IBCC-IE41A5I5F4HB"
0x7fffffffdf80: U""
0x7fffffffdf84: U"*)\x36383137\x65343637\x3432632d\x31312d30\x622d3965\x2d656638\x63303030\x39643932\x61363330"
0x7fffffffdf88: U""
0x7fffffffdf8c: U""
0x7fffffffdf90: U""
0x7fffffffdf94: U""
0x7fffffffdf98: U""

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
已下载 59.3 kB, 耗时 4秒 (13.4 kB/s)
正在选中未选择的软件包 libosspp-uuid16:amd64.
(正在读取数据库 ... 系统当前共安装有 411833 个文件和目录。)
准备解压 .../libosspp-uuid16_1.6.2-1.5+b7_amd64.deb ...
正在解压 libosspp-uuid16:amd64 (1.6.2-1.5+b7) ...
正在选中未选择的软件包 uuid.
准备解压 .../uuid_1.6.2-1.5+b7_amd64.deb ...
正在解压 uuid (1.6.2-1.5+b7) ...
正在设置 libosspp-uuid16:amd64 (1.6.2-1.5+b7) ...
正在设置 uuid (1.6.2-1.5+b7) ...
正在处理用于 mag-db (2.8.5-2) 的触发器
正在处理用于 uuid (1.6.2-1.5+b7) 的触发器
root@kali:~# apt install uuid
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
uuid 已经安装新版本 (1.6.2-1.5+b7)

```

<https://blog.csdn.net/zhang14916>

```

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

```



[创作打卡挑战赛](#) >
[赢取流量/现金/CSDN周边激励大奖](#)