Shiona 2021302191416

Easy_rsa:

从给定的文件 rsa 中可以看到已知 c,e,n，从源码中得知 flag 为原文 m。注意到使用了一个位数较小的私钥 d 来加密，再根据下面对于 d 和 n 大小关系的判断，初步定为低解密指数攻击，低解密指数攻击使用到了拓展维纳攻击。通过在 github 上寻找相应的源码，将 c,e,n 带入之后可以得出明文，将 16 进制的明文转化为 ascii 码即可达到 flag。此处使用 python 包 owiener，通过调用 attack 函数以实现得到密钥 d。

源码：

```
import hashlib
import owiener
d=owiener.attack(e,n)
print(hex(pow(c,d,n)))
```

Easy_md5:

从源文件中可以得知只要在告知姓名之后，读取发来的用于加密的比特流，将其进行 md5 加密后即可返回，不断削减恶龙血量。每次循环之后都会从过第三个选项得知恶龙当前血量，并且在血量低于 0 的时候选择离开，即可得到 flag

Baby_rsa:

从给定的包含多组 e,n,c 的文件当中可以得知，应当使用低加密指数的广播攻击。从文件当中读取 e 和多组 nc 的值，将其放于 list 当中，使用找到的广播攻击工具，调用其 attack 函数即可得到被加密的原文。

源码：

```
import gmpy2
import libnum
E = 11
N=[]
C=[]

f=open("ENC.txt","r")
for i in range(39):
    a=f.readline()
    N.append(int(a[2:-1]))
    a=f.readline()
    C.append(int(a[2:-1]))
```

```
        a=f.readline()
        a=f.readline()

    class BroadcastAttack:

        t = []
        messsage = 0

        def __init__(self, e, N, C):
            self.e = e
            self.N = N
            self.C = C

        def calculate_partials(self):
            for i in range(self.e):
                mod_product = 1
                for j in range(self.e):
                    if i != j:
                        mod_product *= self.N[j]
                t_i = self.C[i]*mod_product*ModUtil.modinv(mod_product,
self.N[i])
                self.t.append(t_i)

        def solve_congruence(self):
            partial_total = 0
            mod_product = 1
            for i in range(self.e):
                partial_total += self.t[i]
                mod_product *= self.N[i]

            self.message = ModUtil.isqrt(partial_total % mod_product, self.e)

        def attack(self):
            self.calculate_partials()
            self.solve_congruence()
            return self.message


    class ModUtil:
        @staticmethod
        def egcd(a, b):
            if a == 0:
                return (b, 0, 1)
            else:
```

```
            gcd, x_old, y_old = ModUtil.egcd(b % a, a)
            return (gcd, y_old - (b // a) * x_old, x_old)


    @staticmethod
    def modinv(a, m):
        gcd, x, y = ModUtil.egcd(a, m)
        if gcd != 1:
            raise Exception('Mod Inv Undefined')
        else:
            return x % m

    @staticmethod
    def isqrt(n, k):
        u, s = n, n+1
        while u < s:
            s = u
            t = (k - 1) * s + n // pow(s, k-1)
            u = t // k
        return s

attack=BroadcastAttack(E,N,C)
print(hex(attack.attack()))
```

Typing game_baby_guees_me:

从源码中得知，在首先需要对给定的 md5 反推其原文。因为位置的数据只有四位所以可以用暴力破解的方法，通过枚举 1kw 种可能性来与给定的 md5 值作比较得到位置的四位。这里通过从文件中读取事先写好的所有可能性来进行破解。在后面的推理当中，可以得知每次需要返回一个给定种子的随机数。由于 python 的随机数采用梅森旋转的方式进行生成，只需要得到了前面连续的 624 个 32 位的随机数，即可反推出整个随机数的 624 个状态，进而可以预测后面所有的随机数。这里使用在 github 上寻得到 RandCrack 的 python 包，对于前面已经给出的 624 个随机数将其提交给函数。之后的随机数则全部可以预测，从而在最后得到 flag。

源码：

```
import hashlib,string,random
from re import L
from pwn import *
from hashlib import sha256
from randcrack import RandCrack

rc=RandCrack()
```

```
io=remote("124.220.41.254","11115")
temp=io.recvuntil("sha",drop=False)
temp1=temp.split(b"==")

temp=io.recvline()
temp1=temp.split()
tail=temp1[0][9:-1].decode()
result=temp1[2].decode()
for x in temp1:
    print(x)

res=open("result.txt","r")
while True:
    head=res.readline()
    whole=head[0:4]+tail
    ans=sha256(whole.encode()).hexdigest()
    if(ans==result):
        print(head[0:4].encode())
        break;
io.sendline(head[0:4].encode())
temp=io.recvuntil("Give")

i=random.getrandbits(4)
prei=i
x=625
head="Mia is friendly. "
while x>=2:
    temp=io.recvline()
    temp1=temp.split(b"==")
    print(temp1[0])
    temp=io.recvline()
    temp1=temp.split(b"==")
    print(temp1[0])
    k=int(temp1[0][35:-1].decode())
    print(head,k)
    rc.submit(k)
    io.sendline((head+str(k)).encode())
    x-=1
for x in range(625,1000):
    temp=io.recvline()
    temp1=temp.split(b"==")
    print(temp1[0])
    temp=io.recvline()
    temp1=temp.split(b"==")
```

```
        print(temp1[0])
        k=rc.predict_getrandbits(32)
        print(head,k)
        io.sendline((head+str(k)).encode())

    while True:
        temp=io.recvline()
        temp1=temp.split(b"==")
        print(temp1[0])

    res.close()
```

Get_my_number:

从源码中得知，需要输入一个比 1000*4 的数更大但是比 1000 更小的数字，但是这里可以得知得到，前面的 1000*4 为无符号数，再进行比较的时候会将我们输入的数字视作无符号数与其进行比较。因而可以得知，当输入一个负数的时候，因为符号位的存在，它一定会比 4*1000 大；在后面的比较当中，因为 1000 为有符号数，所以一定会小于 1000，进而满足条件，可以对端口进行操纵。使用终端命令 find flag 即可寻找到 flag 文件，将其输出即可得到 flag。

No_copy:

在网页中使用开发者工具（f12）可以看到 flag 的内容。复制之后将代码中的所有<snap>删除之后即可得到正确的 flag。

Sign_in:

加入频道后获得。