

1. SSL/TLS and Tor

(1) Reference: <https://en.wikipedia.org/wiki/POODLE>

<https://whatis.techtarget.com/definition/POODLE-attack>

<https://www.howtogeek.com/199035/what-is-the-poodle-vulnerability-and-how-can-you-protect-yourself/>

POODLE attack : 利用 POODLE (Padding Oracle On Downgraded Legacy Encryption)的弱點進行攻擊。由於 SSL3.0 的設計上出了問題，讓 CBC mode Padding attack 可以實行。許多 browser 在無法成功進行 TLS connection 時轉換成用 SSL3.0，而這時攻擊者作為中間人，就可以指定 browser 都使用 SSL3.0，實行 man-in-the-middle attack 如此一來，由於 padding attack 最多只要嘗試 256 次就可以 reveal 一個 byte，攻擊者很快就能得到明文。總結來說，這個攻擊結合了 downgrade attack 與 padding attack。

防範方式：client 與 server 都不要去使用 SSL3.0，讓 connection 都是以 TLS 進行。client 可以將瀏覽器移除掉對 SSL3.0 的 support，server 則是可以進行更新以支援 TLS connection。

(2) SSL Stripping attack and Super Cookie

(a) Reference: 上課投影片

SSL Stripping attack 是指，當使用者打了 [http://\[url\]](http://[url]) 來進行搜尋，中間人的攻擊者就可以將 HTTPS “Strips”為 HTTP。如此一來，由於 HTTP 並沒有被加密，攻擊者就可以獲取資訊；而 HTTPS 不會對 client 進行驗證，因此也無法發現是跟 Eve 做互動。而 HSTS 需要 web server 的合作：當 client 第一次連到某 web server 時，該 web server 會告知瀏覽器以後要互動的話，只能使用安全的 HTTPS connection 來進行。如此一來，就能防止其他不安全的格式。



