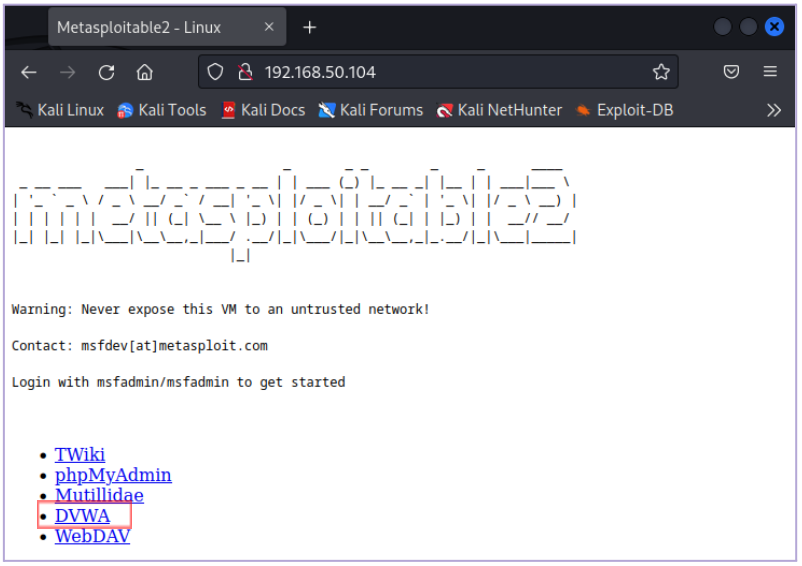
EXPLOIT DVWA - XSS E SQL INJECTION

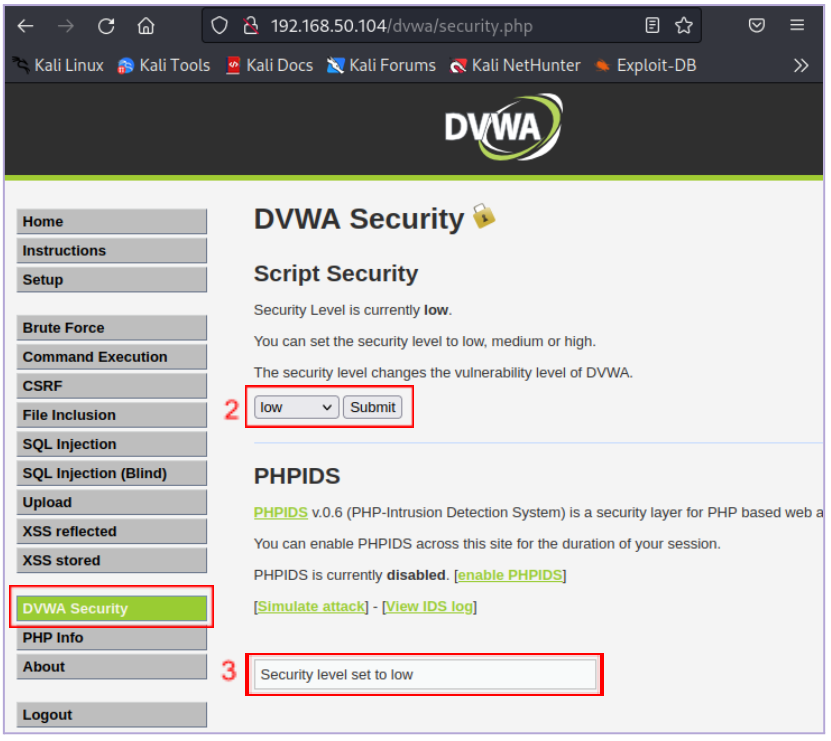
ATTACCHI ALLE WEB APP(3)

