1. Super Cookie
    1. HSTS 令 web server 能詔告天下所有與之連線的 browser 強迫僅能使用 https 進行連線。第一種運作方式為 Trust-On-The-First-Use，在 https 連線時將 HSTS 資訊放在 response header 中，但須要假設第一次連線會是沒問題的，而接下來的連線皆僅會使用 https。第二種運作方式為使用 preload list，因為預載在瀏覽器中，會在送出前進行 Internal Redirect 將連線升級為 TLS，因此 first request 也可被保護到，但 list 的更新就會相對較為困難。若網站有啟用 HSTS 便無法使用 http 進行連線，且無法 bypass。
    2. 利用 HSTS 的特性，各瀏覽器所會記住的資訊會有所不同，因此可以知道使用者是否為同一人、連過哪些網站來達到 across multiple browsing sessions on the same website 的追蹤。在 client 第一次連線 request 的時候，會執行於 server(S)上的 tracker script 將 ID 識別碼回傳給 client，要求 client 送出 requests 至 tracking service(T)去，以 activate 相對應的 bit，接著再回傳 HSTS header 給 client，而 client 便會儲存有 enable HSTS 的連線。待下回 client 連線時，S 會要求 client 對 T 的相關網站利用 http 進行連線，而 client 會對有 enable HSTS 的網站進行 https 連線，其他則否，因此便可透過不同的 http、https 組合來得知 client 的 unique ID 進而知道不同 user。
    3. 在防禦方面，因許多 tracking service 很多很長且很多不同層級的域名來進行追蹤，因此可將 HSTS 狀態限制為主機名或頂級域名+1，防止 tracking service 在大量不同的位址上設置 HSTS。此外，也可設置 redirect 的上限，因此可限制 bits 可被限制的上限。在另一方面，也可忽略第三方(非 origin website)網址的 HSTS 升級請求，而仍僅使用原始網址，令 tracking service 進會獲得全部皆 0 的 ID。而 client 若欲保護自己，也可透過清理 cookie 一併將所有儲存於瀏覽器的 HSTS database 清空。
2. BGP
    1. 因 BGP 較會選擇比較 specific prefix 的路徑，因此若攻擊者若宣告 AS999 為 10.10.220.0/24 便可將流量皆導向至 AS999
    2.
        i. {10.10.220.0/23, {AS 2, AS 1, AS 1000}}
        ii. 因為 loop prevention 的緣故，故當 attacker 在 announce 自己 prefix 且 prepend AS2、AS1、AS1000 至 ASPATH 時，在路徑上的 AS2、AS1 會忽略此訊息，然而其他 AS 則會受到影響，而先送到 attacker 去，才會再透過特定路徑送至 Target。
        iii. 以攻擊者優勢而言，相較於傳統 BGP hijacking，因封包最終仍會達到目的地，甚至透過修改 TTL 而使看起來較為正常，較不容易

被發覺。在劣勢方面，因使用者仍可透過 traceroute 等方式，得知路由路徑方向，若有異狀仍可能會被察覺，此外因延遲時間可能會叫長，也較易令人起疑。

3. PIN Authentication
    1. 因為 PIN 的 size 並不算太大，而當一但嘗試為錯誤的 PIN1，伺服器便會立刻斷線，也就能知道猜錯了，直到嘗試到正確的 PIN1 就可以成功得到伺服器回傳的 RS1；接下來便可繼續嘗試 PIN2 的部分直到伺服器能成功回傳 RS2 為止。為一種藉由 oracle 達成的 PIN recover attack。
4. CBC
    1.
    - 首先將 cipher 拆成三個 block(c1、c2、c3)，將後面 64 個 bit 的 hex 值上傳至 server 解密，可得到前半部為 decrypt(c2) XOR IV 的值
    - 將明文也切成三塊(p1、p2、p3)，將 p2 與 c1 進行 XOR，可得 decrypt(c2)的值
    - 再將該值與 decrypt(c2) XOR IV 的值進行 XOR 可得 flag
    - **BALSN{IV=KEY=GG}**

    2.
        i. 應為利用 padding oracle attack 暴搜方式，轉成 int 再用^來 xor，後面先解出 padding，先戳 padding 格式(b"\x01")，要解最後一位應該是從倒數第二個 block 的最後一位開始改掉來猜

    3.
        i. 僅知道應使用 poodle attack 並與 padding 機制相關
5. MitM
    - 伺服器具有三個 size 很小的數，對每個 password 會利用簡單的 D-H key exchange 來達成，然而 generator 會依據 password 來生成(SPEKE protocol)，而所有密碼的金鑰都會被 XOR 在一起
    - 需要對伺服器跟伺服器自己進行中間人攻擊，因伺服器是唯一知道密碼並且能以有意義的方式傳遞協議者。因此針對每個 password 可利用兩個連線來 exchange g^a mod p 跟 g^b mod p，將可獲得每個密碼相同的 secret 以及相同的最終 master key
    - 若猜對 password 便可計算 generator (g mod p)來執行 MitM 攻擊。若密碼猜測正確便可以將最後的 shared key XOR 掉，來驗證新的 shared key 是否一致。因此便可分別暴力破解各個 password。
6. Cloudburst
    - 先利用 nmap 掃描 140.112.31.96 看開啟 https 的 port 為 443
    - 利用 zmap 掃描 https 的 443 port: zmap –B 10M –p 443 –n 60000 –o result.txt
    - 再利用 curl 收集各 IP 回傳資訊，尋找具有 BALSN 字元 curl –k

https://”$IP”

- 可尋找出真實 IP: https://140.112.91.250
- **BALSN{what_a_C1oudPiercer}**

7. One-time wallet
   - 先從 serer 接收全部資料 將 address 拆分為三個 state 將 password 拆分成五個 state
   - 取出最後 78 輪，共可獲得 624 個 state 轉換為十進位，利用工具（https://github.com/x64x6a/MersenneTwister ）將自己的 random state 設定與 server 同步
   - predict 出接下來會呈現的 password 五個 state 轉換為 hex 格式回傳給伺服器可得 flag
   - **BALSN{R3tir3_4t_tw3nty}**
8. TLS certificate (相關檔案 code8.ipynb、my2.crt、mydomain.csr)
   - 首先，先到 csie 網站下載憑證，將 certificate 讀入(load_certificate)，並將所有相關資料讀出來(cc.C, cc.ST , cc.L ,cc.O,cc.OU,cc.CN)
   - 接著利用 ubuntu openssl 套件，創建 signing request: openssl req -new –key rootCA.key -out my.csr，將所需資訊依照上一步輸出填入
   - 在透過 openssl 產生 certificate，藉由 my.csr, rootCA.key, rootCA.crt: openssl x509 -req -in my.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out my2.crt
   - 將 my2.crt 內容透過 base64 加密，回傳給 server 可得 flag
   - **BALSN{t1s_ChAiN_0f_7ru5t}**