ABCTF 2016

Old RSA Write Up

Author : hideroot(M.O.K)

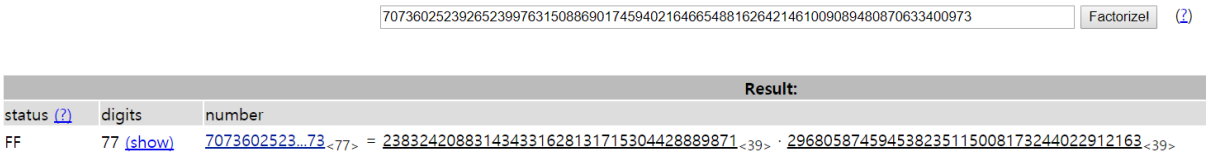
Data : 10/11 2017

간단한 RSA 문제다. RSA에 대해서 설명을 간단하게 하자면 NP문제 즉, 아주 큰 두개의 소수 p, q 를 곱한 N 값이 있을 때 이 N 값을 소인수분해 하여 다시 p, q 로 나타내기 힘들다는 문제에 기반하여 만들어진 암호다.

물론 현재는 아주 크더라도 어느정도 테이블이 나와 있는 상태다. 주어진 숫자들을 보자

```
c = 29846947519214575162497413725060412546119233216851184246267357770082463030225
n = 70736025239265239976315088690174594021646654881626421461009089480870633400973
e = 3
```

c 와 n , e 가 주어졌다. C는 암호화된 메시지 일 것이고 여기서 대부분 RSA의 n을 두수의 곱으로 나타낸다. 그래서 n을 테이블에 넣어서 확인을 해보자



http://www.factordb.com/index.php?query=70736025239265239976315088690174594021646654881626421461009089480870633400973

그렇게 긴 숫자가 아니기 때문에 충분히 테이블에 등록이 되어 있다. N은 39자리 수의 곱으로 이루어져 있다는 걸 알 수 있다. 두 소수 까지 알았다면 RSA는 완전히 뚫렸다.

이제 n을 통해 구한 개인키를 이용해 c를 복호화 시키면 끝난다.

```
from Crypto.PublicKey import RSA
import gmpy
n = 707360252392652399763150886901745940216466548816264214610090894808706334009730
c = 298469475192145751624974137250604125461192332168511842462673577700824630302250
p = 23832420883143433162813171530442888987l
q = 29680587459453823511500817324402291216 3
e = long(3)

d = long(gmpy.invert(e,(p-1)*(q-1)))
key = RSA.construct((n,e,d))
print key.decrypt(c)
print hex(key.decrypt(c))
print hex(key.decrypt(c))[2:-1].decode("hex")
```

여기서 gmpy와 Crypto 모듈을 사용했다. Gmpy는 큰 수를 extention 해주는 모듈이다.

```
687255797750574777816118221724271222836487386007058011149452654604 5
0x41424354467b746831735f7761735f683472645f696e5f313938307dL
ABCTF{th1s_was_h4rd_in_1980}
```