PREREQUISITI

- Raggiungere la pagina DVWA



- Settare il livello di sicurezza a LOW

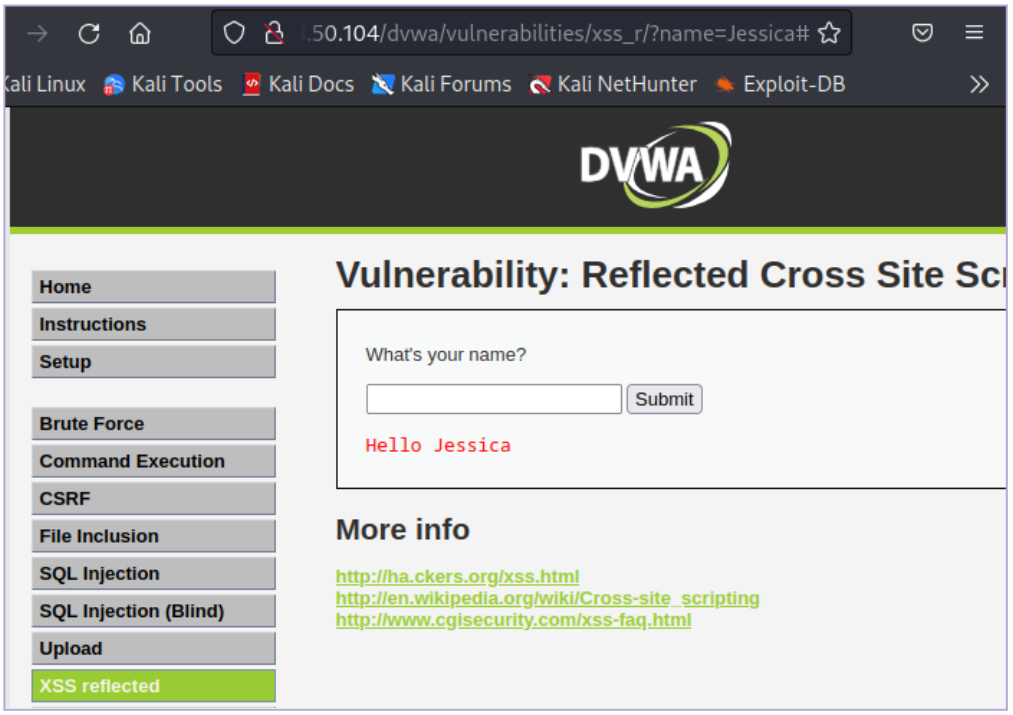


XSS

Cross Site Scripting

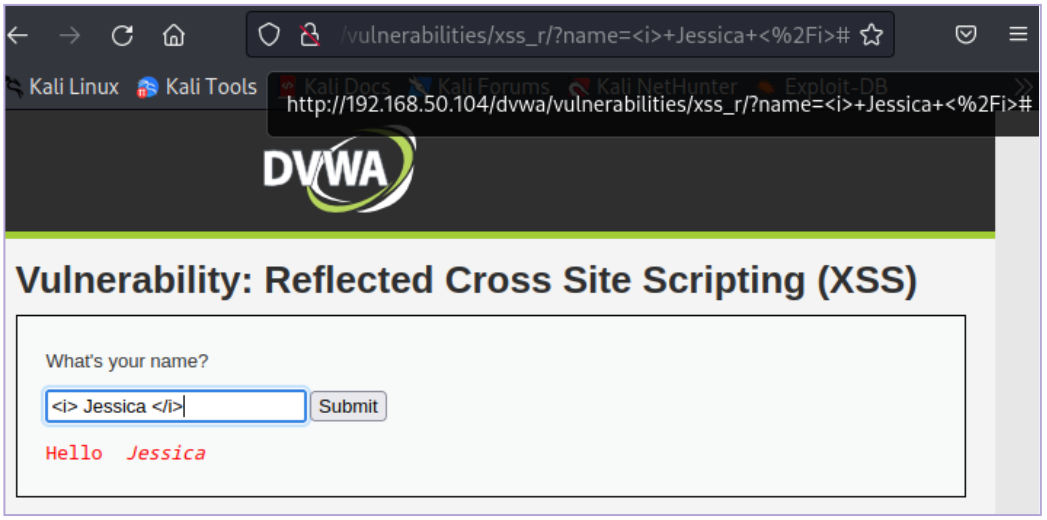
1. Esempio base di XSS reflected

Gli attacchi cross site scripting implicano l'iniezione di codice malevolo nell'output di una pagina Web. Questo codice viene poi eseguito dal browser degli utenti che visiteranno il sito.



