

06 Web安全实验报告模板

《信息安全综合实践》实验报告

Web安全实验

一、实验目的

1. 掌握web服务中典型安全问题-SQL注入攻击的原理；
2. 学习SQL语言、MySQL数据库相关的知识；
3. 了解SQL注入攻击的过程、工具和方法；
4. 掌握SQL注入攻击的防范方法；
5. 了解其它web服务中的典型安全问题。

二、实验内容

序	内容	实验结果
1)	手工SQL注入攻击	- [] 失败 - [√] 成功
2)	利用SQLmap进行SQL注入攻击	- [] 失败 - [√] 成功
3)	XSS攻击（选做）	- [] 未做 - [] 失败 - [√] 成功
4)	DVWA其它攻击（选做）	- [] 未做 - [] 失败 - [√] 成功

三、分析和思考（90分）

1. 通过本次SQL注入攻击，你得到了关于该WEB服务器、数据库管理平台以及数据库、数据表的哪些信息？请分别列出。（20分）

获得以下信息：

- web server operating system: Windows
- web application technology: PHP 5.4.45, Apache 2.4.23
- back-end DBMS: MySQL >= 5.0
- available databases [7]:
 - challenges
 - dvwa
 - information_schema
 - mysql
 - performance_schema
 - security
 - test

- Database: dvwa
- 2 tables
 - guestbook
 - users

2. BurpSuite是Web应用程序测试的最佳工具之一。在本次SQLmap实验中，BurpSuite发挥了什么作用？试简单解释实验中test.txt文件的内容。（10分）

在此次实验中，burpsuite有以下作用：

- 拦截HTTP请求和响应：BurpSuite可以通过代理模式，拦截服务器返回的HTTP响应，即抓包，从而让我们可以对这些数据进行分析 and 修改。
- 数据包分析：BurpSuite可以对拦截的http响应进行分析和报告生成。
- 生成文件：其内容含有http请求，http协议版本，url，宿主机ip地址，cookie等等。

