../home

TAMUctf 2017 - Dachshund

April 27, 2017

Solved by: @dbaser and @shrimpgo **Category:** Cryptography **Points:** 100 **Solves:** 66 **Description:**

> >> *Attacking this challenge with a dachshund is your best bet at winning.*

Write-up

let's see what's the type of the file

```
 $ file df5e76dedfe9afc0


df5e76dedfe9afc0: ASCII text, with very long lines
```

the content of the file given us the `n` (modulus), `e` (public exponent) and `c` (cyphertext), RSA crypto here.

```
 $ cat df5e76dedfe9afc0


C: AR/ar3SualKVNKXZ4ox9JNlajNxTAhRRwI09n/F5LaL066s0LPZPdwwnU5r5h6o...
e: MzY3MTgyMDc5NDYzMjY5NDg2OTg3MDcyODc2OTg0MzE0MDM3NDg2OTA2Mjg4OTk...
N: NDY0NTE3NDY1NDA2Nzg1OTUzODU3NTU2NDU3NjQ5NTMxOTUwMjkzNzkyNDY5NzI...
```

ok, after searching for dachshund on google, we found a hint for the attack type, because de nickname from de dachshund is “Wiener-Dog”

Dachshund

From Wikipedia, the free encyclopedia

"Wiener-Dog" redirects here. Fo

The **dachshund** (UK /ˈdæksənd/ or

breed.

there are some of types of attacks:

- Weak public key factorization
- **Wiener's attack** (*wiener-dog!*)
- Hastad's attack (Small exponent attack)
- Small q (q<100,000)
- Common factor between ciphertext and modulus attack
- Fermat's factorisation for close p and q
- Gimmicky Primes method
- Past CTF Primes method

  - Self-Initializing Quadratic Sieve (SIQS) using Yafu
  - Common factor attacks across multiple keys

after google, we found this write-up with the solution to this chall, only replace `n`, `e` and `c`.

final script! we have to convert `n` and `e` from base64 to ascii and `c` from base64 to hex

```
#!/usr/bin/python
import ContinuedFractions, Arithmetic, RSAvulnerableKeyGenerator
import time
import sys
import base64
import binascii
import gmpy
import sympy
import math
import fractions
import struct
sys.setrecursionlimit(100000)
# modulus from the RSA public key
n=4645174654067859538575564576495319502937924697297596750757351560512816296707979225
# exponent from the RSA public key
e=3671820794632694869870728769843140374869062889975259470509935206716056282640601372
# cyphertext converted to hex
c=0x011fdaaf74ae6a529534a5d9e28c7d24d95a8cdc53021451c08d3d9ff1792da2f4ebab342cf64f77
def hack_RSA(e,n):
 print "Performing Wiener's attack. Don't Laugh..."
 time.sleep(1)
 frac = ContinuedFractions.rational_to_contfrac(e, n)
 convergents = ContinuedFractions.convergents_from_contfrac(frac)
 for (k,d) in convergents:
   #check if d is actually the key
   if k!=0 and (e*d-1)%k == 0:
     phi = (e*d-1)//k
     s = n - phi + 1
     # check if the equation x^2 - s*x + n = 0
     # has integer roots
     discr = s*s - 4*n
     if(discr>=0):
       t = Arithmetic.is_perfect_square(discr)
       if t!=-1 and (s+t)%2==0:
         return d
hacked_d = hack_RSA(e, n)
print "d=" + str(hacked_d)
m = pow(c, hacked_d, n)
```

```
print "So the flag is:"
print("%0512x" %m).decode("hex")
```

run the script and don’t forget to clone this repo before running the script

```
 $ python rsa.py

Performing Wiener's attack. Don't Laugh...
d=34423659517451757817217793949772913434630556566965109599840482542632279361311
So the flag is:
gigem{h0Tdogs_85faf27b642d2f94}
```

The flag is: `gigem{h0Tdogs_85faf27b642d2f94}`

« prev

next »

//Comments...