

Project2 SSL

0316025 – 賴文揚

Get password from websocket (No encryption)

- Password:)o()O(AliceForBob

The image shows a Wireshark packet capture window. The top pane displays the raw data of a packet, which is a JSON message: `sage",{"username":"Alice","msg":"And the password for the file is \"')o()O(AliceForBob\""}]`. The middle pane shows the packet's structure in a tree view, with the message content expanded. The bottom pane shows the packet's details, including the message content: `sage",{" username`. The status bar at the bottom indicates the packet number (3611), time (162.792658), source (172.17.0.2), destination (172.17.0.3), protocol (WebSocket), length (170), and info (WebSocket Text [FIN]).

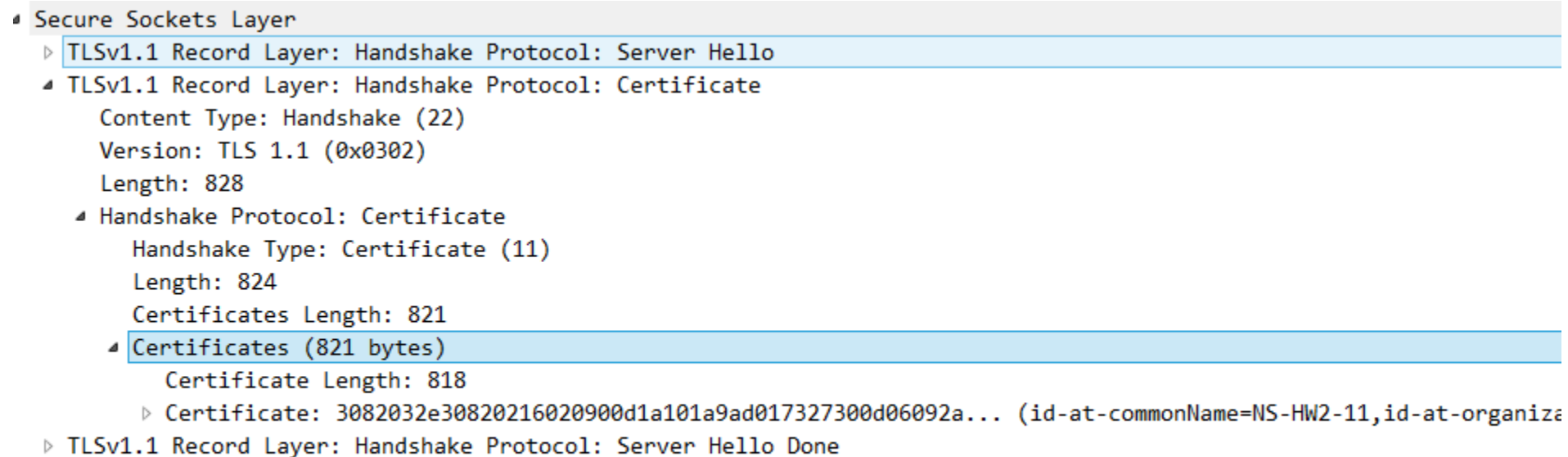
0000 02 42 ac 11 00 03 02 42 ac 11 00 02 08 00 45 00 .B.....BE.
0010 00 9c 84 2f 40 00 40 06 5e 05 ac 11 00 02 ac 11 .../@.@. ^.....
0020 00 03 0b b8 a4 34 fe b9 a2 9d 42 e4 63 08 80 184.. ..B.c...
0030 00 ec 58 b6 00 00 01 01 08 0a 00 05 b5 2a 00 05 ..X.....*...
0040 b5 2a 81 66 34 32 5b 22 63 68 61 74 20 6d 65 73 .*.f42[" chat mes
0050 73 61 67 65 22 2c 7b 22 75 73 65 72 6e 61 6d 65 sage",{" username
0060 22 3a 22 41 6c 69 63 65 22 2c 22 6d 73 67 22 3a ": "Alice ", "msg":
0070 22 41 6e 64 20 74 68 65 20 70 61 73 73 77 6f 72 "And the passwor
0080 64 20 66 6f 72 20 74 68 65 20 66 69 6c 65 20 69 d for th e file i
0090 73 20 5c 22 29 6f 28 29 4f 28 41 6c 69 63 65 46 s \"')o() O(AliceF
00a0 6f 72 42 6f 62 5c 22 22 7d 5d orBob\""}]]

No.: 3611 · Time: 162.792658 · Source: 172.17.0.2 · Destination: 172.17.0.3 · Protocol: WebSocket · Length: 170 · Info: WebSocket Text [FIN]

Close Help

Export certifications

- Export certificates as .der file
 - Right click, “Export packet bytes”



Convert format

- First, convert .der file to pem format file
 - `$ openssl x509 -inform der -in in_file.der -outform pem -out out_file.pem`
- Second, convert pem format file to public key format
 - `$ openssl x506 -pubkey -noout -in in.pem > pubkey.pem`

RSA algorithm

- If we can divide the n , we can gain the private key.
 - Do common factorizing.

Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Common factorizing

- Using python-crypto library to load public key file and calculate gcd from pair of these 12 public files.
 - \$./common_factorize.py

```
1 #!/usr/bin/python
2
3 from Crypto.PublicKey import RSA
4 from fractions import gcd
5 pems = []
6 for i in range(1, 11):
7     p = open("pems/pubkey_" + str(i) + ".pem").read()
8     pems.append(RSA.importKey(p))
9
10 for i in range(0, 10):
11     for j in range(i+1, 10):
12         n = gcd(pems[i].n, pems[j].n)
13         if n != 1:
14             print (str(i) + " and " + str(j) + " have common factor: ")
15             print (n)
16
17
```

Common factorizing (Cont.)

- We can get the information below.

```
10:22 wylai@rails-env(192.168.220.137) [~/github/NS/NetworkSecurity/proj2] ± [master]
[XD] % ./common_factorize.py
3 and 8 have common factor:
14656665144589336868890576345676445233783803276368267622102594568299164979334002689085447204937159234673
04541912218503714084065814754185790088811115710921735307483316671075826228613097271501609144807818412051
55449584530166428770678446245420268299373990760393892275516496045323891286171163252445865368303271017
10:23 wylai@rails-env(192.168.220.137) [~/github/NS/NetworkSecurity/proj2] ± [master]
[XD] %
```

Retrieve the private key

- Using gmpy library to calculate the multiple modular inverse and export the private key to a file.
 - In this code, I retrieve the private key using in https3.pcapng

```
1 #!/usr/bin/python
2
3 from Crypto.PublicKey import RSA
4 from fractions import gcd
5 import gmpy
6
7 targets = ['pems/pubkey_3.pem', 'pems/pubkey_8.pem']
8
9 pems = []
10
11 for target in targets:
12     p = open(target).read()
13     pems.append(RSA.importKey(p))
14
15 print ("Get n1:")
16 print (pems[0].n)
17
18 p = gcd(pems[0].n, pems[1].n)
19 print ("Get p: ")
20 print (p)
21
22 q1 = pems[0].n // p
23 print ("Get q1: ")
24 print (q1)
25
26 d1 = long(gmpy.invert(pems[0].e, (p-1) * (q1-1)))
27 print ("Private key d1")
28 print (d1)
29
30 private_key = RSA.construct((pems[0].n, pems[0].e, d1, p, q1))
31 open("private_key_3.pem", 'w').write(private_key.exportKey('PEM'))
32
```


Decryption via wireshark

- Edit -> preferences -> protocol -> SSL
 - Add the private key to here,
 - IP: server ip
 - Port: 443
 - Protocol: http
 - File path: /path/to/private_key.pem

After decryption

2.158319	172.17.0.4	172.17.0.5	TLSv1...	583 Client Hello
2.158329	172.17.0.5	172.17.0.4	TCP	66 443 → 42296 [ACK] Seq=1 Ack=518 W
2.158520	172.17.0.5	172.17.0.4	TLSv1...	981 Server Hello, Certificate, Server
2.158542	172.17.0.4	172.17.0.5	TCP	66 42296 → 443 [ACK] Seq=518 Ack=916
2.169180	172.17.0.4	172.17.0.5	TLSv1...	380 Client Key Exchange, Change Ciper
2.170931	172.17.0.5	172.17.0.4	TLSv1...	304 New Session Ticket, Change Cipher
2.178063	172.17.0.4	172.17.0.5	HTTP	1070 POST / HTTP/1.1
2.179760	172.17.0.5	172.17.0.4	HTTP	405 HTTP/1.1 200 OK (text/plain)
2.216075	172.17.0.4	172.17.0.5	TCP	66 42296 → 443 [ACK] Seq=1836 Ack=14
3.648947	172.17.0.4	151.101.0.133	TCP	54 45534 → 443 [ACK] Seq=1 Ack=1 Win
3.649132	151.101.0.133	172.17.0.4	TCP	54 [TCP ACKed unseen segment] 443 →
4.180555	172.17.0.5	172.17.0.4	TLSv1...	93 Alert (Level: Warning, Descriptio
4.180585	172.17.0.4	172.17.0.5	TCP	66 42296 → 443 [ACK] Seq=1836 Ack=15
4.180618	172.17.0.5	172.17.0.4	TCP	66 443 → 42296 [FIN, ACK] Seq=1520 A