以下是文件的内容：

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 192.168.239.138
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://192.168.239.138/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=9ubs54fbpd8agvr94rthpdiiv2
Upgrade-Insecure-Requests: 1
```

文件内容是一个HTTP 请求报文，其中包含了客户端向服务器请求访问某个页面的具体信息和参数。具体解释如下：

- 第一行是请求行，包括请求的方法（GET）、请求的页面路径（/dvwa/vulnerabilities/sqli/），以及请求参数（id=1&Submit=Submit）。
- Host: 指定请求的服务器域名或者IP地址。
- User-Agent: 标识客户端使用的浏览器类型和版本号。
- Accept: 指定客户端接受的 MIME 类型，用于在请求中通知服务器客户端希望接受的响应内容类型。
- Accept-Language: 指定客户端接受的自然语言。
- Accept-Encoding: 指定客户端接受的内容编码方式，例如 gzip、deflate。
- Connection: 指定客户端与服务器之间连接的类型，如 close、keep-alive。

- Referer: 指定客户端浏览器当前页面的来源地址，即从哪个页面跳转到当前页面。
 - Cookie: 包含客户端的会话信息，以便服务器可以识别客户端的身份和状态。
 - Upgrade-Insecure-Requests: 指示客户端希望通过 HTTPS 访问页面，用于在请求中通知服务器客户端希望使用安全连接。
 - **特别说明**，文中的 q=* 是 Accept 请求头中的一个属性，用于指定客户端对不同 MIME 类型的接受程度。在 Accept 请求头中，每个 MIME 类型都可以被赋予一个质量值 (q 值)，表示客户端接受该 MIME 类型的程度，q 值的范围是 0 到 1，数值越高表示客户端接受该 MIME 类型的程度越高。
3. 在利用SQLmap实施sql注入攻击时，结合页面反馈及缓存数据文件，总结其攻击思路，说明其采用了哪些攻击手段。（15分）

攻击思路可以分为大致几步：

- 扫描目标网站：使用 SQLmap 对目标网站进行扫描，寻找可能存在 SQL 注入漏洞的页面和参数。
- 利用漏洞获取信息，确定注入点：通过对页面反馈的分析，确定哪些参数可能存在注入漏洞，并运用不同的注入方法进行测试，确定具体的注入点。
- 获取数据库信息：利用 SQLmap 的自动化工具，获取数据库的版本、表名、列名等信息，以便进一步的攻击。
- 访问数据库，利用漏洞对数据库进行操作：通过 SQLmap 提供的命令行界面或者图形化界面，访问数据库，执行 SQL 语句，获取或修改数据。

采用的攻击手段：

- GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable (with --not-string="Me") 此处是OR 关键字的布尔盲注注入漏洞。
- GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause' injectable 此处是基于 MySQL 版本 5.0 及以上的错误注入漏洞。
- GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable 此处是基于 MySQL 版本大于等于 5.0.12 的时间盲注注入漏洞。
- GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable 此处是基于 MySQL UNION 查询的注入漏洞
- 还有其它攻击手段，例如堆叠注入、二次注入、盲注等。SQL可以根据不同情况采用适当的攻击手段。

4. 在DVWA系统中安全级别分别设置为low、medium、high时，尝试实施SQL注入手工攻击，进行各级别漏洞的分析，说明攻击方法，并阐述如何防止该类漏洞出现（hint:可结合该实验的impossible级别代码进行分析）。（各级别提供关键截图至少1张，总截图不得超过8张）（30分）

- 源码:

vulnerabilities/sqli/source/impossible.php

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if( is_numeric( $id ) ) {
        $id = intval ( $id );
        switch ( $DVWA[ 'SQLI_DB' ] ) {
            case MYSQL:
                // Check the database
                $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
                $data->bindParam( ':id', $id, PDO::PARAM_INT );
                $data->execute();
                $row = $data->fetch();

                // Make sure only 1 result is returned
                if( $data->rowCount() == 1 ) {
                    // Get values
                    $first = $row[ 'first_name' ];
                    $last = $row[ 'last_name' ];

                    // Feedback for end user
                    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                }
                break;
            case SQLITE:
                global $sqlite_db_connection;

