Hitcon2018-Lost key

0x00 RSA加解密

$$C = Encrypt(P) = P^e \ mod \ N$$

$$P = Decrypt(C) = C^d \ mod \ N$$

0x01 利用选择明文攻击获取N：

选择一明文 a，并按如下方式加密

$$R_{a1} = Encrypt(a) = a^e \ mod \ N = a^e - k_{a1} * N$$

$$R_{a2} = Encrypt(a^2) = a^{2e} \ mod \ N = a^{2e} - k_{a2} * N$$

$$\begin{aligned} KN_1 = R_{a1}^2 - R_{a2} &= (a^e - k_{a1} * N)^2 - (a^{2e} - k_{a2} * N) \\ &= k_{a1}^2 * N^2 - 2a^e k_{a1} * N + k_{a2} * N \end{aligned}$$

同理加密明文 b 可得

$$KN_2 = k_{b1}^2 * N^2 - 2b^e k_{b1} * N + k_{b2} * N$$

KN_1和KN_2都含有因子N，这样我们选择多组明文以得到多个KN，然后计算$gcd(KN_1, KN_2, ...)$，则在很大概率上有

$$N = gcd(KN_1, KN_2, ...)$$

在RSA相关题目中，一般取四五组明文便可确定N，本题目取2, 3, 4, 5四组明文即可。

```python
# get n
r = []  # remainder
for i in [2,3,4,5]:
    p.sendlineafter("cmd:", "A")
    p.sendlineafter("input:", str(i).zfill(2))
    r.append(p.recvline())
    p.sendlineafter("cmd:", "A")
    p.sendlineafter("input:", str(hex(i**2)[2:]).zfill(2))
    r.append(p.recvline())
kn = []
for i in range(len(r)/2):
    kn.append(int(r[2*i],16)**2-int(r[2*i+1],16))
# print kn
gcd = kn[0]
for i in range(len(kn)):
    gcd = GCD(kn[i],gcd)
n = gcd
```

0x02 Byte Oracle

总思路：利用服务器解密算法所返回的最后一个字节来确定flag的范围。

$$cipher256 = Encrypt(256) = 256^e \ mod \ N$$

$$C = C * cipher256 \tag{1}$$

$$\begin{aligned} Decrypt(C) &= Decrypt((256P)^e \ mod \ N) \qquad (2) \\ &= 256P \ mod \ N \\ &= 256P - kN \end{aligned}$$

则有

$$kN \leq 256P < (k+1)N$$

$$\frac{kN}{256} \leq P < \frac{(k+1)N}{256}$$

一般$P < N$，故有$k \in [0, 255]$

服务器返回的结果是$256P - kN$的最后一个字节，即$(256P - kN) \ mod \ 256 = -kN \ mod \ 256$ 为了确定 k 的值，我们可以构造一张 $lastbyte - k$ 的映射表：

```python
# get lastbyte--k map
mapTable = {}
for i in range(256):
    mapTable[-i*n%256] = i
```

这样确定 k 后也就确定了 P 的一个初始范围，但这个初始范围是非常大的，我们需要重复(1)(2)步骤来不断缩小 P 的范围。下面用数学归纳法推导 P 取值范围在这个过程中的的收敛规律：

设第 i 次迭代后

$$\frac{xN}{256^i} \leq P < \frac{(x+1)N}{256^i} \tag{3}$$

则第 $i+1$ 次迭代时

$$C = cipher256^{i+1} * C_0$$

$$Decrypt(C) = 256^{i+1}P - kN(k = 256y - k_{i+1}, k_{i+1} \in [0, 255])$$

$$\frac{256yN + k_{i+1}N}{256^{i+1}} \leq P < \frac{256yN + (k_{i+1}+1)N}{256^{i+1}}$$

$$\frac{yN + \frac{k_{i+1}N}{256}}{256^i} \le P < \frac{yN + \frac{(k_{i+1}+1)N}{256}}{256^i} \tag{4}$$

可用反证法证明 $y = x$ ，首先假设 $y < x$ ，则 $y \le x - 1$ ，带入 (4) 式右侧得

$$P < \frac{yN + \frac{(k_{i+1}+1)N}{256}}{256^i} \le \frac{xN - N + \frac{(k_{i+1}+1)N}{256}}{256^i} \le \frac{xN}{256^i}$$

上式表明第 i 次迭代所得 P 的范围与第 $i+1$ 次得到的范围没有交集，这与 P 的存在性相矛盾，同理可证 $y > x$ 亦不成立，故得证 $y = x$ ，则第 $i+1$ 次迭代后所得范围

$$\frac{xN + \frac{k_{i+1}N}{256}}{256^i} \le P < \frac{xN + \frac{(k_{i+1}+1)N}{256}}{256^i} \tag{5}$$

其中 k_{i+1} 即从 $lastbyte - k$ 映射表中所获取的 k 值。

由 (3)(5) 两式可以看出 P 范围收敛的一般规律，设初始范围

$$\frac{P}{N} \in [L_0, R_0], (L_0 = 0, R_0 = 1)$$

第 i 次迭代后

$$L_i = L_{i-1} + \frac{k_i}{256^i}$$
$$R_i = L_{i-1} + \frac{k_i + 1}{256^i}$$

至此，还剩最后一个问题，就是迭代次数。由于 $flag$ 的最后一个字节可以直接由服务器解密得到，我们只需要限定 P 的范围至 $P_{max} - P_{min} < 256$ ，即

$$N * (R_i - L_i) = \frac{N}{256^i} < 256$$

其中 $N = getPrime(512) * getPrime(512)$ 则 $N < 2^{1024}$ ，可由上式解得 $i \ge 128$ 。一般规律为对于这样一个Oracle，最多需要 $(\log_{2^{bits}} N)$ 次迭代即可确定明文，其中 $bits$ 为泄露的明文 bit 数，本题中 $bits = 8$ 即一个字节。

```python
# get flag
L=0  # the initial range of flag/n is [0,1)
for i in range(128):
    cipherflag = cipherflag * cipher256 % n
    p.sendlineafter("cmd:", "B")
    p.sendlineafter("input:", long_to_hex(cipherflag))
    temp_lastbyte = int(p.recvline(),16)
    k = mapTable[temp_lastbyte]
    L = L + Fraction(k, 256**(i+1))

flag = int(L*n)-int(L*n)%256+lastbyte
print long_to_hex(flag).decode('hex')
```

```
root@CLtheorem:~/downloads/hitcon/lostkey# ./lostKeyExp.py
[+] Starting local process './encrypt.py': pid 18836
\xaaV��-\xb0\x9e#\xb8C\xab<(B`\xb8\x7f\x88\xba\x86a\xb0\xa4"�/�D\x89T\x030�Mu�H\xa2\x8d\x1
7\x9d\x9d\x93[\x89\x8e\xb8u\x83\x82\x00�.\x10\x83\x9a\x83`\x04\x87�qlF1hitcon{AAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABC}

[*] Stopped process './encrypt.py' (pid 18836)
```