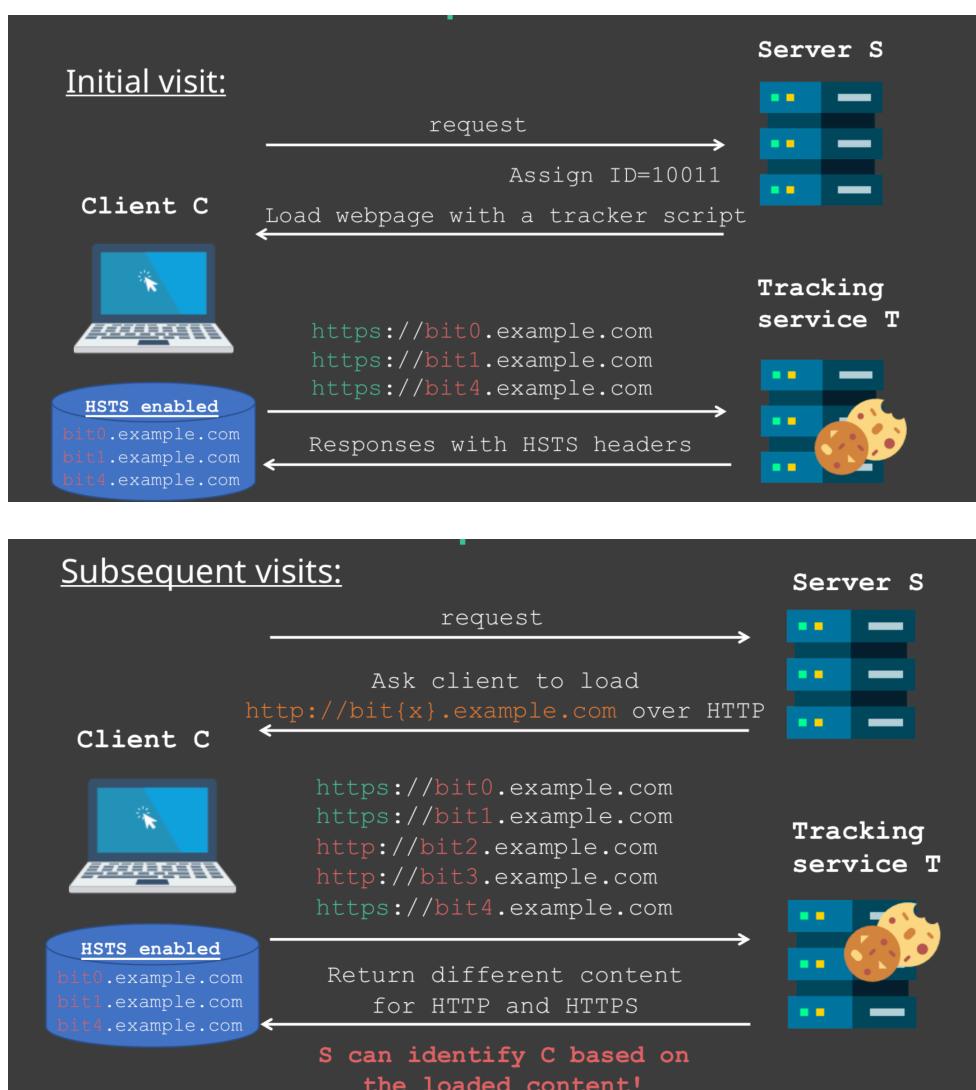
(b) Reference: 上課投影片

<https://security.stackexchange.com/questions/79518/what-are-hsts-super-cookies>

<https://money.cnn.com/2015/01/09/technology/security/super-cookies/index.html>

新造訪一個 Server S 時，如果該 server 想要對 client C 進行 track，可以隨機發送一個 id 給 client C。接著根據該 id 的 bits，只要對應是 1 的就讓 client 以 https 造訪掌控的網域，如 <https://bit{x}.example.com>，x 為 bit 為 1 的位元。如此一來，HSTS 就會紀錄這些網頁，以後連線到這些網頁的方式如果是以 HTTP 連線的話，就會自動先轉成 HTTPS 的方式再進行連線。

在這之後，client C 即使是以 private mode 進行連線，不想讓 server S 知道是 client C 的身份，Server S 可以透過 tracker script，讓 user 對每個位元對應到的網頁進行 HTTP 連線。此時由於之前 HSTS 紀錄過 1 的位元 index，這些連線會被轉成 HTTPS。此時 Server S 就可以藉由比對哪些網頁是用 HTTP 連；哪些用 HTTPS 連，就可以推出 client C 的身份，即 track 了 Client C。



(c) Reference: <https://webkit.org/blog/8146/protecting-against-hsts-abuse/>

由於 Tracker 會於 TLD (Top Level Domain) 之後做一連串的 add-ons，像是 <https://bit{x}.example.com>。因此針對這種操作模式，設定 HSTS 只能對 TLD plus one 或是 loaded hostname 才會紀錄 bit 1。這樣一來，tracker 就會得到一堆的 0 而無法有效的推測出 client 身份。

(3) About Tor

Reference: 上課投影片

<https://blog.torproject.org/run-tor-bridges-defend-open-internet>

- (a) 為了避免攻擊者輕易就知道並掌控 first & last layers，於是 client 挑選一個固定的 relays 集合 (也就是所謂的 Entry Guard Relay) 作為 first hop。只有穩定且可信賴的 relay 才能被選為 Entry Guard Relay。除此之外，clients 會每四到八週將這些 guard relays 進行輪換。如果這些 relays 之中沒有攻擊者所掌控的 relay，那麼 client 就是安全的；反之如果總共有 n 個 relays，其中 attacker 握有 c 個 relays，而 Entry Guard Relay 集合之中有一個是攻擊者的，那麼被成功攻擊的機率就是 c/n 。
- (b) Tor Bridges 是指沒有被公開的 relays，透過這些不公開的 relays，即使公開的 Tor relays 被國家、政府、組織等 block 住時，Tor 的使用者依舊可以透過 Tor 瀏覽器連至網路。Tor Bridges 還有個特色，就是他們會修改其網路封包，讓觀察者很難辨別這些封包是否使用 Tor。而既然是 private 的 relays，要獲取就必須透過比較私下的方式，例如寄信到 bridges@torproject.org 去索取。

2. BGP

Reference: <https://www.noction.com/blog/as-path-and-as-path-prepending>

(1) AS999 只要 announce 一個更精確的 IP address，如：10.10.220.128/24，並宣稱說自己能夠連到 AS1000。這樣一來，根據 BGP longest prefix match 的原則，全部的 AS 就會如圖所示，被導向到 AS999 了。

(2) - a

AS999 跟(1)相比，多加了 AS1, AS2 於 update message 中，即：

{10.10.220.128/24, {AS1, AS2, AS1000}}。這樣一來，根據 Loop prevention，AS1, AS2 就不會走到 AS999。而其他 AS 依然會根據 longest prefix match 原則，將 route path 通過 AS999。

(2) - b

一旦用 Loop prevention 機制，攻擊者就可以讓特定的 ASes 還是正常導向 AS1000；而藉由 path prepending，攻擊者可以讓特定路徑變得不容易被 BGP 選擇，也就是故意多 append 重複的 ASes，讓 BGP 以為通過該路徑要經過比較多個 hop 而盡量不走。

(3) Reference:

https://en.wikipedia.org/wiki/Longest_prefix_match?fbclid=IwAR1oO6l3bHuikUkPQK0m961HJWIC1YPwzVh7XuqBEI_sMCtmzu5WmPJ260s

Advantage:

在 AS 宣告的 prefix length 達到 MPL 的情況下，攻擊者沒辦法用 prefix hijacking 的方式攻擊 BGP，因為沒辦法在宣告更長的 prefix length。

Disadvantage:

一旦 prefix length 有最高限制，有些比較 specific 的網域就無法被連到。例如 MPL 為 24，而有個網域為 x.x.x.x/26，該網域就無法過任何 AS announce 去連至。

