撰寫：資管三 B04705001 陳約廷
日期：2018.04.27
標題：密碼學與資訊安全 作業二

---

1. Super Cookie (15%)

1. HSTS ，HTTP 強制安全傳輸技術。是讓瀏覽器強制使用 HTTPS 與網站通訊，藉此可以減少連線劫持（Session hijacking），避免遭到中間人攻擊。 HTTPS 利用 SSL/TLS 來加密封包，保護交換資料的隱私與完整性。 HSTS 是在 HTTPS response header 中加入 “Strict-Transport-Securrity” 的標頭，強制所有連線都是已 HTTPS 來進行連線。
2. 讓網路伺服器由若干個 sub-domain 所組成。舉例好了，就 20 個 sub-domain。每個都會在標頭中送 HSTS 給使用者，好讓使用者在之後的連線中都使用 HTTPS。在使用者第一次連線時，讓使用者對這 20 個 sub-domain 都送要求，而讓其中幾個要求有 HTST header 。這樣曾經送過 HTST 要求的再下次傳送 HTTP 時就會被轉送成 HTTPS。於是當使用者下一次連線時，讓使用者傳送 20 個 HTTP 要求，曾經被標明 HSTS 的會被轉至 HTTPS。由「是否轉送」而形成的零一陣列就會是一個數字的二進位表示，作為使用者的 ID ，也就可以用來紀錄使用者是否來過。
3. 方法一：限制 HSTS 的設定必須是由 hostname 或是造訪網站的上層 domain 來設定，也就是不能由 sub-domain 來設定。不過如果網站伺服器有很多個 domain，還是可以用同樣的方法來製造使用者 ID。
   方法二：如果某個 domain 的 cookie 已經被阻擋，那就忽略掉這個 domain 的 HSTS 要求。這樣會使得使用者 ID 變成一串 0。但是這樣對這網站的連結會是不安全的 HTTP。不過你應該不會對信任的網站這樣做，所以這個方法可以使陌生網站無法使用 super cookie 來記錄使用者。

---

2. BGP (15%)

1. 在圖片中原本應該被送到 AS 1000 的路線被全部改成送到 AS 999，遭到 BGP hijacking！因為 Route selecting 永遠會挑選比較詳細的 prefix ，也就是較長的 prefix，例如 10.10.220.0/24 。所以 AS 999 宣傳比 AS 1000 更詳細的 prefix，而會使所有原本該通往 AS 1000 的封包全部送到 AS 999。
2. a. AS 999 可以施展 ASPP based prefix interception attack（https://pdfs.semanticscholar.org/b63b/a27c8ab27fc8e7a05fb7389f79f4b1729107.pdf）。 BGP 的路徑選擇是由 path vector algorithm 來決定，也就是會優先選擇較短的路徑。原本 AS 999 在正常 routing 的情況下會選擇網 AS 2 的路徑，應該代表說 AS 1000 在對 AS 4 advertising 的時候有做 AS path prepending。

   所以假設送往 AS 4 的 BGP update 是 { IP prefix, {AS 1000, AS 1000, AS 1000, AS 1000} }。
   則下列為 AS 們原本收到的 update：（底線是 AS 選擇的路徑）
   AS 1：{ IP prefix, {AS 4, AS 1000, AS 1000, AS 1000} }，**{ IP prefix, {AS 1000} }**
   AS 2：{ IP prefix, {AS 1, AS 4, AS 1000, AS 1000, AS 1000, AS 1000} }，**{ IP prefix, {AS 1, AS 1000} }**
   AS 4：**{ IP prefix, {AS 1000, AS 1000, AS 1000} }**，{ IP prefix, {AS 1, AS 1000} }
   AS 3：{ IP prefix, {AS 4, AS 1000, AS 1000, AS 1000, AS 1000} }
   AS 5：{ IP prefix, {AS 4, AS 1000, AS 1000, AS 1000, AS 1000} }
   AS 999：
   { IP prefix, {AS 2, AS 1, AS 1000} }，
   { IP prefix, {AS 3, AS 4, AS 1000, AS 1000, AS 1000, AS 1000} }，
   { IP prefix, {AS 5, AS 4, AS 1000, AS 1000, AS 1000, AS 1000} }

如果 AS 999 惡意向 AS 3 和 AS 5 宣傳錯誤的封包：{ IP prefix, {AS 999, AS 1000} }
AS 4 和 AS 3 和 AS 5 就會選擇較短的路徑而選擇通往 AS 999 的 update。

b. 因為 path prepending 加上了許多重複的 AS 1000 所以 AS 999 可以刪去若干個重複的 AS 1000 使得更新過後的 advertisement 因為更被其他 AS 所接受而使目的地被更新。而 loop prevention 則會使得 AS 1000 無法透過新的 advertisement 把路徑搶回來。

c. 好處是這個攻擊相較於阻斷式攻擊更難被發現，因為封包仍然會抵達 AS 1000。壞處是如果 AS 1000 用 traceroute 的話就會發現資料傳輸的路徑似乎被更動了，而意識到遭到攻擊。

---

3. PIN Authentication (15%)

這個認證機制的壞處是 PIN 碼的可能空間太少了，而且使用 HMAC 作為驗證機制。因此可以利用對 HMAC 的 Length extension 進行暴力搜索，尋找出正確可以通過驗證的 PIN 碼（只要沒有 connection abort 就代表那一個半段的 PIN 碼在 length extension 之後是猜對了）。