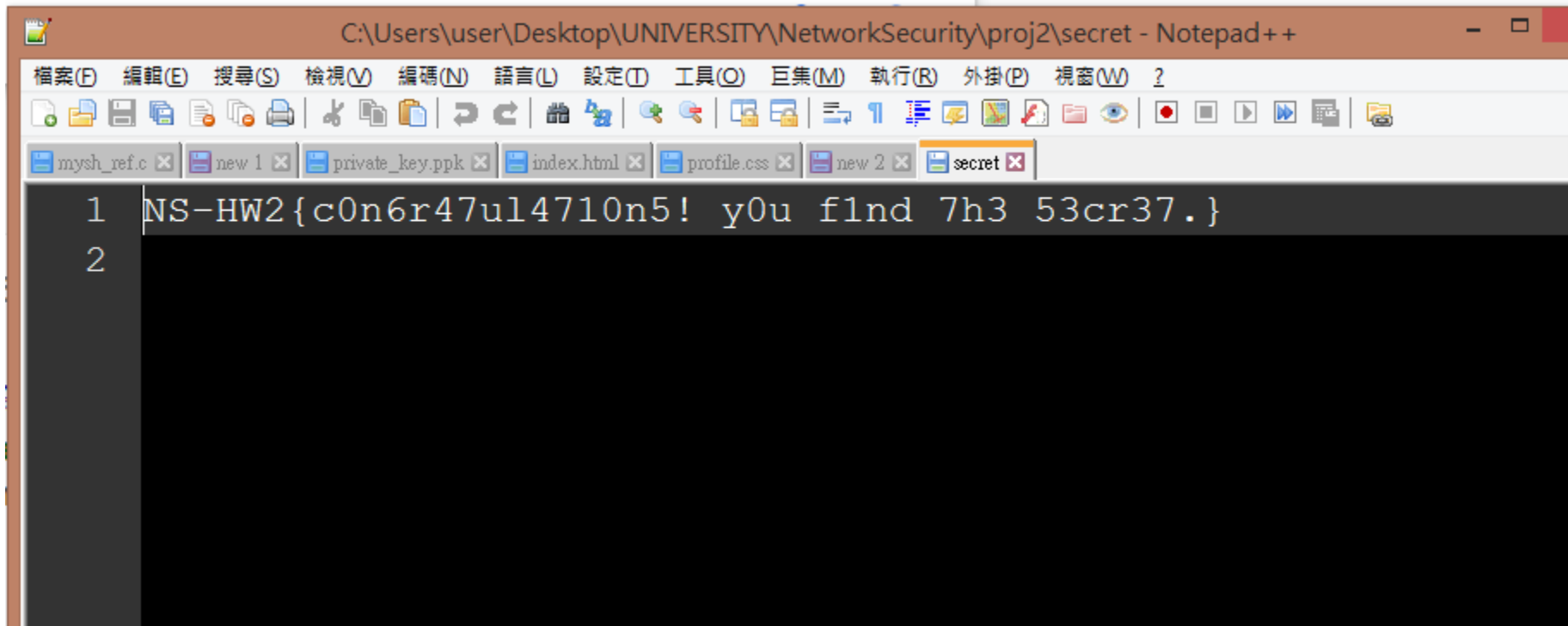
Export secret.zip

- See the data payload section, and export “Data” as “secret.zip” file

```
-----
  Encapsulated multipart part: (application/octet-stream)
    Content-Disposition: form-data; name="file[]"; filename="secret.zip"\r\n
    Content-Type: application/octet-stream\r\n\r\n
  Data (236 bytes)
    Data: 504b03040a0009000000c47b8a4adf08e58b3a0000002e00...
    [Length: 236]
Last boundary: \r\n-----1510992731146319113775032890--\r\n
```

Retrieve the secret

- Decompressed the zip file by the key gain in “websocket”, and get the secret.

A screenshot of a Notepad++ window. The title bar reads "C:\Users\user\Desktop\UNIVERSITY\NetworkSecurity\proj2\secret - Notepad++". The menu bar includes options like 檔案(F), 編輯(E), 搜尋(S), 檢視(V), 編碼(N), 語言(L), 設定(O), 工具(T), 巨集(M), 執行(R), 外掛(P), 視窗(W), and ?. The toolbar contains various icons for file operations. The tab bar shows several open files: mysh_ref.c, new 1, private_key.ppk, index.html, profile.css, new 2, and secret. The main text area has a dark background and shows two lines of text: "1 NS-HW2{c0n6r47u14710n5! y0u f1nd 7h3 53cr37.}" and "2".

```
1 NS-HW2{c0n6r47u14710n5! y0u f1nd 7h3 53cr37.}
2
```

Summary – What I learned

- Python-crypto library
 - We can use this library to help us generate the RSA key pair and load RSA keys.
- Wireshark
 - It will help us to snoop the network traffic and get information.
- RSA common factory vulnerability
 - If the p, q pair chose not randomly enough. Collect enough packets, and the attack will retrieve the private key and crack the secret.