

Network Security Project1

Hacking the Cipher

0416246 王彦茹

Dependency

- pycrypto
- pwntools
- base64
- binascii
- sys
- os
- signal
- math

step0. Extract public key from pub.pem

- use `RSA.importKey()` to get `key.n` and `key.e`

```
def getpubkey():  
    with open('./pub.pem', 'rb') as f:  
        pub = f.read()  
        key = RSA.importKey(pub)  
    return key
```

```
key = getpubkey()  
n = key.n  
e = key.e
```

step1. Choose X where X is relatively prime to n

- use `gcd()` to check wheter X is relatively prime to n

```
def coprime(a, b):  
    return bltin_gcd(a, b) == 1
```

```
tmp = 3  
while (coprime(tmp, n)!=True):  
    tmp += 1  
X = tmp
```

step2. Create $Y = C * X^e \text{ mode } n$

- read from flag.enc
- use `base64.b64decode()` to decode the flag
- use `binascii.hexlify()` to transfer byte string into integer
- convert hexadecimal to decimal
- create Y

```
with open('./flag.enc', 'r') as f:
    flag = f.read().strip()
    flag = base64.b64decode(flag)

    C = int(binascii.hexlify(flag), 16)

    Y = C * (X**e) % n
```

step3. Connect to server to get $Z = \text{decrypted } Y$

- use `binascii.unhexlify` to transfer the Y above from integer into byte string
- use `base64.b64encode` to encode the Y before send to server
- use `pwnlib's remote` to connect to the server
- use `recvuntil` to receive the server's first reply
- send Y to the server
- retrieve the decrypted Y
- use `base64.b64decode` and `binascii.hexlify` to decode and transfer into integer

```
en = SHA256.new()
en = binascii.unhexlify(hex(Y)[2:])

send_code = base64.b64encode(en)

#connect to server nc 140.113.194.66 8888
conn = remote('140.113.194.66', 8888)
reply = conn.recvuntil(':')

conn.sendline(send_code)

conn.recvline() #get the server reply 'decrypted message.....\n'
Z = conn.recvline() #get the decrypted Y
conn.close()

Z = base64.b64decode(Z)
Z = int(binascii.hexlify(Z), 16)
```

step4. Find out modular inverse of X

- `x_inverse = modinv(X, n)`

```
def egcd(a,b):
    if a==0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b%a, a)
        return (g, x-(b // a)*y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g!= 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
```

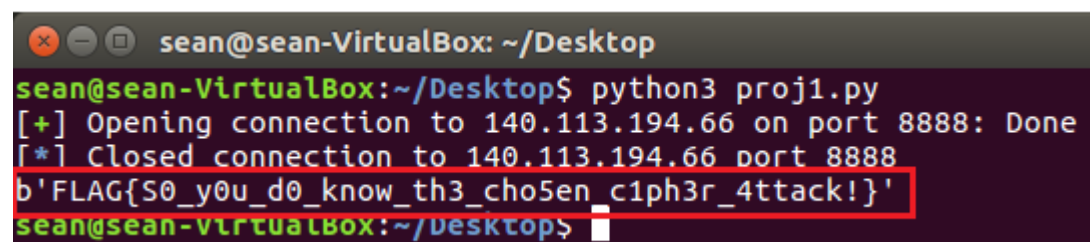
step5. $P = Z * (\text{inverse } X) \text{ mode } n$

- solve P
- use `binascii.unhexlify()` to transfer hexadecimal P into byte string
- write the decrypted flag to the file

```
P = (Z * x_inverse) % n

ans = SHA256.new()
ans = binascii.unhexlify(hex(P)[2:])
#write ans to file
print(ans)
with open('flag', 'wb') as file:
    file.write(ans)
```