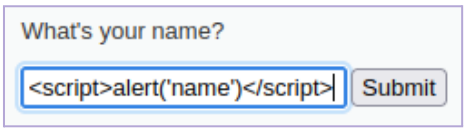
a. i (corsivo di HTML)

Possiamo utilizzare tag semplici che non arrecano danno, come <i>, che eseguito ci restituisce in output una stringa in corsivo. Ciò significa che il tag con la stringa in input vengono eseguiti e riflessi in output. In senso tecnico si dice che il *codice viene interpretato*

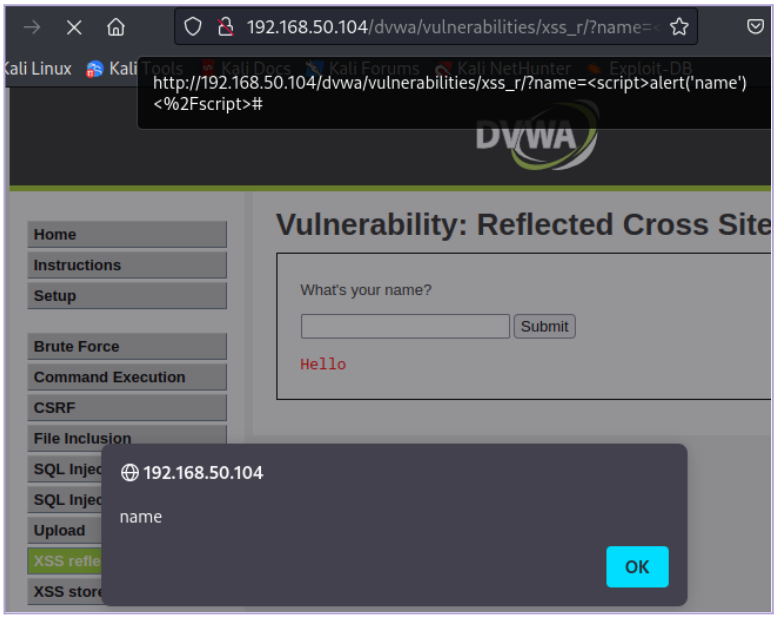


b. alert (di javascript)

Per testare l'XSS possiamo anche inviare un codice `<script> alert('...')</script>` che, se eseguito correttamente, dovrebbe darci in output una finestra pop-up con la dicitura che abbiamo riportato. Questo avviene perché i parametri che abbiamo passato al campo ricerca vengono eseguiti e successivamente utilizzati come output della pagina nel punto di riflessione.



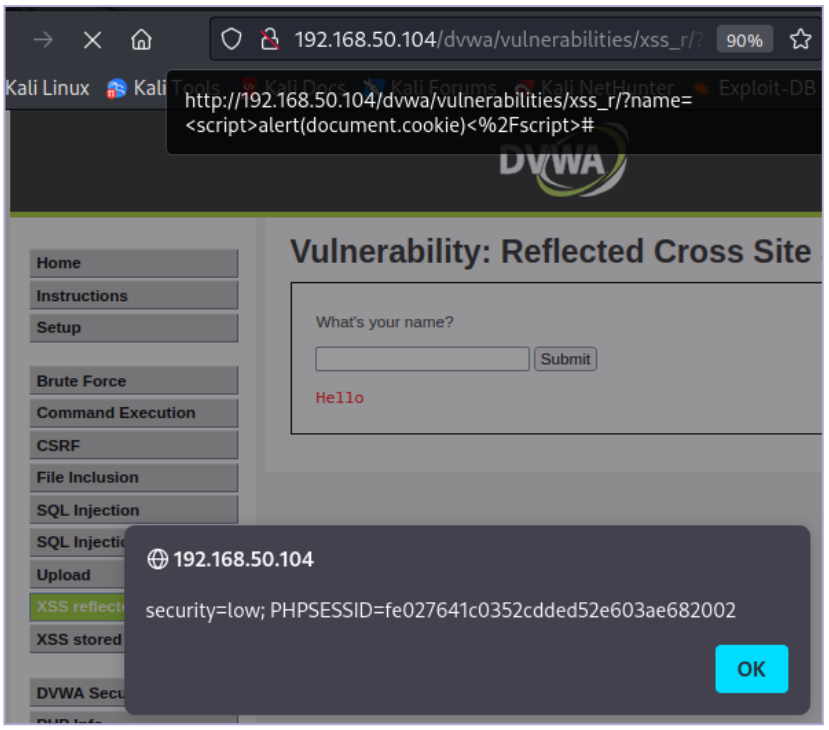

```
<pre>Hello <script>alert('name')</script></pre>
```



2. Recupero Cookie

Gli attaccanti sfruttano le vulnerabilità di XSS per attaccare gli utenti di un sito in diversi modi, tra questi c'è il recupero dei cookie di sessione, avendo quindi modo di impersonificare gli utenti legittimi.

```
<pre>Hello <script>alert(document.cookie) </script></pre>
```