而且他還把八位數的 PIN 碼分成兩段來驗證，這樣更縮小了可能空間（從 999999999 變成 9999 * 2），因此就更容易猜中了。

---

4. Can’t bear CBC (15%)

(1) 既然一直都是同樣的 IV …… 而且 CBC mode ，將 plaintext 和 ciphertext 都分成 3 個 block。

```
m0 = "QQ Homework is t"
m1 = "oo hard, how2dec"
m2 = "rypt QQ"
e0 = "296e12d608ad04bd3a10b71b9eef4bb6"
e1 = "ae1d697d1495595a5f5b98e409d7a7c4"
e2 = “37f24e69feb250b347db0877a40085a9"
```

根據 CBC mode，已知……
m1 = D(e1) XOR e0
如果把後兩個 block(e1,e2) 丟進去 decrypt，那麼可以得到 兩個解密後的 block a0,a1……
a0 = D(e1) XOR IV
已知 e0 的情況下可以得到 D(e1) ，再來就可以得到 IV = a0 XOR D(e1) 。

得到 FLAG ！！！ BALSN{IV=KEY=GG}

(2) 這題要用 Padding Oracle Attack 破解。已知 IV 與 ciphertext ，把 IV 轉成 hex encoding 之後與密文串在一起做 POA 就可以得到 FLAG 。

```
time python cbc2_poa.py -c
30714b4c7a793530736b6e7a55354e67630a7f704723502797c9e4d112b2f2ae7232c30dc752bc451d691528b6901f372cd7a4b065f
95e473b9675e321e6e901f4e091dc8403f4d25fec65b55177dc172c122a39c7d597f83c460d8f85a4c6f2 -l 16
[+] Decrypted value (HEX):
42414C534E7B31545F31355F563352595F46554E5F54305F3533335F5448335F464C34475F3450503334525F304E335F42595
F304E335F52314748545F58447D10101010101010101010101010101010
[+] Decrypted value (ASCII):
BALSN{1T_15_V3RY_FUN_T0_533_TH3_FL4G_4PP34R_0N3_BY_0N3_R1GHT_XD}
```

得到 FLAG ！！！
BALSN{1T_15_V3RY_FUN_T0_533_TH3_FL4G_4PP34R_0N3_BY_0N3_R1GHT_XD}

(3) 這題需要進行 POODLE Attack (Padding Oracle On Downgraded Legacy Encryption) 。

5. Man-in-the-middle Attack (15%)

這是 Diffie Hellman key exchange 。不知道 g ，但是 g 的可能空間很小，所以就先來猜測 g 。總共有 3 個回合，開兩個 remote ，然後對想要猜的那個回合對兩邊送出 g 與自己選的次方數，然後剩下的就讓兩個 remote 中回傳的數字互傳，這樣生成出來的 key ，如果是正確的 g 的話，在把猜的那個回合 xor 掉之後應該要是同樣的 key 。於是由這個方法得出三回合的 password 分別是 13, 19, 17 。

在得到 g 之後，與 server 完成一次 Diffie Hellman key exchange 之後再把 key XOR 掉，就可以得到 flag 了～～

得到 FLAG ！！！ BALSN{Wow_you_are_really_in_the_middle}

6. Cloudburst (15%)

先用 nmap 生成 open 的 port，用下列的指令抓出 IP

```
nmap -n 140.112.0.0/16 -p 443 --open -oG - | awk '/Up$/{print $2}' >> open.txt
```

再來比較網站之間的 fingerprint ，要與 https://the-real-reason-to-not-use-sigkill.csie.org:10130/ 的 SSL certificate 一樣，也就是 ”6ee4b82b8a0f9c24a11d22c75b9a8519a5647b76“。
比較之後發現是 140.112.91.250。

```
============================================================
[+] Found matched fingerprint
[+] IP: 140.112.91.250
[+] SHA1: 6ee4b82b8a0f9c24a11d22c75b9a8519a5647b76
============================================================
```

於是連線到 https://140.112.91.250 得到 FLAG。

得到 FLAG ！！！ BALSN{what_a_C1oudPiercer}

7. One-time Wallet (15%)

Python 的 random 函數 PRNG，也就是這並不是真正的隨機，只要獲得當下的 seed 或 entropy，就可以生成出同樣的亂數序列。 Python 所使用的 PRNG 是 Mersenne Twister ，於是把已知的序列轉換成生成的數字，然後推算出原本的 entropy ，就可以產生 Wallet 101 的密碼了！

得到 FLAG ！！！ BALSN{R3tir3_4t_tw3nty}

8. TLS Certificate (15%)

要自己簽憑證，首先要提出 signing request，對照 https://www.csie.ntu.edu.tw 中的憑證資料填入。

```
openssl req -new -key rootCA.key -out power_ranger.csr
```

再來簽憑證……

```
openssl x509 -req -in power_ranger.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out
power_ranger.crt -days 500 -sha256
```

然後寫個 python script 來上傳 certificate 的 base64 encoding ，就可以了～～

得到 FLAG ！！！ BALSN{t1s_ChAiN_0f_7ru5t}