                $stmt = $sqlite_db_connection->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = :id LIMIT 1;' );
                $stmt->bindValue( ':id', $id, SQLITE3_INTEGER );
                $result = $stmt->execute();
                $result->finalize();
                if ( $result !== false ) {
                    // There is no way to get the number of rows returned
                    // This checks the number of columns (not rows) just
                    // as a precaution, but it won't stop someone dumping
                    // multiple rows and viewing them one at a time.

                    $num_columns = $result->numColumns();
                    if ( $num_columns == 2 ) {
                        $row = $result->fetchArray();

                        // Get values
                        $first = $row[ 'first_name' ];
                        $last = $row[ 'last_name' ];

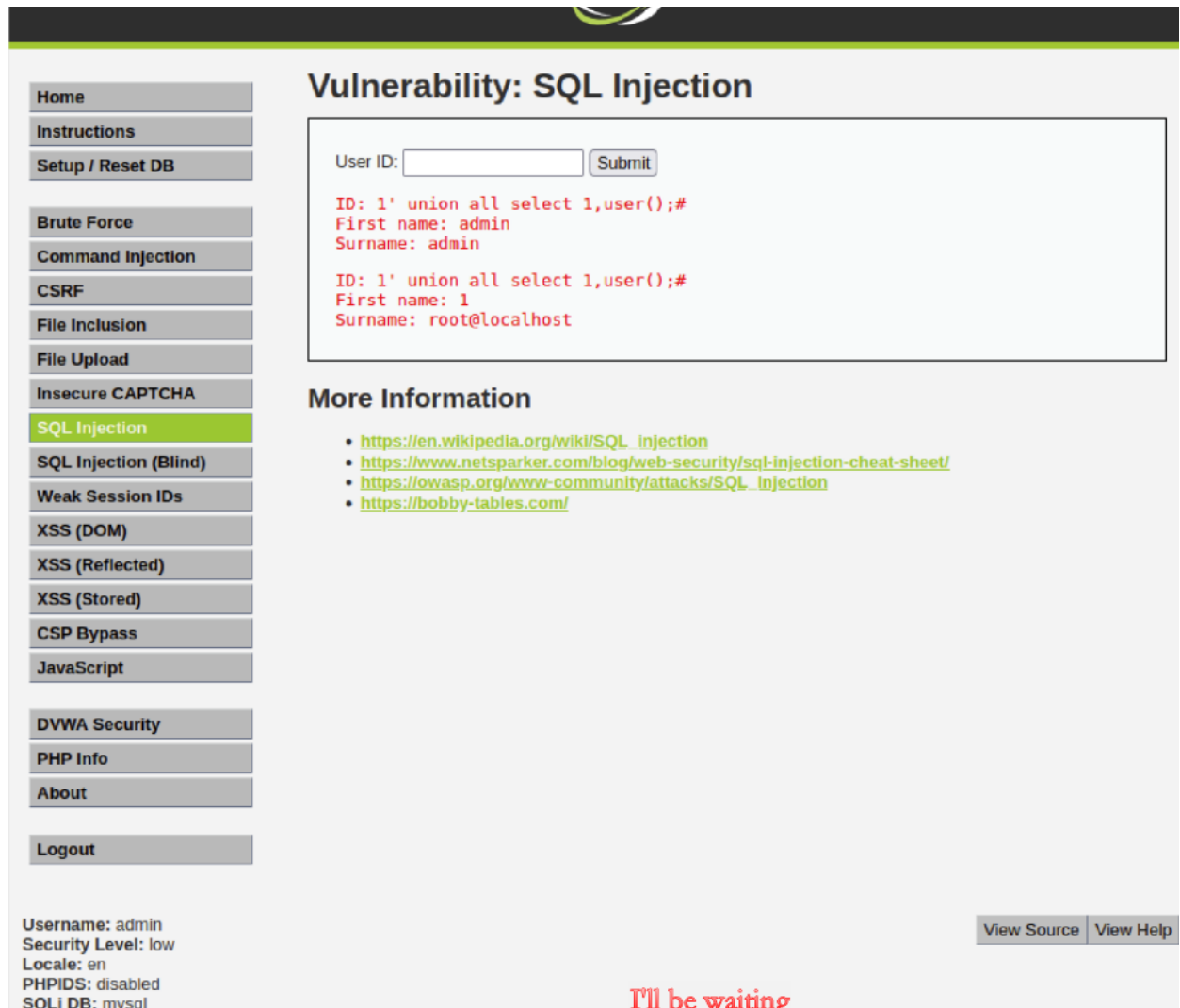
                        // Feedback for end user
                        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                    }
                }
                break;
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

该代码存在一些安全问题，例如缺乏授权检查以及如果用户ID参数未得到适当的清理，则可能存在SQL注入攻击的风险。


- low级别下:



The screenshot shows the DVWA web application interface. On the left is a sidebar with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a form with a "User ID:" input field and a "Submit" button. Below the form, the output shows the result of a successful SQL injection attack: "ID: 1' union all select 1,user();#", "First name: admin", "Surname: admin", "ID: 1' union all select 1,user();#", "First name: 1", "Surname: root@localhost". Below the output is a section titled "More Information" with four links: https://en.wikipedia.org/wiki/SQL_injection, <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>, https://owasp.org/www-community/attacks/SQL_injection, and <https://bobby-tables.com/>. At the bottom left, the user information is displayed: "Username: admin", "Security Level: low", "Locale: en", "PHPIDS: disabled", and "SQLi DB: mysql". At the bottom right, there are "View Source" and "View Help" links. A red text "I'll be waiting" is visible at the bottom center.

- 攻击方法：基于联合查询技术的SQL注入攻击。
- 漏洞分析：结合代码可以知道query变量为直接构造的sql查询语句，没有对用户的输入进行任何的过滤，导致sql注入的存在。而在实际操作中，通过一系列尝试注入猜测大概是什么类型注入，最后发现可以通过联合注入来进行sql注入攻击，以恶意查询，删改数据等等。
- 漏洞防范：可以考虑对输入的特殊字符进行转义，也可以限制用户在前端输入的内容。

- medium级别下:



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection

User ID:

ID: 1
First name: admin
Surname: admin

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Username: admin
Security Level: medium
Locale: en
PHPIDS: disabled
SQLi DB: mysql

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

1 x +

Send Cancel < >

Request

Pretty Raw Hex