3. Mix Network

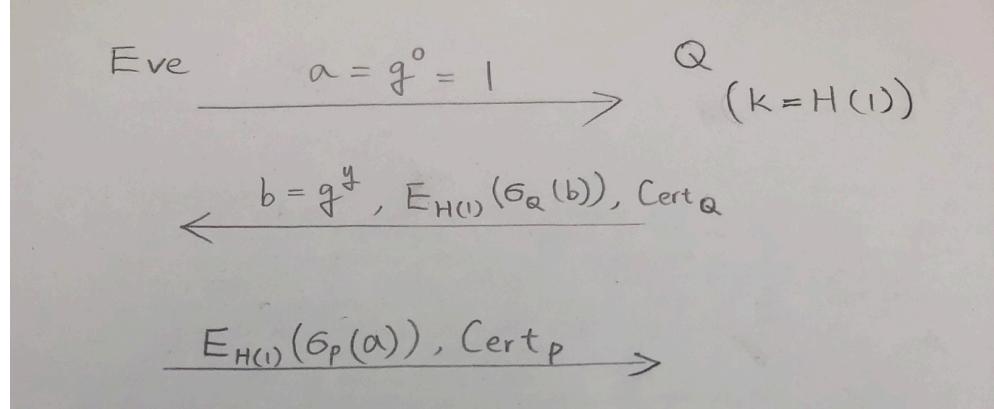
- (1) 觀察 csv 檔，發現說 r_3 、 r_4 都有單獨出現的情況，因此可以確定這兩個都是 s_1 的 recipient。而觀察 r_1 、 r_2 、 r_5 ，只能推測：當 r_2 、 r_5 其中有個不是 s_1 的 recipient 時， r_1 可以確認是 s_1 的 recipient。也就是 r_1 、 r_2 、 r_5 無法確認是否為 s_1 的 recipient，而 r_3 、 r_4 則確定是。
- (2) 假設只有一個 recipient 出現時，可以故意傳送一些 dummy 封包給其他 recipients。例如有 x 個 sender, y 個 recipients, $x > y$ 。那麼就多發送 $x-y$ 個 dummy 封包。這樣一來，攻擊者就沒辦法觀察後去做推斷。

4. STS Protocol

Reference: <https://www.slideshare.net/alagumani1984/diffie-hellman-26604314>

- (1) 在 signature 沒有被加密的情形下，攻擊者(這裡用 E 代表 Eve)可以攔截並分別獲取 g^x, g^y ，同時在 P 傳給 Q $\sigma_P(a, b), Cert_P$ 時將其竄改為 $\sigma_E(a, b), Cert_E$ 。這時 P 會覺得他是在跟 Q 進行通訊，而 Q 却會覺得他在跟 E 進行通訊。而在 signature 加密過的情形下，由於 Eve 沒辦法得到 key k，因此沒辦法做出 $E_k(\sigma_E(a, b))$ ，就無法進行 identity misbinding attack。
- (2) Eve 可以要求 P 對 $\sigma_E(a)$ 做 encryption，這樣一來 Eve 就能得到 $E_k(\sigma_E(a, b))$ 。接著只要如同(1)的方式，將 $\sigma_E(a, b), Cert_E$ 傳給 Q，就能達到類似的攻擊效果。

- (3) Eve 看得到 P 的 public key，因此用 P 的 public key 註冊一個 $Cert_E$ 。在最後 P 傳給 Q $E_k(\sigma_P(a, b)), Cert_p$ 時，將其竄改為 $E_k(\sigma_P(a, b)), Cert_E$ 。這樣一來，P 會覺得是在跟 Q 做 communication，而 Q 會覺得是在跟 Eve 做 communication。
- (4) Eve 要 impersonates 的話，只要如以下圖表就可以達成：



利用這題 RSA 裡面的 hash function 是 identity function，由於 1 不論多少次方都還是 1，而 $H(1)$ 是可以知道的，因此 Eve 就可以成功的達到 impersonate P。

Capture The Flag

5. Timmy & Amy

(1) Flag: CNS{this_is_TIMMinY_attack!!!!}

觀察助教的扣之後發現有 import datetime，且一旦 call 到 check 函式，由於經過大量的運算會花掉許多時間，能看出明顯的時間差。因此首先先嘗試密碼長度，發現長度為 10 的時候會進入到第一次 check。接著每次都修改最前面一位數字，間隔時間一變長就表示該位數字猜對了，直到猜出密碼為 8604263255 為止。

```
# loyichun @ luoyijunde-MacBook-Pro in ~/Desktop/CNS on git:master ✘ [15:07:18]
$ nc cns.csie.org 10224
-----
Timmy: Hi, I'm Timmy, I'm looking for my friend, Amy. If you are Amy, you must know what is my favorite number. - 07:20.29
-----
> 8604263254
You: 8604263254 - 07:22.15
-----
Timmy: No, you are not my friend! - 07:25.10
-----
(base)
# loyichun @ luoyijunde-MacBook-Pro in ~/Desktop/CNS on git:master ✘ [15:07:25]
$ nc cns.csie.org 10224
-----
Timmy: Hi, I'm Timmy, I'm looking for my friend, Amy. If you are Amy, you must know what is my favorite number. - 07:32.71
-----
> 8604263255
You: 8604263255 - 07:36.87
-----
Timmy: Hi~Amy, long time no see! - 07:39.82
-----
CNS{this_is_TIMMinY_attack!!!!}
(base)
```