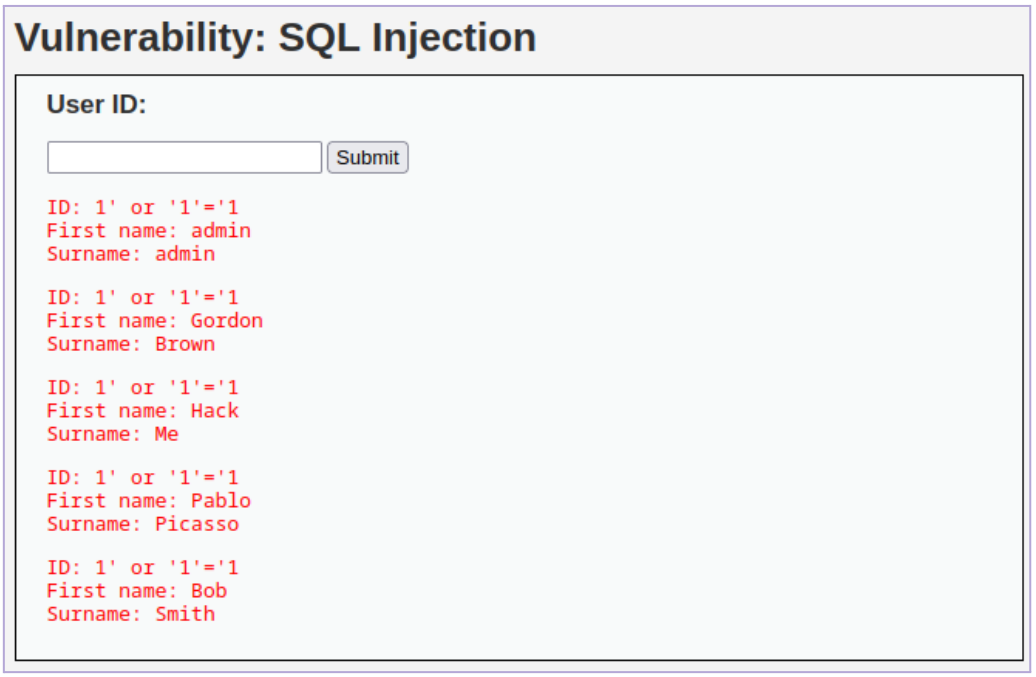
SQL
Structured query language

1. Controllo di injection

La prima cosa che bisogna fare è identificare un injection point (punto di iniezione) dove poi si può costruire il payload per modificare una query dinamica. Contestualmente bisogna testare l'input utente, odvvero provare ad iniettare:

- Terminatori stringa come <'> e <">
- Comandi SQL come SELECT ed UNION
- Commenti SQL
- Operatori logici

Provare una delle opzioni sopra per volta, altrimenti non capiremo quale dei seguenti vettori ha avuto successo



In-band SQL (SQL Classico)

E' l'attacco SQL Injection più comune e facile da sfruttare. L'iniezione SQL in banda si verifica quando un utente malintenzionato è in grado di utilizzare lo stesso canale di comunicazione sia per lanciare l'attacco che per raccogliere i risultati. Le due tipologie più utilizzate di In-band SQL sono:

I) Error-Based SQL

E' una tecnica SQL Injection in banda che si basa sui messaggi di errore generati dal server del database per ottenere informazioni sulla struttura del database. In alcuni casi, la sola SQL injection basata sugli errori è sufficiente affinché un utente malintenzionato possa enumerare un intero database. Sebbene gli errori siano molto utili durante la fase di sviluppo di un'applicazione Web, dovrebbero essere disabilitati su un sito live o invece registrati in un file con accesso limitato.

II) Union-based SQL

E' una tecnica di iniezione SQL in banda che sfrutta l'operatore UNION SQL per combinare i risultati di due o più istruzioni SELECT in un unico risultato che viene quindi restituito come parte della risposta HTTP.