```
1 POST /dwa/vulnerabilities/sqli/ HTTP/1.1
2 Host: 192.168.153.138
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 129
9 Origin: http://192.168.153.138
10 Connection: close
11 Referer: http://192.168.153.138/dwa/vulnerabilities/sqli/
12 Cookie: security=medium; PHPSESSID=t0ss8lpejbu9h14o7q6sp495
13 Upgrade-Insecure-Requests: 1
14
15 id=1 union select 1,group_concat(column_name)from information_schema.columns where table_name = 0x7573657273#6Submit=Submit
```

Response

Pretty Raw Hex Render

Vulnerability: SQL Injection

User ID: 1 Submit

ID: 1 union select 1,group_concat(column_name)from information_schema.columns where table_name = 0x7573657273#6
First name: admin
Surname: admin

ID: 1 union select 1,group_concat(column_name)from information_schema.columns where table_name = 0x7573657273#6
First name: 1
Surname: user_id,first_name,last_name,user,password,avatar,last_login,failed_login,ip_address

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Username: admin
Security Level: medium
Locale: en
PHPIDS: disabled
SQL DB: mysql

View Source View Help

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

SQL Injection Source

vulnerabilities/sqli/source/medium.php

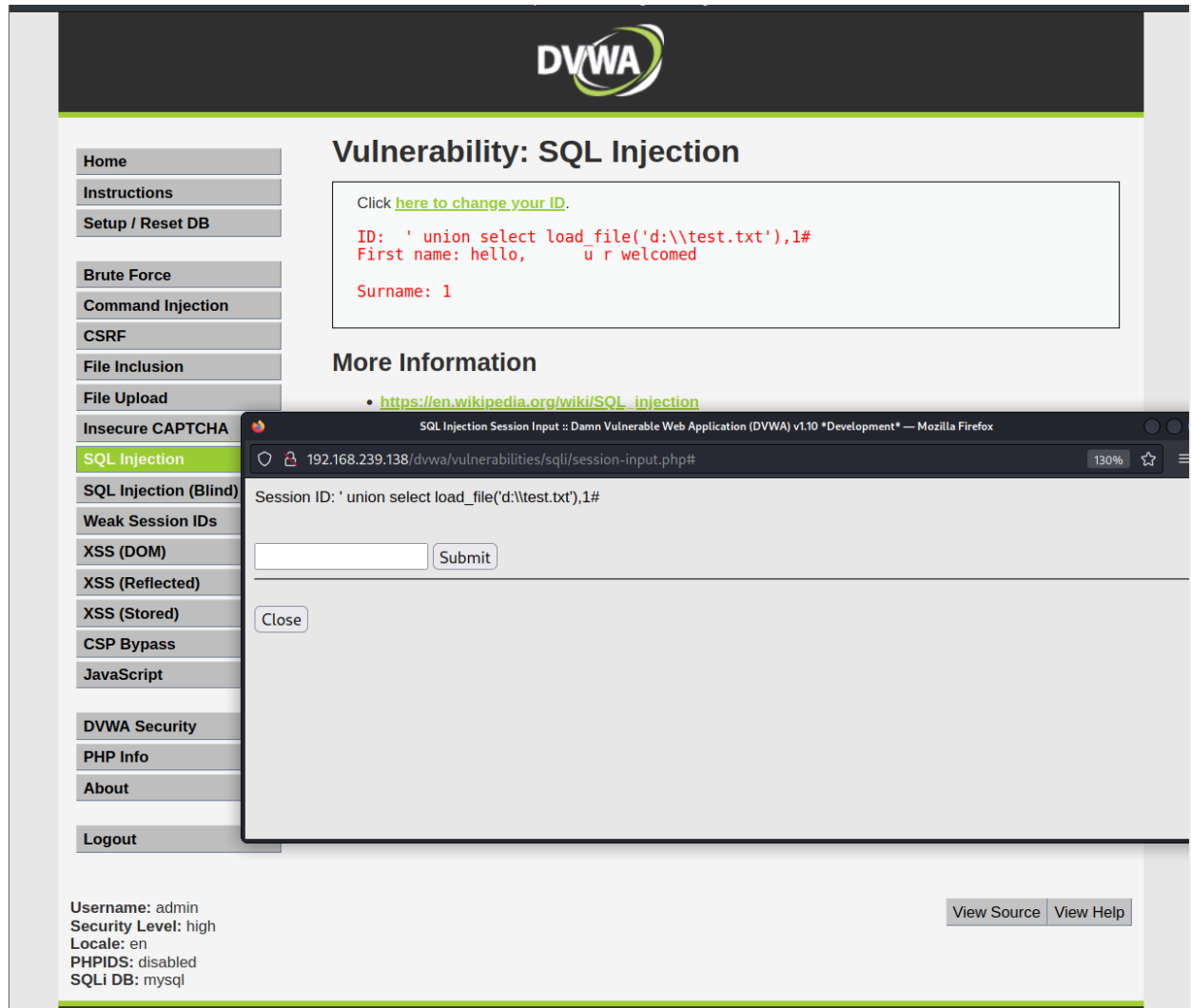
```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];