(2) Flag: CNS{>Be_4ware_0f_the_\$33D<}

Server 會給一組 nonce1，而我們也提供一組 nonce2 給他，server 再傳給我們把 nonce1,nonce2,key hash 過後的傳給我們，要我們回傳 nonce2,nonce1,key 的 hash 值給他。如果出現 nonce1 == nonce2，則判定失敗。觀察 nonce1 的產生方式，發現是使用 time.time() 作為 random 種子的起始。因此，如果可以在極短時間內呼叫該函式兩次，可以預期第一次和第二次的結果是相同的。因此使用 thread 來達到近似平行：第一個 function 用來作為告知 server 提供的 nonce1 為何，第二個 function 則是讓我們可以在預先知道 nonce1 之後，故意將 nonce2 設為 nonce1,nonce1。如此一來，server 回傳給我們的 hash 值就是我們要傳回去的 hash 值，得到第二個 flag。

```
$ python code5.py
[*] Opening connection to cns.csie.org on port 10225: Done
[*] Opening connection to cns.csie.org on port 10225: Done
[*] Switching to interactive mode
target: 1063947887
$ 1063947887,1063947887
You: 1063947887,1063947887 - 42:23.47
-----
Timmy: In order to prevent reflection attack, I hash 2 messages together with the secret key
      This is my message, N_t: 1063947887
      This is the hash, H(N_t,N_a,key): 967ec558935684a4a465e1e2a5db8b123b5c2d6c078aa4652c0c3dd5e0f129bd
      Now it's your turn to prove that you are Amy. Send me the cipher, H(N_a,N_t,key) - 42:23.47
-----
> $ 967ec558935684a4a465e1e2a5db8b123b5c2d6c078aa4652c0c3dd5e0f129bd
You: 967ec558935684a4a465e1e2a5db8b123b5c2d6c078aa4652c0c3dd5e0f129bd - 42:32.14
-----
Timmy: It's you! Amy! I miss you so much! - 42:32.14
-----
CNS{>Be_4ware_0f_the_$33D<}
[*] Got EOF while reading in interactive
```

6. TLS

(1) Reference:

https://www.johndcook.com/blog/2018/10/28/fermat-factoring/?fbclid=IwAR1rM12_WdWTrCqsTyy5vX5buqZoTz-k-8-VCmiC1LbmFF_FPkQEmkFo8WE

<https://www.cryptologie.net/article/340/tls-pre-master-secrets-and-master-secrets/>

<https://scapy.readthedocs.io/en/latest/api/scapy.layers.tls.crypto.prf.html?highlight=master#module-scapy.layers.tls.crypto.prf>

<https://medium.com/@clu1022/那些關於ssl-tls的二三事-九-ssl-communication-31a2a8a888a6>

Flag1: CNS{Ch0O53_CiPH3r_5UIt3S_C4rEfU1Ly}

從 wireshark 擷截到的封包之中，會看到以下這幾個封包：

26	93.369583	192.168.0.108	140.112.30.39	TLSv1...	266 Client Hello
27	93.374113	140.112.30.39	192.168.0.108	TLSv1...	1506 Server Hello
28	93.374115	140.112.30.39	192.168.0.108	TLSv1...	1260 Certificate, Certificate Request, Server Hello Done
29	93.523041	192.168.0.108	140.112.30.39	TLSv1...	1506 Certificate, Client Key Exchange

在 Server Hello 裡面，可以看到 Cipher Suite，而這裡看到的是

TLS_RSA_WITH_AES_128_CBC_SHA。這個 cipher suite 是有機會被破解的：首先分別從封包中拿到 client, server 的 random 以及 RSA 的 n 和 e。根據 Hint1 · $n=p \cdot q$ 的 p 和 q 過於接近，因此嘗試破解該 RSA。假設 $p > q$ 且 $p = a + b$ ·

$q = a - b$ · 則 $n = p \cdot q = a^2 - b^2$ · 移項得到 $\sqrt{n+b^2} = a$, $a \in \mathbb{N}$ ·

接著只要 brute force 去尋找 b · 就可以得到 p, q 了。這樣一來 · 我們就知道 $\varphi(n) = (p - 1) \times (q - 1)$, $ed \equiv 1 \pmod{\varphi(n)}$ · 進而算出 d 值。下一步 · 拿 d 計算出 Pre Master = encrypted_preMaster $\wedge d \pmod{n}$ · 轉成 hex 取後面 48 bits ·

接著準備著手解 TLS 的 PRF :

PRF(pre_master_secret, label, seed) = P_<hash>(pre_master_secret, label + seed)
而 P_hash(pre_master_secret, seed) = HMAC_hash(secret, A(1) + seed) +
HMAC_hash(pre_master_secret, A(2) + seed) +
HMAC_hash(pre_master_secret, A(3) + seed) + ...

