guess_game

发表于 2021-01-21 分类于 Challenge， 2020， 网鼎杯， 朱雀场， Crypto
Challenge | 2020 | 网鼎杯 | 朱雀场 | Crypto | guess_game

点击此处获得更好的阅读体验

WriteUp来源

本WP由Vanish提供

题目考点

- *LCG*
- *LFSR*

解题思路

LCG攻击原理这里看，NCTF出现过的。

LFSR的攻击手法ctf-wiki上看。

解密流程：

第一步我们通过六个level的值恢复LCG的参数从而预测第七个code，从而达到(500,10)这个level。~~开局拥有10个coin！~~

然后我们用9次机会，可以得到72bit的输出，其中前40bit输出逆序可以得到初始的r

但是这里的lfsr其实是39级的，所以至少需要78bit才能恢复

所以我们要爆破6bit的输出

需要说明的是即使有78bit的输出，也有可能因为矩阵没有满秩从而解不出key来。

然后这里我换了个思路，我硬猜，要求前九次猜中一个，这样我就可以多拿两组数据（其实多拿一次就够了）

这样我就能拿到80bit的输出，然后我用下面那个sage脚本，根据wiki所述构造矩阵，解个线性方程，得到key。~~点解密的时候要放《好运来》，增加解密成功的概率，唱《国歌》应该也可以。~~

```
output = '10100010010100010010101001100110000000000111101111011101010101000111110011111011'

N = 39
F = GF(2)
ans=[]
out = output
Sn = [vector(F,N) for j in range(N+1)]
for j in range(N+1):
    Sn[j] = list(map(int,out[j:j+N]))

X = matrix(F,Sn[:N])
invX = (X^-1)
Y = vector(F,Sn[-1])
Cn = Y * invX
res = ''.join(str(i) for i in Cn)
ans.append(int(res[::-1],2))
print (ans)
print(len(ans))

>> [71834525659]
>> 1
```

