攻防世界crypto高手题之best_rsa

原创

沐一 · 林 于 2021-09-13 21:52:47 发布 180 收藏 1

分类专栏： CTF 密码学 文章标签： ctf

本文链接：https://blog.csdn.net/xiao__1bai/article/details/120274386

CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

订阅专栏

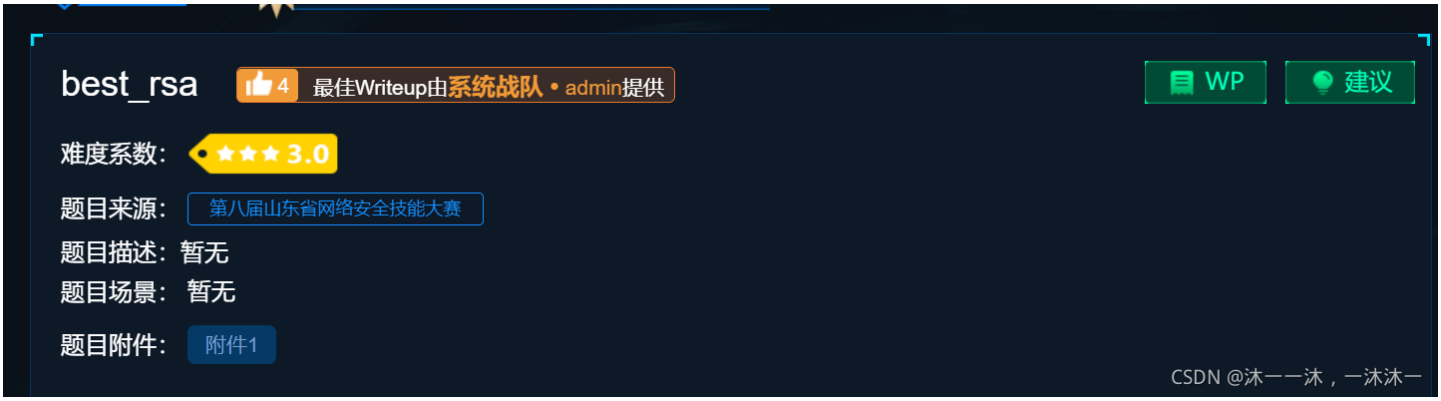
密码学

51 篇文章 1 订阅

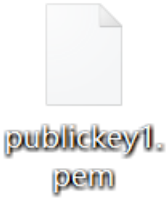
订阅专栏

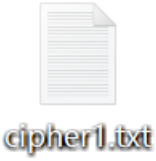
攻防世界crypto高手题之best_rsa

继续开启全栈梦想之逆向之旅~
这题是攻防世界crypto高手题的best_rsa



.
.
下载题目，是一个明文和密钥的4个附件：





.
.
其实一开始我并不知道`RSA`的明文和密钥是怎么生成的，`CTF-RSA-tool`中应对的也只有一对明文密钥文件而已,这里两对的话就只能查资料弄懂原理了再做了。
.
(这里积累第一个经验)
从别人的博客`https://www.cnblogs.com/vict0r/p/13542398.html`中找到了从密钥文件和明文文件读取出对应数字的方法，提取`n、e`都是用`Crypto.PublicKey.RSA`模块，再抽取对应的`n,c属性`的。
提取加密`密文c`，则是直接二进制读取文件后用`Crypto.Util.number`模块的`bytes_to_long`函数转二进制流为数字的。

.

附上我修改后的脚本，写脚本的时候出了点小插曲，就是`print("n1 =",n1)`使用逗号,来同时输出字符和变量的，而不是用加号`print("n1 ="+n1)`或点号`print("n1 =".n1)`。。。可能只有我会这么傻吧~

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *
f1 = open("publickey1.pem","rb").read()
f2 = open("publickey2.pem","rb").read()
c1 = open("cipher1.txt","rb").read()
c2 = open("cipher2.txt","rb").read()
pub1 = RSA.importKey(f1)
pub2 = RSA.importKey(f2)
n1 = pub1.n
e1 = pub1.e
n2 = pub2.n
e2 = pub2.e
c1 = bytes_to_long(c1)
c2 = bytes_to_long(c2)
print("n1 =",n1)
print("e1 =",e1)
print("c1 =",c1)
print("n2 =",n2)
print("e2 =",e2)
print("c2 =",c2)
```

.

.

结果：