其中 A() 定義為 $A(0) = \text{seed}$, $A(x) = \text{HMAC_hash}(\text{pre_master_secret}, A(x-1))$ ·
seed 就是 Server Hello 和 Client Hello 提供的兩個 random ·

從網路上找到一個工具可以幫助計算 PRF 以求出 Master · 再拿 Master 去算出 key block · 就可以從最後 16 bits 拿到 server write 的 key · 最後把這個 key 以及 cipher 的 IV(前 16 bits)拿去解 AES · 就拿到 hex 過的明文了。

```
$ python code6.py
p = 117522482039686371043614782379288857512602022687994559639552472894891756122551814626122376628510951975974521315170987660280328
2267388660386572743680712202765552644818731507488993716593706993074935939092906469903181696887126633806035110981915152401336015073
3360015941665824961062910740035610219642169149224989

q = 117522482039686371043614782379288857512602022687994559639552472894891756122551814626122376628510951975974521315170987660280328
2267388660386572743680712202765552644818731507488993716593706993074935939092906469903181696887126633806035110981915152401336015073
33600159416658249610629107400356002196642169149224521

FLAG1: CNS{Ch0053 CiPH3r SUIt3S C4rEfU1Ly}
```

- (2) 因為有些封包選用的 Cipher Suite 是 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
使用 elliptic curve Diffie-Hellman 作為 key exchange 方式 · 由於計算的困難 ·
使得只要用這個 Cipher Suite 的封包作加密的都無法破解。換言之 · 只要使用如同
FLAG1 的 cipher suite · 都可以如同(1)的方式 · 利用 RSA 的 p, q 過於相近的弱點去破
解得到封包內容 ·

- (3) Reference: <https://terryoy.github.io/2015/02/create-ssl-ca-root-and-self-sign.html?fbclid=IwAR0tHUKO2yIB1uTARu2jCx0mI51r6cUsoSBrfrTVnPP5niyI2mL4HnXX-TQ>

Flag2: CNS{LOSING_CA_PRIVATE_KEY_WILL_BE_A_DISASTER:()}

在 wireshark 第 59 個封包裡面的 encrypted data 比其他封包都還要來得長 · 將其解密之後發現裡面有 RSA PRIVATE KEY · 得到了 CA 的 private key · 接著要偽
造出 CA 憑證 · 才可以連進去 server · 將 RSA PRIVATE KEY 寫入 rootCA.key ·

指令步驟：

- 利用 openssl，先自己簽一個憑證，用 private key 去產生出一個 Root CA。

```
$ openssl req -x509 -new -nodes -key rootCA.key -days 3650 -out rootCA.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:TW
State or Province Name (full name) [Some-State]:Taiwan
Locality Name (eg, city) []:Taipei
Organization Name (eg, company) [Internet Widgits Pty Ltd]:BALSN
Organizational Unit Name (eg, section) []:BALSN
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

- 目標是讓自己的 device 被接受，於是產生 certificate 並且用 Root CA 簽。

(Create device 的 private key)

```
$ openssl genrsa -out device.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
```

(Create the CSR)

```
$ openssl req -new -key device.key -out device.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:TW
State or Province Name (full name) [Some-State]:Taiwan
Locality Name (eg, city) []:Taipei
Organization Name (eg, company) [Internet Widgits Pty Ltd]:BALSN
Organizational Unit Name (eg, section) []:BALSN
Common Name (e.g. server FQDN or YOUR name) []:test
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123
string is too short, it needs to be at least 4 bytes long
A challenge password []:12345678
An optional company name []:.
```

(Create 簽過的 certificate)

```
$ openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out device.crt -days 3650
Signature ok
subject=C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN, CN = test
Getting CA Private Key
```

- 連線到 cns.csie.org:10223，key 就用剛剛產生的 device key，certificate 也是用剛剛創造出的 device crt，就能夠順利過關，得到 flag2。