Inferential SQL (Blind SQL)

A differenza di In-band SQL, può richiedere più tempo per essere sfruttata da un utente malintenzionato, tuttavia, è altrettanto pericolosa di qualsiasi altra forma di iniezione SQL. In un attacco Inferential SQL, nessun dato viene effettivamente trasferito tramite l'applicazione Web e l'utente malintenzionato non sarebbe in grado di vedere il risultato di un attacco In-band (motivo per cui tali attacchi sono comunemente indicati come "attacchi SQL injection ciechi"). Un utente malintenzionato è invece in grado di ricostruire la struttura del database inviando payload, osservando la risposta dell'applicazione Web e il comportamento risultante del server di database.
I due tipi di SQL injection inferenziale sono SQLi basato su Blind-boolean-based e SQL basato su Blind-time.

I) Boolean-Based SQL injection

L'iniezione SQL basata su Boolean è una tecnica di SQL injection inferenziale che si basa sull'invio di una query SQL al database che forza l'applicazione a restituire un risultato diverso a seconda che la query restituisca un risultato TRUE o FALSE. A seconda del risultato, il contenuto all'interno della risposta HTTP cambierà o rimarrà lo stesso. Ciò consente a un utente malintenzionato di dedurre se il payload utilizzato ha restituito true o false, anche se non vengono restituiti dati dal database. Questo attacco è in genere lento (soprattutto su database di grandi dimensioni) poiché un utente malintenzionato dovrebbe enumerare un database, carattere per carattere.

I) Time-Based SQL injection

L'iniezione SQL basata sul tempo è una tecnica di iniezione SQL inferenziale che si basa sull'invio di una query SQL al database che forza il database ad attendere un determinato periodo di tempo (in secondi) prima di rispondere. Il tempo di risposta indicherà all'utente malintenzionato se il risultato della query è VERO o FALSO.
A seconda del risultato, una risposta HTTP verrà restituita con un ritardo o immediatamente restituita. Ciò consente a un utente malintenzionato di dedurre se il payload utilizzato ha restituito true o false, anche se non vengono restituiti dati dal database. Questo attacco è in genere lento (soprattutto su database di grandi dimensioni) poiché un utente malintenzionato dovrebbe enumerare un carattere del database per carattere.

2. Union

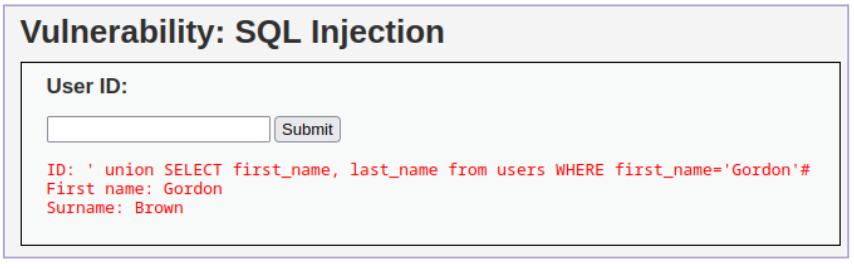
Se il nostro payload fa in modo che il risultato della query originale sia vuoto, possiamo visualizzare il risultato di un secondo query tramite il comando UNION.

- La query originale viene terminata con l'operatore <'> prima di aggiungere una seconda query con UNION scelta da noi.
- I commenti alla fine del comando fanno in modo che la parte successiva della query non venga eseguita dal database.

Il comando UNION, esegue un'unione tra due risultati per esempio tra due SELECT

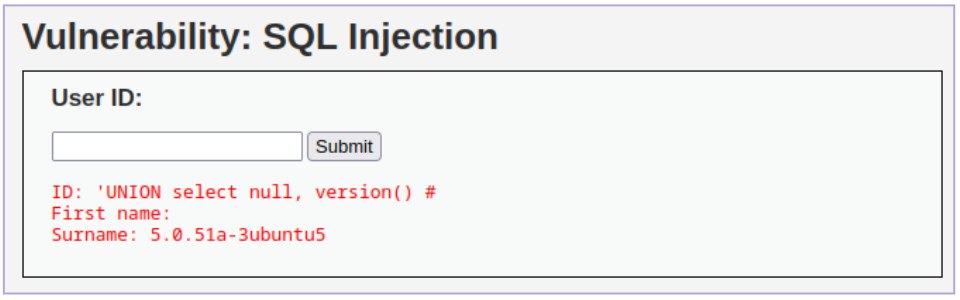
❖ Per individuare e selezionare l'utente che ha come nome iniziale 'Gordon'

```
payload: 'UNION SELECT first_name, last_name FROM users WHERE first_name='Gordon'#
```