```
n1 =
13060424286033164731705267935214411273739909173486948413518022752305313862238166593214772698793487761875251
03042351699351971421530680867772410469247419921511938772574190607155343784025678622048458288469328614053749
25410930869530054867045424351885217240132510878873514099461845012952247448196219373224691407712453800816635
60150133162692174498642474588168444167533621259824640599530052827878558481036155222733986179487577693360697
39015237090174611265375833845608344087872600722930783003780868105030299041123866672760825345257369690408313
38660937919855651180327428932470769474807668379413192519015796052339160764255729 61
e1 = 117
c1 =
12847007370626420814721007824489512747227554004777043129889885590168327306344216253180822558098466760014640
87074828701652382826189026221088361333670476818286107501436837860941425598217976968658236521947765747494854
88867948079999527808409810219357339843480556420031163869390140046209142738400480617960634136419367545253747
90951194617245627213219302958968018227701794987747717299752986500496848787979475798026065928167197152995841
74784005002841753945938328073512422978995285943448074662357324106146555030300847873014089874074599903556359
91346677087534572117619698062780001264629187884577070986656124964546406161554770 50
n2 =
13060424286033164731705267935214411273739909173486948413518022752305313862238166593214772698793487761875251
03042351699351971421530680867772410469247419921511938772574190607155343784025678622048458288469328614053749
25410930869530054867045424351885217240132510878873514099461845012952247448196219373224691407712453800816635
60150133162692174498642474588168444167533621259824640599530052827878558481036155222733986179487577693360697
39015237090174611265375833845608344087872600722930783003780868105030299041123866672760825345257369690408313
38660937919855651180327428932470769474807668379413192519015796052339160764255729 61
e2 = 65537
c2 =
68308576617031565989734336170550458032770042742873009976346488004482336557564980706935978398560214312692375
65020303935757530559600152306154376778437832503465744084633164767864997303080852153757211172394903940863225
98114250288812692898200949397207601348675846089441671012281124990332243774224126968193455123743166818700617
64181249344887755058165447339292419279003929248866494209436993563142782556834849983596634046112360566641497
25644051300950988495549164517140159041907329062655574220869612072289849679613024196448446224406889484578310
51223266557118835162158552825550154694133278244644814403399706791798471910306851 9
```

.

.

.

这里两个相同的n，就是公模攻击了，在`CTF-RSA-tool`工具里有对公模攻击的解法：

```
# 共模攻击
python2 solve.py --verbose -i examples/share_N.txt
```

```
n = 49717352238903813258167965634872644122363546007764600124211711250852904300466263993456901629465018966 8
e = 13720370251305502198453722303188629715020031854527043028194234587721186750392338568840741980913602828 87
c = 33995928221963087827338803441589901571528069870645123081701004401546919289265619397264329787933998397 64

n = 49717352238903813258167965634872644122363546007764600124211711250852904300466263993456901629465018966 8
e = 13521927979417175825463063347222803267672338126236919097685639546419640649137631879086526055040440600 76
c = 20663727752331650736213993392793715555002283733124450484396596147661401938360252713595785342791730649 22
```

CSDN @沐一一沐，一沐沐一

.

.

所以摆好格式后直接跑脚本：

```
INFO: Here are your plain text:
flag{interesting_rsa}
```

.

.

(这里积累第二个经验)

当然完全依靠工具是不行的，所以抽取出`CTF-RSA-tool`对应代码做成脚本：(注意：这里因为用`python2`运行，所以不能用直接用`open().read()`来读取文档内容，会报错。在外面要裹上`libnum.s2n()`才行)

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *
import gmpy2
import libnum
def share_N(N, e1, e2, c1, c2):
    gcd, s, t = gmpy2.gcdext(e1, e2)
    if s < 0:
        s = -s
        c1 = gmpy2.invert(c1, N)
    if t < 0:
        t = -t
        c2 = gmpy2.invert(c2, N)
    plain = gmpy2.powmod(c1, s, N) * gmpy2.powmod(c2, t, N) % N
    print(libnum.n2s(plain))

c1=libnum.s2n(open('cipher1.txt','rb').read())
c2=libnum.s2n(open('cipher2.txt','rb').read())

pub1=RSA.importKey(open('publickey1.pem').read())
pub2=RSA.importKey(open('publickey2.pem').read())
n = pub1.n
e1= pub1.e
e2= pub2.e

share_N(n,e1,e2,c1,c2)
```

.

.

总结：

1：
(这里积累第一个经验)
从别人的博客 `https://www.cnblogs.com/vict0r/p/13542398.html` 中找到了从密钥文件和明文文件读取出对应数字的方法，提取 `n、e` 都是用 `Crypto.PublicKey.RSA` 模块，再抽取对应的 `n,c属性` 的。
提取加密 `密文c` ，则是直接二进制读取文件后用 `Crypto.Util.number` 模块的 `bytes_to_long` 函数转二进制流为数字的。

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *
f1 = open("publickey1.pem","rb").read()
f2 = open("publickey2.pem","rb").read()
c1 = open("cipher1.txt","rb").read()
c2 = open("cipher2.txt","rb").read()
pub1 = RSA.importKey(f1)
pub2 = RSA.importKey(f2)
n1 = pub1.n
e1 = pub1.e
n2 = pub2.n
e2 = pub2.e
c1 = bytes_to_long(c1)
c2 = bytes_to_long(c2)
print("n1 =",n1)
print("e1 =",e1)
print("c1 =",c1)
print("n2 =",n2)
print("e2 =",e2)
print("c2 =",c2)
```

2：
(这里积累第二个经验)
当然完全依靠工具是不行的，所以抽取出 `CTF-RSA-tool` 对应代码做成脚本：(注意：这里因为用python2运行，所以不能用直接用open().read()来读取文档内容，会报错。在外面要裹上libnum.s2n()才行)

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *
import gmpy2
import libnum
def share_N(N, e1, e2, c1, c2):
    gcd, s, t = gmpy2.gcdext(e1, e2)
    if s < 0:
        s = -s
        c1 = gmpy2.invert(c1, N)
    if t < 0:
        t = -t
        c2 = gmpy2.invert(c2, N)
    plain = gmpy2.powmod(c1, s, N) * gmpy2.powmod(c2, t, N) % N
    print(libnum.n2s(plain))

c1=libnum.s2n(open('cipher1.txt','rb').read())
c2=libnum.s2n(open('cipher2.txt','rb').read())

pub1=RSA.importKey(open('publickey1.pem').read())
pub2=RSA.importKey(open('publickey2.pem').read())
n = pub1.n
e1= pub1.e
e2= pub2.e

share_N(n,e1,e2,c1,c2)
```


解毕！敬礼！