```
$ openssl s_client -connect cns.csie.org:10223 -key device.key -cert device.crt
CONNECTED(00000005)
depth=1 C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN
verify error:num=19:self signed certificate in certificate chain
verify return:1
depth=1 C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN
verify return:1
depth=0 C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = server
verify return:1
---
Certificate chain
  0 s:C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = server
    i:C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN
  1 s:C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN
    i:C = TW, ST = Taiwan, L = Taipei, O = BALSN, OU = BALSN
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEJjCCAg8IiBF8ss/mKjCGhQ1MMhx6I9P4tMA0GCSqGSIB3DQEBCWUAME8x
CzAJBgNVBAYTA1RXMQ8wDQYDVQQIDAZUYWl3YW4xDzANBgNVBAcMB1RhaXBlaTE0
MAwGA1UECgwwFkFMU04xDjAMBgNVBAAsMBUJBTFNOMB4XDTiMDQwNzE5MTUyMloX
DTIxMDgyME5MTUyM1owUDELMAGA1UEBhMCVFcxDzANBgNVBAgMB1RhaXdhbjEP
MA0GA1UEBwwGVGFpcGvPQ4wDAYDVQQKDAVCQuTTjEPMA0GA1UECwwGc2Vydmy
MIIBIjANBgkqhkiG9wOBAQEAAQCAQ8AMIIBCgKCAQEArJ9aTJcyVZFT1zsMoH61
33gfqg0BcoFF6WZvw/VYjaXeX5AjLZQIZKdv7Urxy0ZvWrmZ3jezfnoHQBHmhhu4
GOD4HDdAmiUCAUoFBuWVYXFQ1iobSwiEwuitYgSn3c+VMZQVQ/2dQ+deA2StSUwz
MB+5TV2ZELHNp2RX0rLLxumSsc51f+GswYanxQaDfsxTRZw8cjG4nw+woY7axuDw
zTi9xGlmvqt3I8WTRia9b91CV7Ho7nmbn5MxqrDrZv4BvsescTGJmtKnTzGPDgG0
K1v6d77AccV9+pRRD6y1ryT4CsZfI4zz0P3RDVho4v6UPCiTRJY05qY0Yi7XZC
6QIDAQABMA0GCSqSIB3DQEBCwUA4ICAQAPbd2hfz0lQcb0fyJlUVQvgbqMChh9
gka0mX81dYGT02Fxln6UDPvo1ZH9u8jHyA3ozJphLZLItc8dL/06Z3sDzxJ8rrb
BlinrAtr2MRFYHTitNbcisFBvKKMDj0ydY5vVkJ95UwCv102a2XbLKenMFawAFxhI
ABxnk85i4E470EDvr+fuw4aNZ2Am0dPxresoi8K0GeyJrjg+N1Q8t1V579YZX4eM
XIB+gNGCBGBnjt/mVN1iuALDJkoG8DC11H0g/ek0joD5Muakx7X/ZJ731KG6qhN
N5umli5JXv80UU3a9YiFbR/7M85ZQ/Gri3zzMqaq9vR1KvcW7EkMxKh71CFI4
Ic+pc3W7z+xv23fdap8p6NQAHML0GM4PriQMU/+Z+Ok/sOYsRExAyf0NO2B/7JA0
7wPcqoAIfhNPws1R0y4kmMM2NR/o1Cmz1LTh/YH32g+uqzovNCYFVq+zC609soN
2w8Ew29tW5FSaPsbXqHWrKw6gJQ7ogoIsdBpePvoDUq9wAToJQNQttP2qpAHR
o2jSnJ0IA2Gx+DaMwp0YdTf+LSpIlsUtc7ozJNvKfT1XAvv7VB1E+LOYHkyKI8Pi
iU496Qpe/UwPGtML10A72u3QPD5nIpIstQiFzXwCnS8PcEEZj5bThBmI6w5Yh+o7
4v5rDKYehJZQgQ==
```

...(中間略)

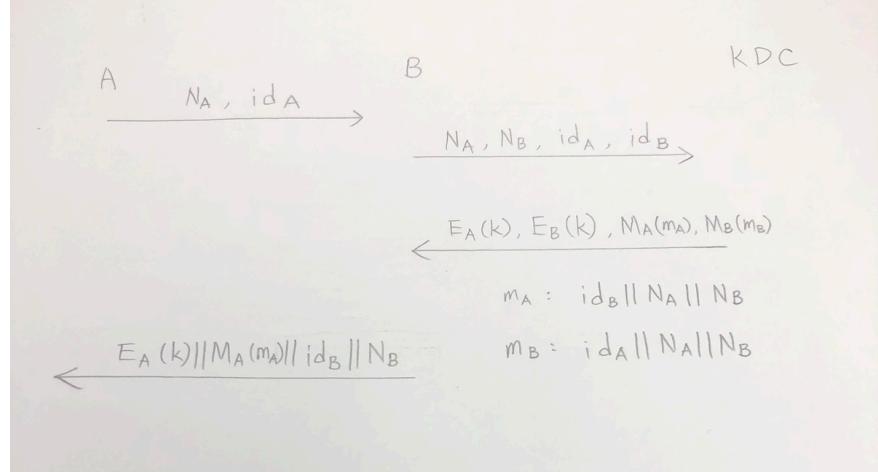
```
0330 - 6d 7b 76 bb 53 b2 10 d4-2c c5 d4 ad 5a 5c c7 9c m{v.5.....Z\..
0340 - 4e 54 63 bd e0 60 76 e1-c7 61 b1 cc 39 ed 43 a9 de NTc..`v. a.9.C..
0350 - 04 3b 4a 59 13 d0 44 f8-f3 3d b0 83 33 4e fc 18 ..;Y..D.=..3N..
0360 - 1d 98 0d 7d db 8b 5b aa-79 b9 cc d7 87 39 c0 3d ...).[~..,9.=
0370 - 1e 62 99 37 fe e3 e8 be-32 5b 34 1e 2b d3 83 5e .b.7...2[4.+.^
