EPICODE-CS0124
S6/L2 - Pratica

Flaviano Sedici

Pratica

Configurate il vostro laboratorio virtuale per raggiungere la DVWA dalla macchina Kali Linux (l'attaccante). Assicuratevi che ci sia comunicazione tra le due macchine con il comando ping.

Raggiungete la DVWA e settate il livello di sicurezza a «LOW». Scegliete una delle vulnerabilità XSS ed una delle vulnerabilità SQL injection: **lo scopo del laboratorio è sfruttare con successo le vulnerabilità con le tecniche viste nella lezione teorica.**

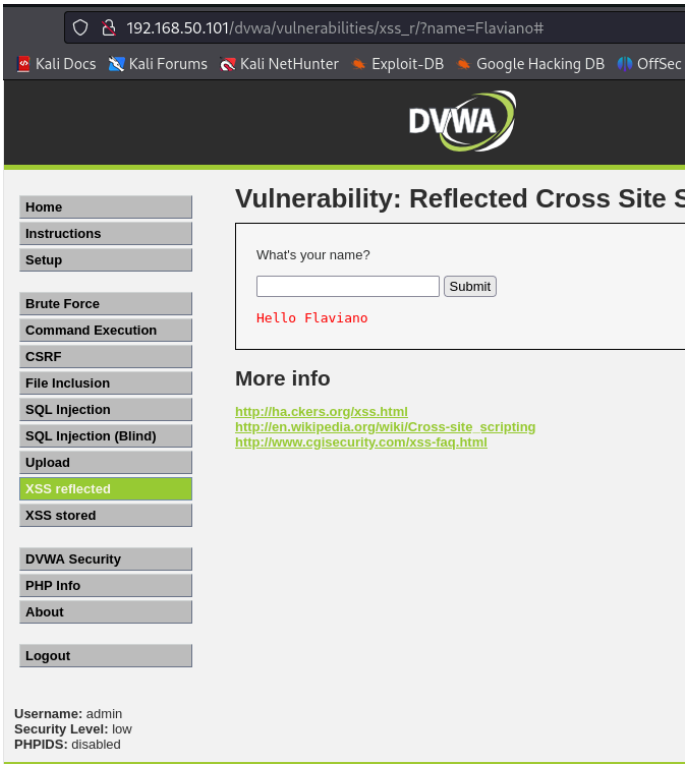
La soluzione riporta l'approccio utilizzato per le seguenti vulnerabilità:

- XSS reflected.
- SQL Injection (**non blind**).

1. XSS Reflected

1.1. Test della funzionalità base

Inizialmente abbiamo testato la funzionalità proposta da DVWA assecondando il suo funzionamento corretto e inserendo un nome:



Test con l'inserimento con Flaviano

Vediamo come l'attributo "name" nell'url è quello che regola il contenuto della pagina web in corrispondenza di dove viene riportato il nome inserito. Proseguiamo con l'analisi.