    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);
```

- 攻击方法：基于联合查询技术的SQL注入攻击。
- 漏洞分析：结合后端代码分析，Medium级别的代码利用mysqli_real_escape_string函数对特殊符号进行转义，还设置了下拉选择表单。通过burpsuite抓包分析可知，参数id为1 or 1=1 #，查询成功，说明存在数字型注入，因为其不需要借助引号，所以mysqli_real_escape_string函数形同虚设。之后通过联合注入攻击就可以获得对应的数据信息。
- 漏洞防范：可以在SQL查询语句中添加LIMIT 1，控制只输出一个结果。

- 高级别下:



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The main heading is "Vulnerability: SQL Injection". Below this, there is a box containing the following text:

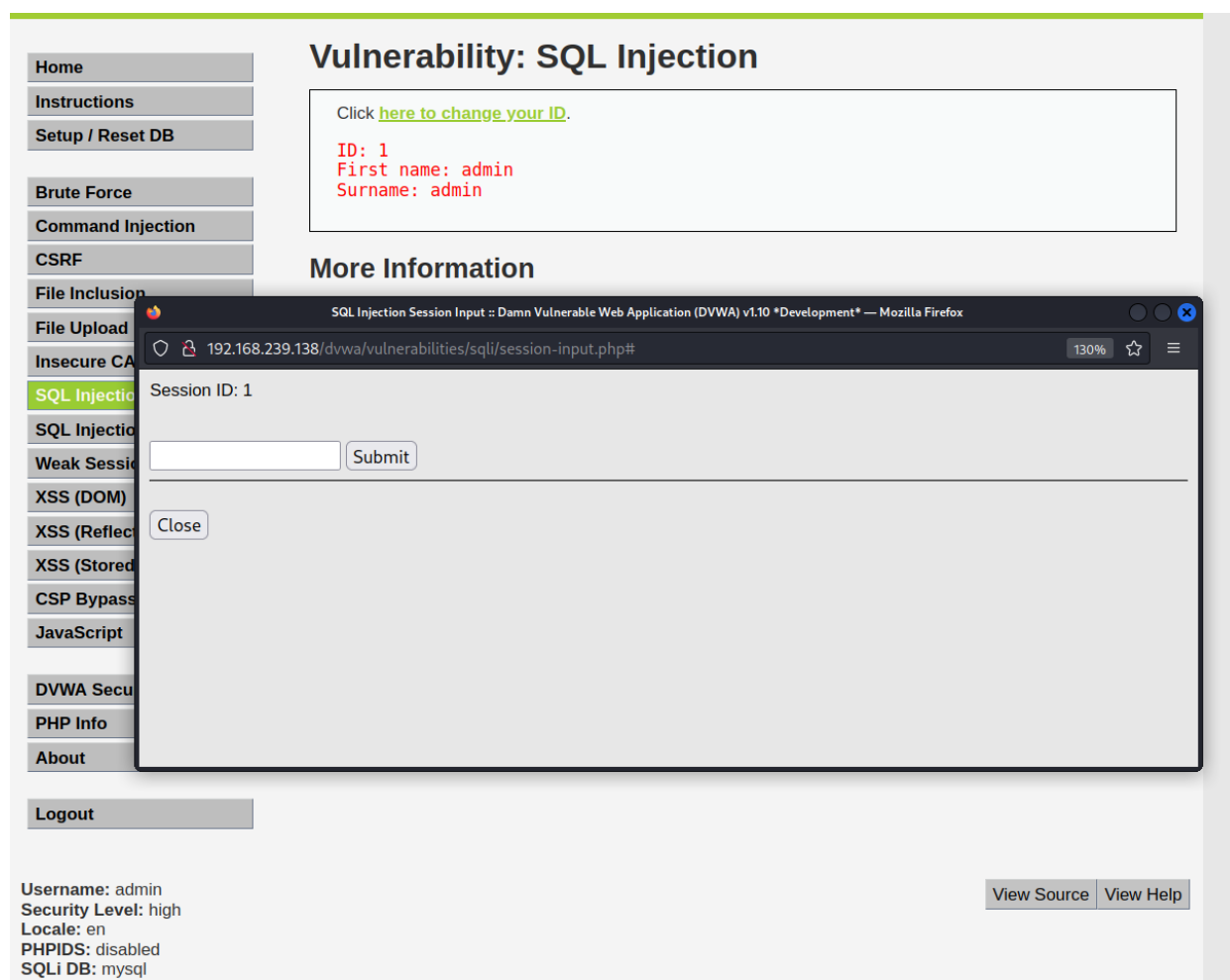
```
Click here to change your ID.  
ID: ' union select load_file('d:\\test.txt'),1#  
First name: hello,      u r welcomed  
Surname: 1
```

Below this box, there is a section titled "More Information" with a link to https://en.wikipedia.org/wiki/SQL_injection.

The left sidebar contains a list of navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout.

At the bottom of the page, there is a status bar showing: Username: admin, Security Level: high, Locale: en, PHPIDS: disabled, and SQLi DB: mysql. There are also links for "View Source" and "View Help".

A browser window is overlaid on the page, showing the URL "192.168.239.138/dvwa/vulnerabilities/sqli/session-input.php#" and the session ID input field with the value "Session ID: ' union select load_file('d:\\test.txt'),1#". The browser window also shows a "Submit" button and a "Close" button.