0380 - 87 02 a6 ab ea 9a 11-cf d1 72 40 80 f6 de 1c .....@...
0390 - 6a 3d 5a 14 09 f8 e3 e1-a0 1f 4b 8a 42 ff b1 4c j=Z.....K.B..L
03a0 - 78 92 dd f3 07 86 8f 0a-12 9b ce 9a 69 5d 15 10 x.....i...
03b0 - 03 7c a3 91 ce ca 9a 60-64 43 ac bf 7c 9a fe 88 .[...].`dC.[...].
03c0 - c5 30 39 52 93 66 a6 bc-08 3d f8 29 25 ad 2b 9c .09.R.f.=...)%+.
03d0 - 09 de fe 99 de 10 1c 30-bc 37 e8 e3 a3 61 42 ce .....0.7...aB.
03e0 - 85 f6 e2 62 9a 80 b6 44-c7 71 e4 f5 73 9b ae 82 ..b..D.q..q...
03f0 - a8 3c 9d 26 a3 c3 b4 79-5c eb 25 cb 6b 78 40 58 .<...y\%.kx@X
0400 - 05 83 3e 5a cd 1a a3 98-71 e7 ee 1f 9c b1 21 b7 ..>Z...q....!
0410 - ea 97 f3 56 dd f5 5f f8-90 b7 0c eb 9a 3f d0 48 ..V.....?H
0420 - 17 d3 7d 84 9e 57 c7-5c 53 a1 71 36 c6 8a ...)W.S>aU6..
0430 - db f9 81 4e 87 8a 21 03-64 84 cc 09 3b 27 82 4a ..N..!..d...';.J
0440 - 45 ee f6 14 d1 cf c7 4c-2d d3 14 3f f3 b4 92 E.....L..?;N.
0450 - bf c1 e8 78 41 31 ed 8a-6e 6d 52 9f 41 cb a6 8c ...x.1..nmb.A.
0460 - aa d2 e0 3c f4 a8 ae dc-a5 11 2a c0 c8 68 ef 95 ...<....*..h..
0470 - f3 bc 2c 41 ba 70 14 cf-dd 9d e9 67 a7 6f 86 68 ..A.p....g.o.h
0480 - 06 37 f1 8b 95 d6 fa 78-51 c3 90 95 3e 58 1e 66 .7....xQ...>P.f
0490 - 8e 2b 5b 48 c6 7b af 0b-9a a0 19 56 ff e5 a8 70 ..[...].{....V..p
04a0 - 51 e7 4d 2c 6f ff d3 f9-55 53 d1 fd 41 ed 9e 34 Q,M,o...US..A..4
04b0 - 99 e7 71 cb 7a 31 6e-e7 e4 e2 29 31 db bc df ..l..q,zin...)1...
04c0 - bc eb aa 7a 5f dd 0f d4-00 7a d8 cf 53 b2 1e 86 ....z....z..S...
04d0 - 43 f5 bf 62 90 61 7b 97-f9 d5 8e 82 5c 98 a1 b8 C..b.a(...`...
04e0 - d0 69 66 70 0a 21 02 9c-ac 66 7b 8a dd b6 fd 46 ..f..!..f{...F
04f0 - 42 59 9d eb 82 99 df 0e-bf f0 d5 e5 05 08 c7 73 BY.....s...
0500 - e1 66 fb 88 39 d3 75 9c-50 ac 4a 5b 79 8e 88 bc .f..9.u.P.J[y...
```

Start Time: 1588932327
 Timeout : 7200 (sec)
 Verify return code: 19 (self signed certificate in certificate chain)
 Extended master secret: no
 Max Early Data: 0

read R BLOCK
 CNS(LOSING_CA_PRIVATE_KEY_WILL_BE_A_DISASTER:()
 read:errno=0

7. Key Exchange with KDC

(1) As the picture shows:



(2) Flag1: CNS{M4n_1n_Th3_Middl3_4nd_r3pl4y_4tt4ck_t0_K3y_Exp0sur3_Att4ck}

這個 protocol 最大的漏洞，就是 key exchange 得到的 share secret 和 MAC 身份驗證是分開來進行的。首先，register 一個 user，拿到 KE 和 KM。接著跟 Alice 互動，拿到 Alice 的 nonce。再來做兩次 key exchange: 一次是自己和 Alice，紀錄 c_1, c_2 ；另一次是 Bob 跟 Alice，紀錄 t_1 。回到 Alice 的 server，欺騙他說現在互動的身份是 Bob。這邊就會用到剛剛所說的漏洞：傳給 Alice 的訊息是 user 跟 Alice 做 key exchange 的 c_1 以及驗證身份用的 t_1 。Alice 檢查身份時只用到 t_1 ，並沒有檢查 c_1 是否也來自 Bob。因此，現在就能以 Bob 的身份獲取了 skey 加密過的 flag1。由於 skey 是由 c_2 經過 KE 加密而成的，因此用 KE 解密 skey 之後，再將 encrypted secret 用 skey 解密，就成功拿到 flag1 了。