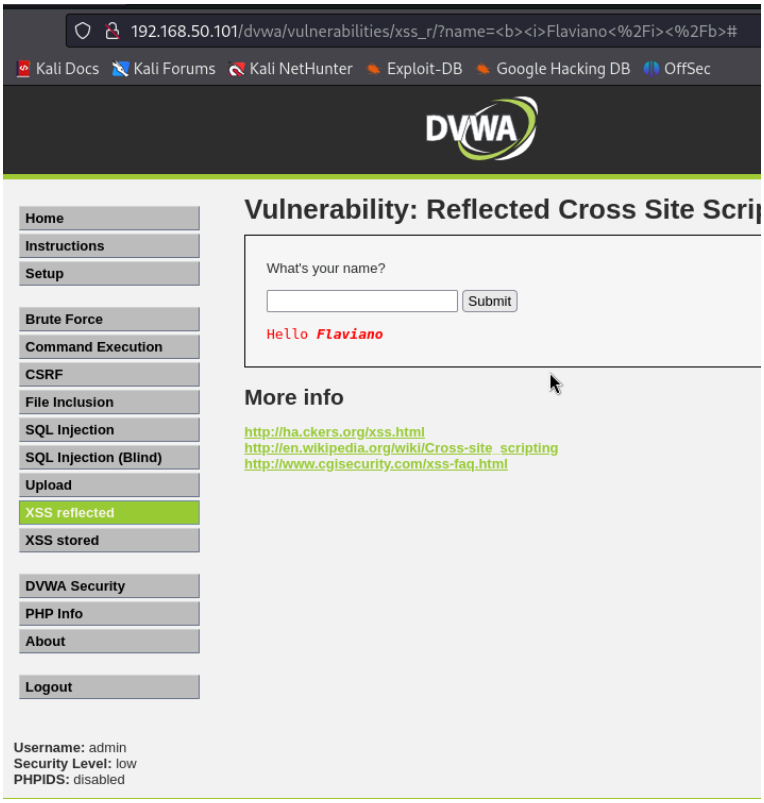
1.2. Inserimento codice html semplice

Per verificare l'effettiva vulnerabilità del campo input, abbiamo inserito il codice:

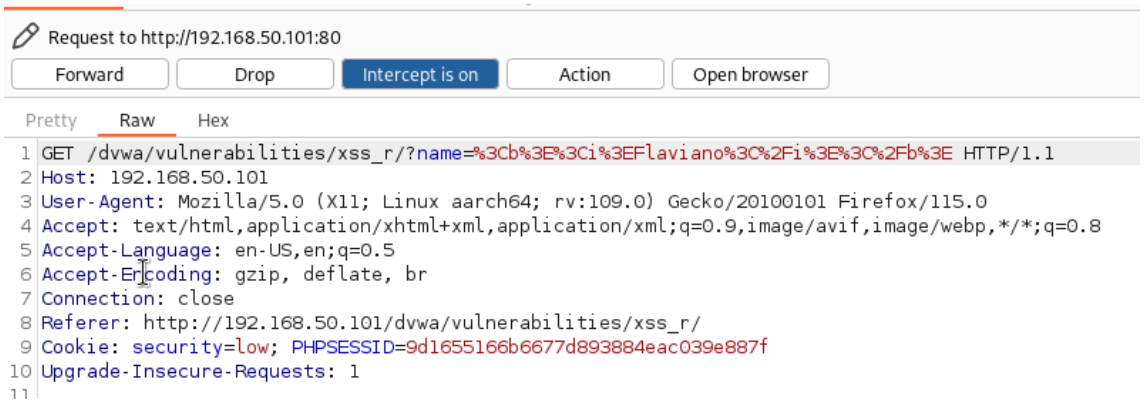
Flaviano

Il quale ci aspettiamo restituisca il nome in bold e italic.

Come previsto la vulnerabilità risulta verificata in quando come si evince dalle immagini il sistema non gestisce correttamente i tag inseriti e quindi vengono interpretati dal browser come tali.



