Weekly Notes for CTF 🚩
Week 2

Astrageldon

2023-11-05

1 Crypto

1.1 SecretSharing

根据 Osint 大法可以知道本题是 Shamir's secret sharing 的一种实现。然而知道这一点对于解出这个问题并没有任何的帮助。看一眼代码逻辑，它等价于要我们求以下方程组中，A_0 的解：

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^{n-1} \\ 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{t-1} & \cdots & x_{t-1}^{n-1} \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{n-1} \end{pmatrix} = \begin{pmatrix} R_0 \\ R_1 \\ \vdots \\ R_{t-1} \end{pmatrix}$$

在以上方程中，x_{t-1} 的未知给我们带来了很大的麻烦。但是可以发现，数组 X 是通过 random.getrandbits 函数生成的，也就是用到了梅森随机数的算法。因此，只要得到 $624 * 32$ 比特位的信息，就可以预测之后生成的所有随机数了。

```
import random
from randcrack import RandCrack

rc = randcrack.RandCrack()
nums = [random.getrandbits(32) for _ in range(624)]
for i in nums:
        rc.submit(i)

print(rc.predict_getrandbits(32))
print(random.getrandbits(32))

#2063303864
#2063303864
```

每次 submit 需提交一个 32 比特位的数字，而对于位数高于 32 的情况，通过以下的例子可以知道，随机数的生成存在一个以 32 比特位为一组由低位向高位生长的过程：

```
import random
```

```
2
3  random.seed(1)
4  print(hex(random.getrandbits(32)))
5  print(hex(random.getrandbits(32)))
6
7  random.seed(1)
8  print(hex(random.getrandbits(64)))
9  #0x2265b1f5
10 #0x91b7584a
11 #0x91b7584a2265b1f5
```

因此 submit 的时候是从低位向高位进行的，我们可以完善一下 RandCrack 类：

```
1  class RandCrack(randcrack.RandCrack):
2          def __init__(self):
3                  self.count = 0
4                  super().__init__()
5
6          def _submit(self, num, bits):
7                  for i in range(0,bits,32):
8                          if self.count < 624:
9                                  self.submit((num>>i)%2**32)
10                                 self.count += 1
```

通过数组 X 的前 20 个 1024 比特位的整数，便可以得到 X 的第 21 个整数，从而我们得到了系数矩阵的完整信息。而众所周知，$x_1, \cdots, x_{n-1}$ 两两不同时，由它们构成的 Vandermonde 行列式不为 0，从而该线性方程组存在唯一解。

Exp 如下

sol.py

```
1  import randcrack
2
3  class RandCrack(randcrack.RandCrack):
4          def __init__(self):
5                  self.count = 0
```

```
                super().__init__()

        def _submit(self, num, bits):
                for i in range(0,bits,32):
                        if self.count < 624:
                                self.submit((num>>i)%2**32)
                                self.count += 1

def getdata():
        X = []
        R = []
        with open('output.txt', 'r') as f:
                for xr in f.readlines():
                        X.append(int(xr.split(' ')[0]))
                        R.append(int(xr.split(' ')[1]))
        return X, R

def main():
        X, R = getdata()
        rc = RandCrack()
        for x in X[:20]:
                rc._submit(x, 1024)
        [rc.predict_getrandbits(32) for _ in range(640-624)]
        X20 = rc.predict_getrandbits(1024)
        with open('data.py', 'w') as f:
                f.write(f'X = {X + [X20]}\nR = {R + [leak]}')

leak =

if "__main__" == __name__:
        main()
```

sol2.sage

```
from Crypto.Util.number import *
import data

def main():
        M0 = [[x**j for j in range(21)] for x in data.X]
        M  = matrix(Zmod(p), M0)
```

```
        P  = int(M.solve_right(matrix(Zmod(p),21,1,data.R
            ))[0,0])

        assert not N % P

        Q   = N // P
        phi = (P-1)*(Q-1)
        d   = pow(e,-1,phi)
        m   = pow(c,d,N)

        return int(m)

#data...

if "__main__" == __name__:
        print(long_to_bytes(main()))
```

在结束之前，笔者想留给读者一个思考：如果泄露的不是 R 的最后一个数，而是 A 的最后一个数，又该如何解决呢？

Flag: flag{2f43430b-3c31-03ee-0a92-5b24826c015c}