用于交互的脚本

```
# -*- coding: utf-8 -*-
from Crypto.Util.number import long_to_bytes,bytes_to_long,inverse
from pwn import *
```

```
4  context.log_level = 'debug'
5  def gcd(a, b):
6      while b:
7          a, b = b, a%b
8      return a
9
10 def crack_unknown_increment(states, modulus, multiplier):
11     """
12     利用伪随机序列s,模数n,乘数m，恢复增数c
13
14     states:伪随机数序列s
15     modulus:模数n，
16     multiplier:乘数m
17     increment：增量c
18     return:根据n,m,s0,s1恢复c，并返回
19     """
20     increment = (states[1] - states[0]*multiplier) % modulus
21     return modulus, multiplier, increment
22
23 def crack_unknown_multiplier(states, modulus):
24     """
25     利用伪随机序列s和模数n，恢复乘数m
26
27     states:伪随机数序列s
28     modulus:模数n，
29     multiplier:乘数m
30     return:根据n,s0,s1,s2恢复m，然后利用crack_unknown_increment恢复c
31     """
32     multiplier = (states[2] - states[1]) * inverse(states[1] - states[0], modulus) % modulus
33     return crack_unknown_increment(states, modulus, multiplier)
34
35 def crack_unknown_modulus(states):
36     """
37     利用初值和随后LCG产生的连续的几个值,恢复模数n
38
39     modulus:模数n
40     states:伪随机数序列s
41     diffs:相邻随机数s的差值：t序列
42               zeroes:与t系列有关的差值，zeroes = t2*t0 - t1*t1,t0,t1,t2相邻
43     return:根据n,s0,s1,s2恢复m，然后利用crack_unknown_increment恢复c
44     """
45     diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
46     zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:], diffs[2:])]
47     modulus = abs(reduce(gcd, zeroes))
48     return crack_unknown_multiplier(states, modulus)
49
50 def lcg(seed,params):
51     """
52     LCG算法
53
54     params:(multiplier,increment,modulus)
55     seed:伪随机序列中第一个值s0
56     return：返回一个迭代器，其中的值是私钥序列
57     """
58     (m,c,n)=params
59     x = seed % n
60     yield int(x)
61     while True:
62         x = (m * x + c) % n
63         yield int(x)
64
65 def decode(pri,pub,c,p):
66     """
67     ElGamal解密算法
68
69     pri:一方私钥
70     pub:另一方公钥
71     sharekey:双方协商密钥
72     c:密文
73     p:模数
74     m:明文
75     return:返回明文字符
76     """
77     sharekey = pow(pub,pri,p)
78     m=long_to_bytes(c*inverse(sharekey,p)%p)
79     return m
```

```
80
81 class Game:
82     def __init__(self, n, key, r, b):
83         self.n = n#40
84         self.key = key
85         self.r = r & (2 ** self.n - 1)
86         self.b = b#8
87
88     def out(self):
89         o = self.r & 1
90         p = self.r & self.key
91         q = 0
92         while p:
93             q = q + p & 1
94             p = p >> 1
95         q = q & 1
96         t = q << (self.n - 2) & (2 ** self.n - 1)
97         self.r = t | (self.r >> 1) & (2 ** self.n - 1)
98         return o
99
100    def next(self):
101        res = 0
102        for i in range(self.b):
103            o = self.out()
104            res |=  o << (self.b - 1 - i)
105        return res
106
107 def getr(s):
108     r = ''
109     for i in s:
110         b = bin(i)[2:].rjust(8,'0')
111         for each in b:
112             r = each+r
113     print int(r,2)
114     return int(r,2)
115
116 def getk(s):
117     k=""
118     for i in s:
119         b = bin(i)[2:].rjust(8,'0')
120         for each in b:
121             k += each
122     return k
123
124 sh = remote('59.110.243.101','5123')
125 sh.recvuntil("Your choice:\n")
126 sh.sendline("2")
127 sh.recvuntil("Baby:")
128 Baby = sh.recvuntil("\n")[:-1]
129 sh.recvuntil("Easy:")
130 Easy = sh.recvuntil("\n")[:-1]
131 sh.recvuntil("Normal:")
132 Normal = sh.recvuntil("\n")[:-1]
133 sh.recvuntil("Hard:")
134 Hard = sh.recvuntil("\n")[:-1]
135 sh.recvuntil("Master:")
136 Master = sh.recvuntil("\n")[:-1]
137 sh.recvuntil("Crazy:")
138 Crazy = sh.recvuntil("\n")[:-1]
139
140 pri=[Baby,Easy,Normal,Hard,Master,Crazy]
141
142 (n,m,c)=crack_unknown_modulus([int(each) for each in pri])
143
144 x=int(pri[0])
145
146 key = lcg(x,(m,c,n))
147 for _ in range(6):
148     next(key)
149 choice = next(key)
150 sh.recvuntil("Input the level code:\n")
151
152 sh.sendline(str(choice))
153 sh.recvuntil("Your choice:\n")
154 sh.sendline("1")
155 #flag=""
```

```
s=[]
F = 0
for _ in range(9):
    sh.recvuntil("Input your answer:\n")
    sh.sendline("0")
    ans = sh.recvuntil("The answer is ")
    if 'Right' in ans:
        F = 1
        print("NICE")
    s.append(int(sh.recvuntil("!")[:-1]))

if F == 1:
    sh.recvuntil("Input your answer:\n")
    sh.sendline("0")
    ans = sh.recvuntil("The answer is ")
    s.append(int(sh.recvuntil("!")[:-1]))

r = getr(s[:5])
output = getk(s[:])
print "r:"+str(r)
print "output:"+ output
print s
k = input("k:")
game = Game(40,k,r,8)
for _ in range(10):
    next(game)

for _ in range(500):
    sh.recvuntil("Input your answer:\n")
    sh.sendline(str(next(game)))

sh.interactive()

[DEBUG] Received 0x3f bytes:
    'Your coin: 500\n'
    'You Win!\n'
    'flag{12727f129b72e685388c6a67538ee9b0}\n'
```


 # Challenge # 2020 # Crypto # 网鼎杯 # 朱雀场
simple
Safe_box