Risultato inserimento codice html



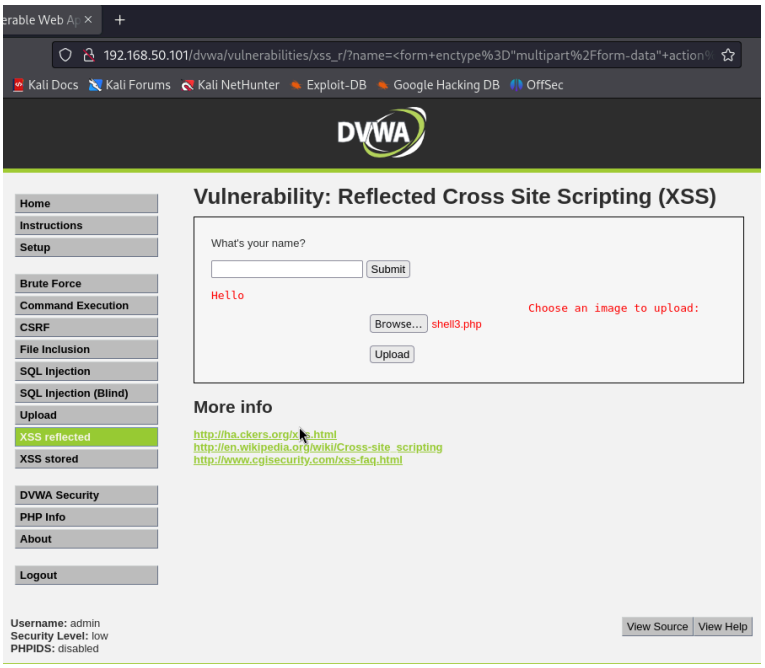
Dati intercettati da BurpSuite

1.3. Inserimento codice html avanzato

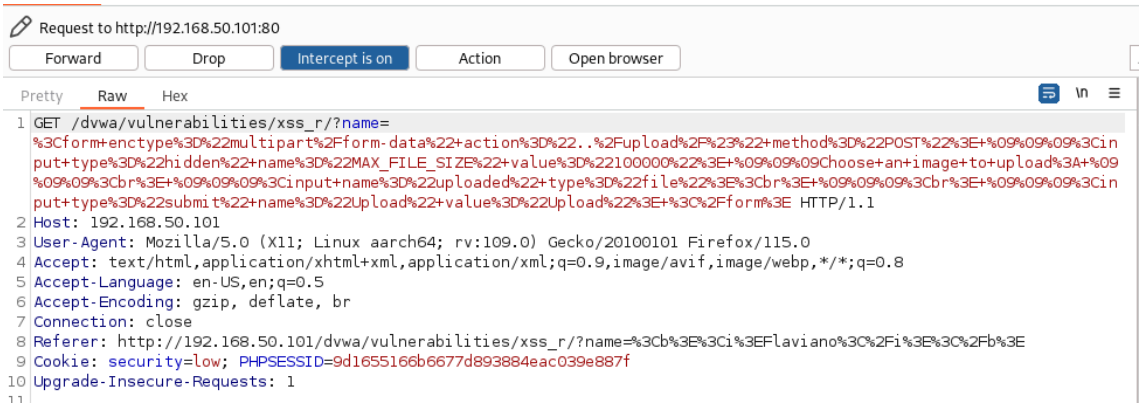
Volendo estendere ulteriormente la ricerca di vulnerabilità, abbiamo inserito del codice avanzato che consente il caricamento di un file tramite la pagina di upload di DVWA pur rimanendo nella pagina dedicata al XSS in modo da eludere eventualmente le restrizioni autorizzative sulla pagina di upload implementate in modo non corretto (ovvero che inibiscono l'accesso diretto al link che magari è semplicemente non visibile all'utente da interfaccia ma non alle sue funzionalità).

```
<form enctype="multipart/form-data" action="../upload/#" method="POST">
        <input type="hidden" name="MAX_FILE_SIZE" value="100000">
        Choose an image to upload:
        <br>
        <input name="uploaded" type="file"><br>
        <br>
        <input type="submit" name="Upload" value="Upload">
</form>
```