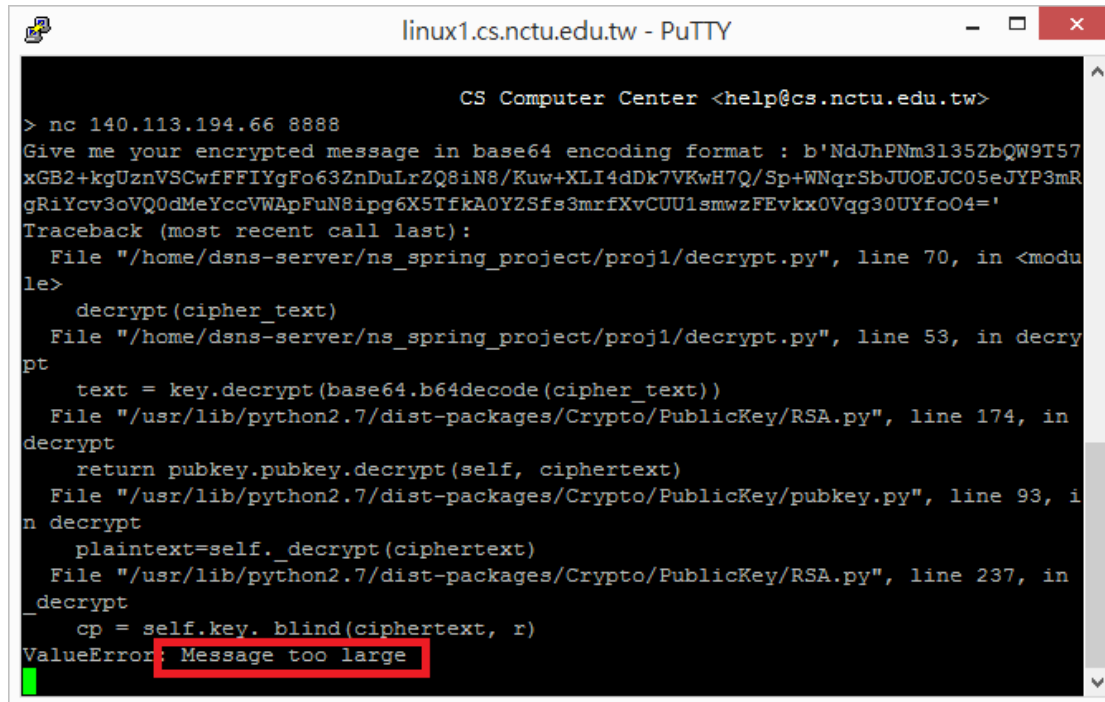
Code execution screenshot



```
sean@sean-VirtualBox: ~/Desktop
sean@sean-VirtualBox:~/Desktop$ python3 proj1.py
[+] Opening connection to 140.113.194.66 on port 8888: Done
[*] Closed connection to 140.113.194.66 port 8888
b'FLAG{S0_y0u_d0_know_th3_ch05en_c1ph3r_4ttack!}'
sean@sean-VirtualBox:~/Desktop$
```

Problem encounter

1. message too large



```
linux1.cs.nctu.edu.tw - PuTTY

CS Computer Center <help@cs.nctu.edu.tw>

> nc 140.113.194.66 8888
Give me your encrypted message in base64 encoding format : b'NdJhPNm3l35ZbQW9T57
xGB2+kgUznVSCwffFIYgFo63ZnDuLrZQ8iN8/Kuw+XLI4dDk7VKwH7Q/Sp+WNqrSbJUOEJC05eJYP3mR
gRiYcv3oVQ0dMeYccVWApFuN8ipg6X5TfkA0YZSfs3mrfXvCUU1smwzFEvKx0Vgq30UYfo04='
Traceback (most recent call last):
  File "/home/dsns-server/ns_spring_project/proj1/decrypt.py", line 70, in <modu
le>
    decrypt(cipher_text)
  File "/home/dsns-server/ns_spring_project/proj1/decrypt.py", line 53, in decry
pt
    text = key.decrypt(base64.b64decode(cipher_text))
  File "/usr/lib/python2.7/dist-packages/Crypto/PublicKey/RSA.py", line 174, in
decrypt
    return pubkey.pubkey.decrypt(self, ciphertext)
  File "/usr/lib/python2.7/dist-packages/Crypto/PublicKey/pubkey.py", line 93, i
n decrypt
    plaintext=self._decrypt(ciphertext)
  File "/usr/lib/python2.7/dist-packages/Crypto/PublicKey/RSA.py", line 237, in
_decrypt
    cp = self.key.blind(ciphertext, r)
ValueError: Message too large
```

(sol) Since I send the wrong format of the code, the server couldn't decrypt the code. Then I realize that I have to use `hex(Y)[2:0]` when doing `binascii.unhexlify` which is the value of Y excluding the '0x' symbols.

```
en = SHA256.new()
en = binascii.unhexlify(hex(Y)[2:])

send_code = base64.b64encode(en)
```

2. unable to send code to server before timeout

```
sean@sean-VirtualBox:~/Desktop$ python3 proj1.py
[+] Opening connection to 140.113.194.66 on port 8888: Done
b'Give me your encrypted message in base64 encoding format : Timeout. Bye~\n'
Traceback (most recent call last):
  File "proj1.py", line 70, in <module>
    conn.recvline() #get the server reply 'decrypted message.....\n'
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/tube.py", line 424, in recvline
    return self.recvuntil(self.newline, drop=not keepsends, timeout=timeout)
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/tube.py", line 301, in recvuntil
    res = self.recv(timeout=self.timeout)
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/tube.py", line 75, in recv
    return self._recv(num, timeout) or b''
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/tube.py", line 149, in _recv
    if not self.buffer and not self._fillbuffer(timeout):
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/tube.py", line 123, in _fillbuffer
    data = self.recv_raw(4096)
  File "/usr/local/lib/python3.5/dist-packages/pwnlib/tubes/socket.py", line 49, in recv_raw
    raise EOFError
EOFError
[*] Closed connection to 140.113.194.66 port 8888
```

(sol) I was trying to use `recvline()` to get the server's first message "Give me your encrypted message in base64 encoding format :", but it failed. Since `recvline()` only returns when it receive a '\n', I used `recvuntil(':')` instead to make sure it will return when it receives the server's first message.

```
#connect to server nc 140.113.194.66 8888
conn = remote('140.113.194.66', 8888)
reply = conn.recvuntil(':')

conn.sendline(send_code)

conn.recvline() #get the server reply 'decrypted message.....\n'
Z = conn.recvline() #get the decrypted Y
conn.close()
```