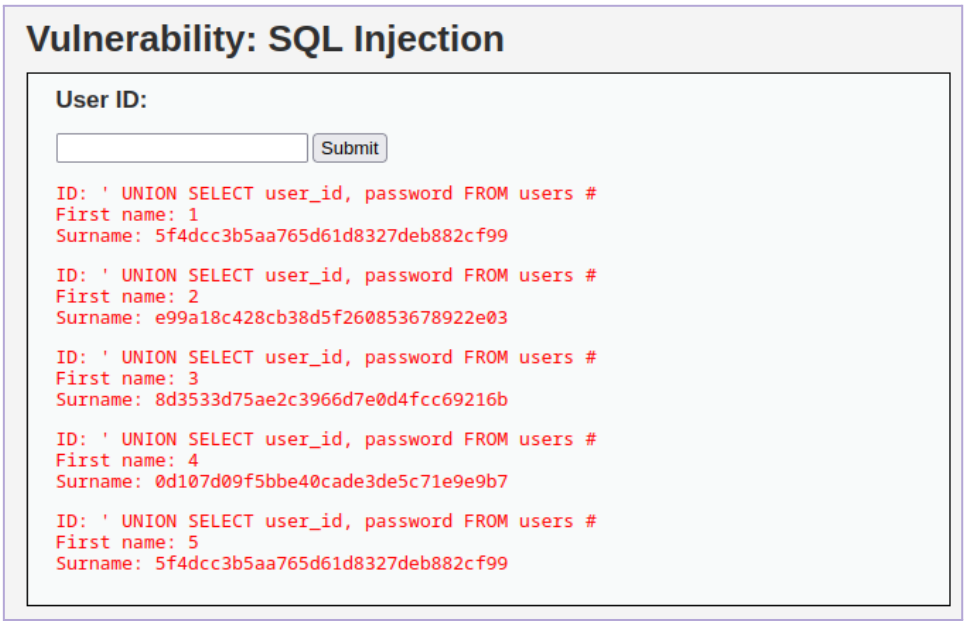
❖ Per capire su quale è la versione del database sulla quale è in esecuzione il DVWA

```
payload:'UNION SELECT null, version() #
```



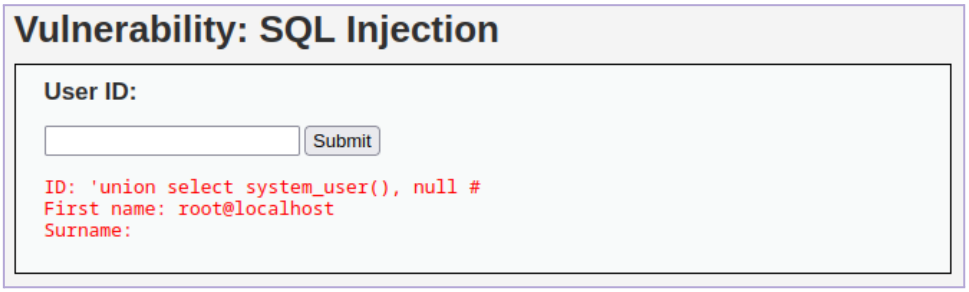
- Per individuare la password associata all'utente

  payload: `'UNION SELECT user_id, password FROM users #`



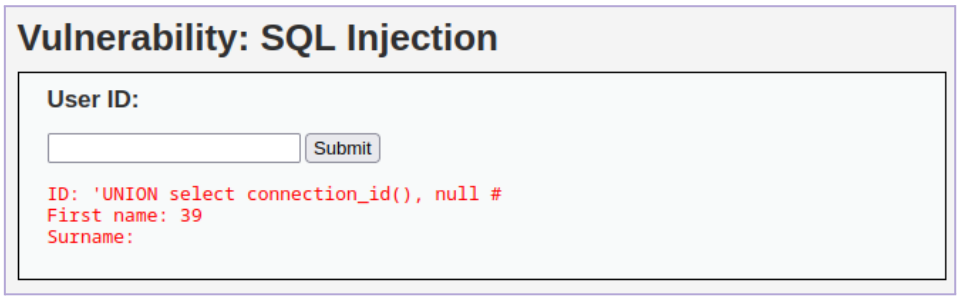
- Per individuare l'utente del sistema

  payload: `'UNION SELECT system_user(), null #`



- Per individuare la porta di connessione

  payload: `'UNION SELECT connection_id(), null #`



- Per visualizzare il nome del database

  payload: `'UNION SELECT database(), null#`