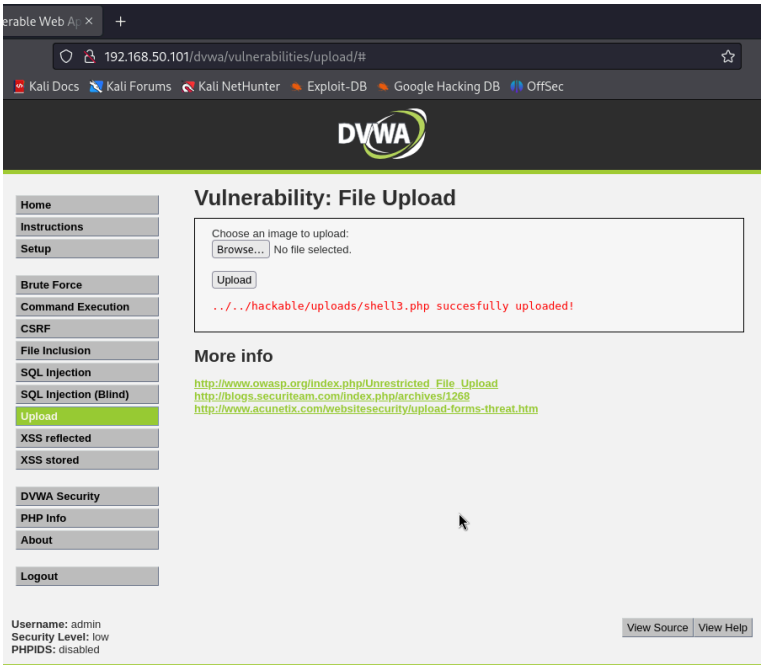
Il risultato del submit è quello di far comparire a schermo la stessa interfaccia di upload presente nella pagina originale, inviando il file caricato alla pagina per l'acquisizione e il caricamento. Utilizzeremo un file dal nome shell3.php che contiene del codice malevolo.



Risultato inserimento codice html



Dati intercettati da BurpSuite



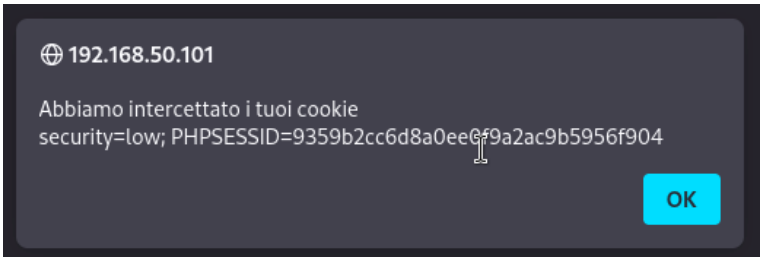
Risultato del file caricato correttamente

Il file risulta caricato correttamente nella sezione upload.

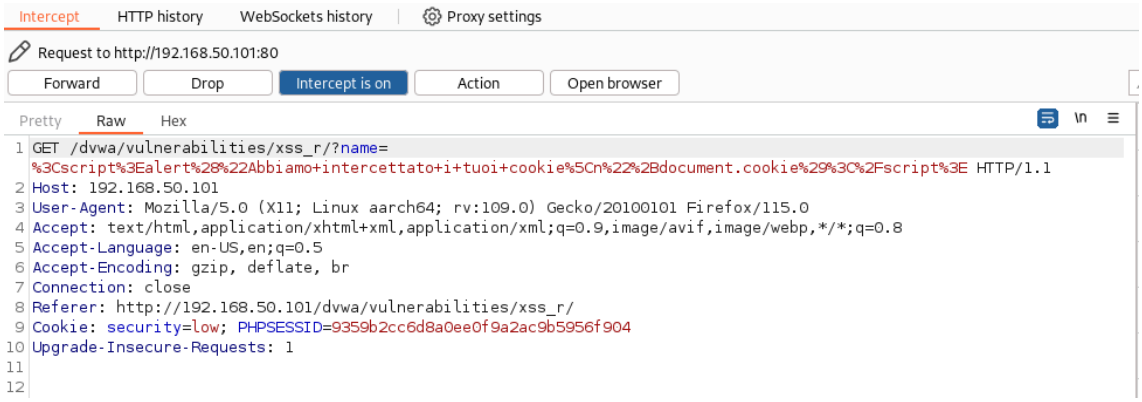
1.4. Inserimento codice script avanzato

Volendo estendere ulteriormente la ricerca di vulnerabilità, abbiamo inserito del semplice codice in Javascript che restituisce un alert con i cookie della sessione.