```

$ python code7_1.py
[*] Opening connection to cns.csie.org on port 10221: Done
Na is: 220769386606980296769453241754793673490
[*] Opening connection to cns.csie.org on port 10220: Done
my KE: b6b67a1365fd71777132fdc3c12e9db13163f21de26efff641a74a4eb46401da3
my KM: 403170c23e3bb5d463896c23053a8ce6e556c5465b2cce6600f4b146a95c2805

[*] Opening connection to cns.csie.org on port 10220: Done
share_c1: d7855eb2cc0ff7b37eccac7889c1d293b9754ddce63f7a7b3e2127eae91f6a9949adbc29c289e44b5c9fb1df51ce573f7b539bb04cb17677588dd847
1133dd7b9c8339786f27e32a
share_c2: f52cee63863630de8a07f9a73ec5435c3193c4950e3741338dc41f9693a546a3f04335d1466f25e10154b6db2ec92c9df48efd0fdbc3378c146ac47
0de6f9289c759aa64478808c5

[*] Opening connection to cns.csie.org on port 10220: Done
t1: 1f54e88e4bf8bb93f71427ca3e461ffaca62bc5ddd2cb604c33061ad70245b703

encrypted message: 1be7d974f9823a3c374c2356174a0ca90a1ad81a2b337f219b6300c3bb19623a6790b86e6a894ec4cd6a795fa346769def1ce4bd95be69
a74c1e60faef5f595e02c5c41fb0192

flag1: CNS{M4n_1n_Th3_Middl3_4nd_r3pl4y_4tt4ck_t0_K3y_Exp0sur3_Att4ck}
[*] Closed connection to cns.csie.org port 10220
[*] Closed connection to cns.csie.org port 10220
[*] Closed connection to cns.csie.org port 10220
[*] Closed connection to cns.csie.org port 10221

```

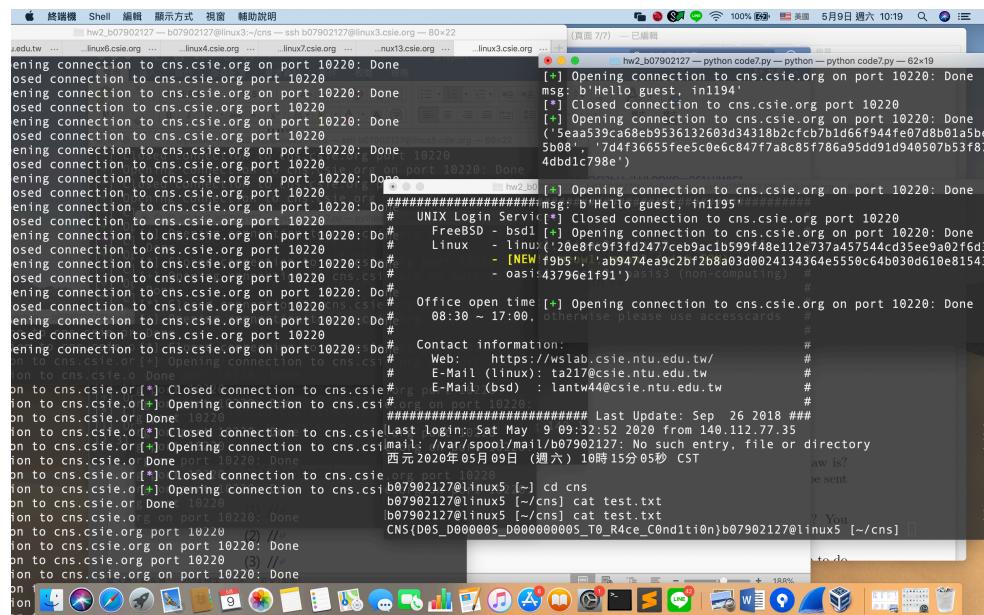
(3) 修補方式是：在驗證身份的同時，同時驗證 share key 是否是同一個身份的。

在 user.py 裡面，可以看到 $d = f'{{__id}}||{_n}||{_n}'$ 。如果在這之後多 append _c 在後面一起做驗證，即 $d = f'{{__id}}||{_n}||{_n}||{_n}||{_c}'$ ，就可以發現說身份明明是 Bob，卻用別的 share key 想矇混過關。

(4) Flag2: CNS{D0S_D00000S_D0000000S_T0_R4ce_C0nd1ti0n}

這個 flag 是透過在 key 洗掉重新註冊 Alice, Bob 以及 Admin 時，透過發送大量的註冊 admin request，搶先在 server 註冊好 admin 前先去得 admin 的權限。這樣一來，就會得到 admin 的 ke, km。接著用 Admin Login，得到 nonce 後回傳 server mac(km, "Admin||nonce")之後，server 會回傳 secret.flag2。接著拿它去做 decrypt，就得到 flag2 了。

我同時七八個工作站跑這支程式，並且額外跑一支程式是不停註冊新帳號的，讓 race condition 成功的機率更大。



The screenshot shows a Mac OS X desktop with a terminal window open. The terminal has several tabs and is displaying multiple ssh sessions to various hosts like 'linux3.csie.org' and 'b07902127@linux3.csie.org'. In the foreground, there's a process monitor tool showing a list of processes with columns for PID, Name, CPU, and Memory usage. One process, 'hw2_b07902127', is highlighted. The terminal output includes messages from the server such as 'Hello guest, in1194', 'msg: b'Hello guest, in1195'', and 'msg: b'Hello guest, in1195''. It also shows system information like 'FreeBSD - bsd1', 'Linux - linu...', and 'Contact information: Web: https://wslab.csie.ntu.edu.tw/ ...'. The terminal window has a blue header bar with the title '終端機' and a status bar at the bottom.