- 攻击方法：基于联合查询技术的SQL注入攻击。
 - 漏洞分析：high级别SQL查询语句中添加了LIMIT 1。
 - 漏洞防范：可以考虑采用PDO技术，划清代码与数据的界限，可有效防御SQL注入，同时只有返回的查询结果数量为一时，才会成功输出，Anti-CSRFtoken机制的加入了进一步提高了安全性。
5. 当前Web网站在典型攻击防范方面，如SQL注入攻击或XSS攻击等，有哪些方法或工具？列举2-3种，并试以某方法或工具为例，分析其优缺点。（15分）
- Web网站在防范典型攻击方面，可以采用以下方法或工具：
 - 输入验证和过滤：对于用户输入的数据进行验证和过滤，防止恶意输入，避免SQL注入攻击和XSS攻击等。例如使用正则表达式、过滤敏感字符等技术，可以避免一些常见的攻击方式。
 - 漏洞扫描工具：利用漏洞扫描工具对Web应用程序进行扫描，检查其中可能存在的漏洞。例如常用的工具有Nessus、OpenVAS等，可以自动化地检测漏洞，并提供详细的报告和建议。
 - 防火墙：使用Web应用程序防火墙（WAF）等技术来保护Web应用程序。WAF可以检测和拦截恶意流量，阻止攻击者利用已知的漏洞攻击系统。例如ModSecurity是一个流行的开源WAF，它可以检测并拦截多种Web攻击类型。
 - 渗透测试工具：比如SQLMap，SQLMap是GitHub上提供的自动SQL和数据库接管工具，这个开源渗透测试工具可以自动检测和利用SQL漏洞或其他接管数据库服务器的攻击，配备了强大的检测引擎，为最终渗透测试人员提供了许多便利的功

能，可进行数据库指纹识别、从数据库获取数据、访问底层文件系统，以及通过带外连接在操作系统上执行命令等。

- 以漏洞扫描工具为例，分析其优缺点：

- 优点：

- 自动化：漏洞扫描工具可以自动化地扫描Web应用程序中的漏洞，减少了手动检查的工作量，提高了扫描的效率。
 - 详细报告：漏洞扫描工具可以提供详细的报告和建议，帮助Web应用程序的管理员快速识别并解决潜在的安全问题。
 - 覆盖面广：漏洞扫描工具可以扫描多种Web攻击类型，包括SQL注入攻击、XSS攻击、CSRF攻击等，可以有效地帮助Web应用程序识别和防范这些攻击。

- 缺点：

- 误报率高：漏洞扫描工具存在误报率的问题，有些扫描工具可能会将正常的行为误认为是攻击行为，从而导致误报。
 - 漏报率高：漏洞扫描工具可能无法检测出所有的漏洞，有些漏洞可能需要手动检查或其他技术手段来识别。
 - 成本较高：一些商业漏洞扫描工具的价格较高，可能不适合小型Web应用程序或个人使用。

四、实验总结（收获和心得）（5分）

通过本次实验我初步了解了数据库，大概了解了如何SQL注入，感到十分有意思并很期待大三能够深入学习相关知识。

五、尚存问题或疑问、建议（5分）

- PHP语言不是很懂，部分函数的功能不了解