<script>alert("Abbiamo intercettato i tuoi cookie\n"+document.cookie)</script>



Risultato inserimento codice Javascript



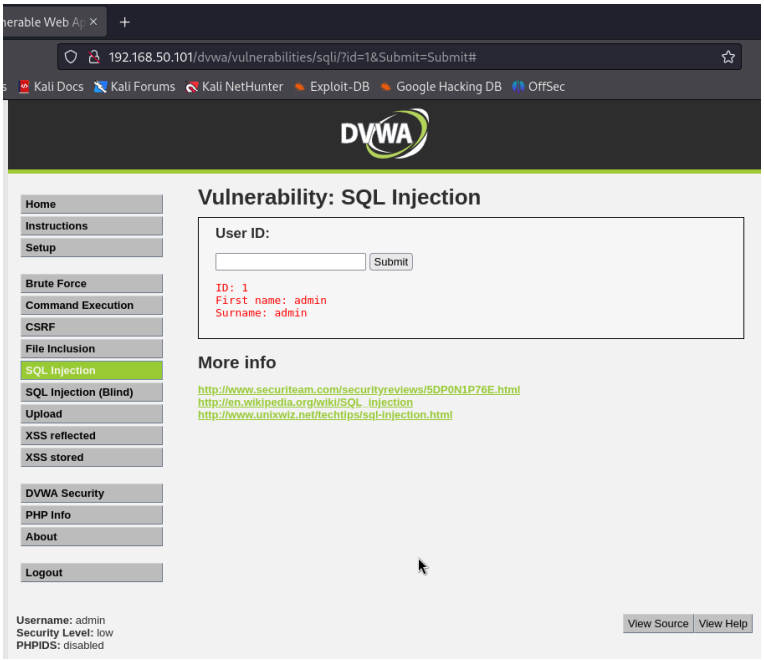
Dati intercettati da BurpSuite

2. SQL Injection (non blind)

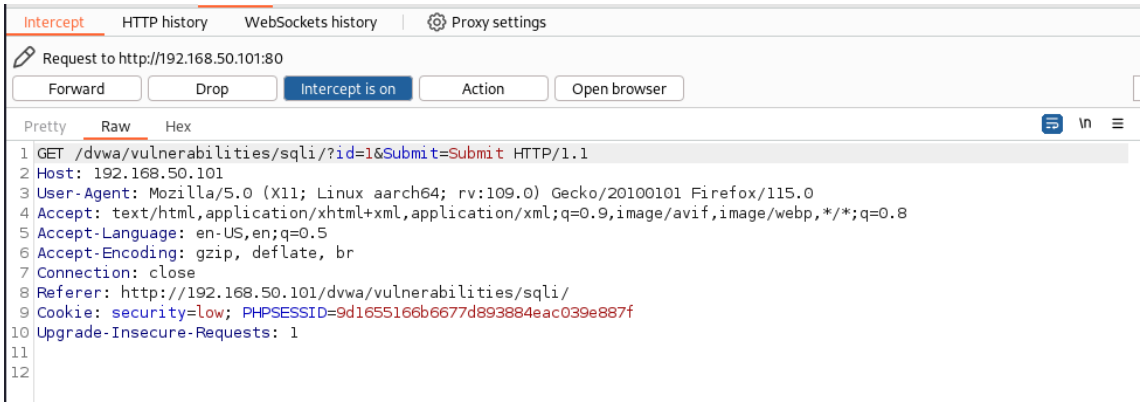
2.1. Test della funzionalità base

Inizialmente abbiamo testato la funzionalità proposta da DVWA assecondando il suo funzionamento corretto e inserendo un numero.

Analizzando il sistema di input è possibile notare come il parametro “id” è quello che passa il valore alla query che interroga dinamicamente il database target.



Risultato inserimento dai corretto



Dati intercettati da BurpSuite

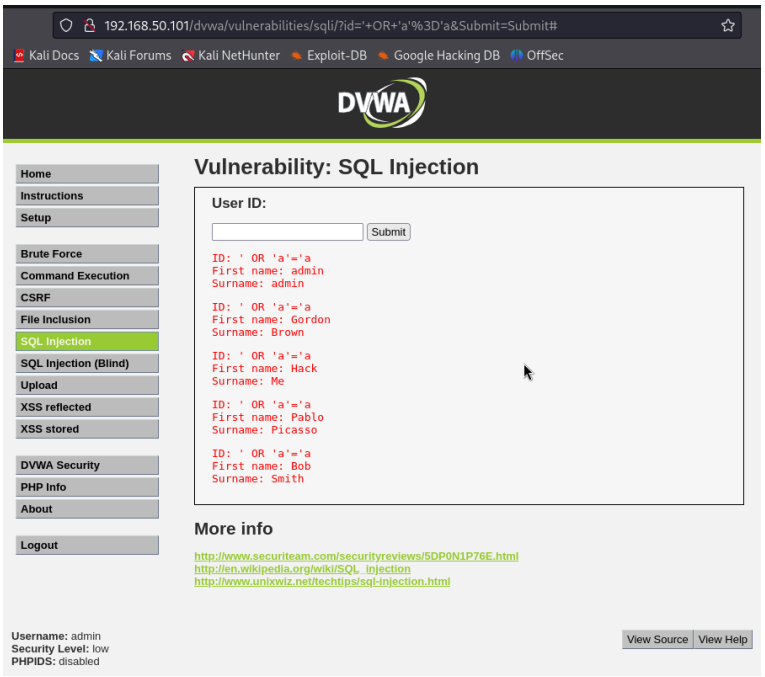
2.2. Inserimento codice SQL Injection semplice

Al fine di testare la reale presenza di una vulnerabilità, abbiamo provato ad inserire del semplice codice per effettuare un listato di tutta la tabella users.

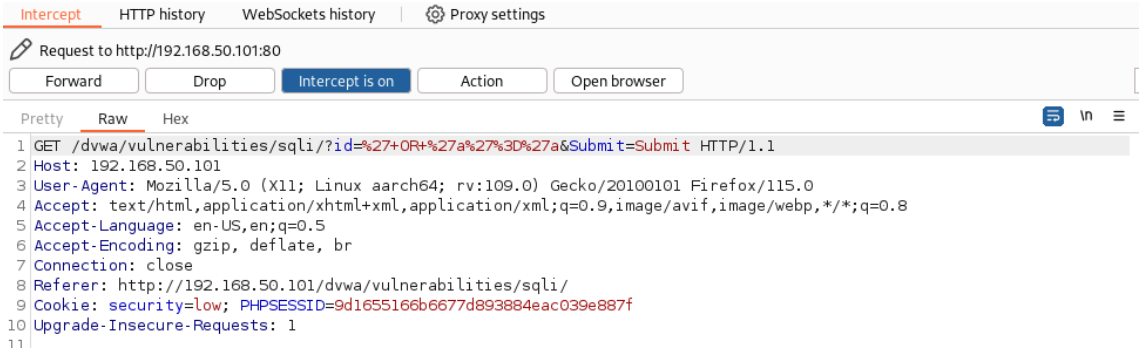
' OR 'a'='a

Il risultato di questo input è il seguente:

Il sito quindi risulta vulnerabile agli attacchi SQL Injection.



Risultato della SQL Injection semplice



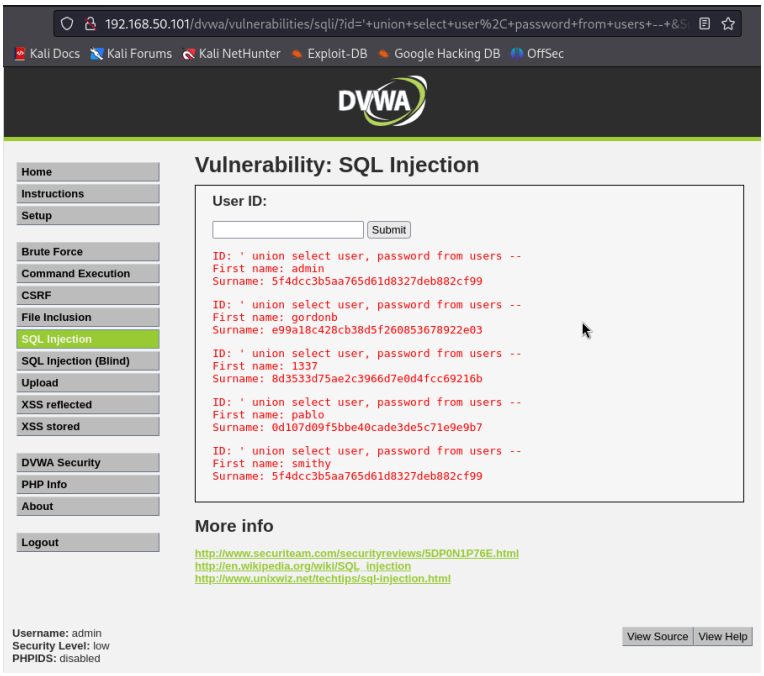
Risultato analisi con BurpSuite

2.3. s Inserimento codice SQL Injection avanzato

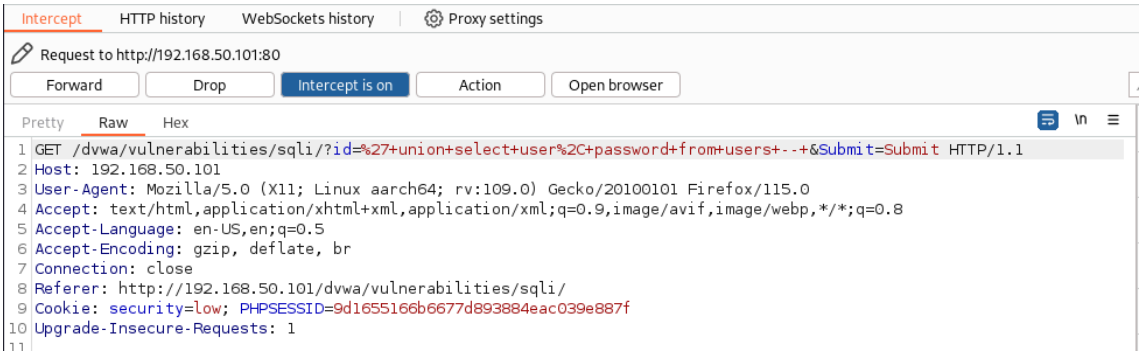
Proviamo ora ad inserire un codice che ci consenta, tramite una UNION, di visualizzare il contenuto della tabella users, ma che riporti il nome utente e le password tramite il codice:

' *union select user, password from users --*

Il risultato della Injection è il seguente:



Risultato della SQL Injection avanzato



Risultato analisi con BurpSuite

Vediamo come con nel campo “surname” vengono visualizzate le password.

Le password sono criptate, ma il codice, per lunghezza e tipologia, sembra una semplice trasformazione con un HASH MD5.

Consultando delle rainbow tables dove le password più comuni sono trasformate direttamente con Hash di vario tipo, tra cui MD5, possiamo provare ad individuare la password corretta.

Ad esempio la trasformazione di “*password*” con l’**HASH MD5** è:

5f4dcc3b5aa765d61d8327deb882cf99

ovvero la password trasformata dell’account **admin**

3. Conclusioni

Con sufficiente tempo a disposizione è possibile estendere ulteriormente la tipologia di danni che possono essere effettuati tramite questi attacchi, fino ad arrivare ad esempio ad un comando DROP table per distruggere i dati del Database dopo essersene impossessati, oppure inviare i cookie di sessione direttamente ad un sito creato dall'attaccante che a quel punto può usarli per accedere agli account della vittima.

Le possibili mitigazioni risiedono principalmente nell'implementazione di metodologie di escape dei caratteri speciali che rendono inoffensivi gli inserimenti alfanumerici pericolosi e che possono essere interpretati dal browser o dal codice php in modo errato.