

Jared Ricks, Jack Lin, Kolbe Williams-Wimmer

Part 1: Setting Up the Environment in Ubuntu:

Update the Ubuntu System:

The system was updated, but it restarted so we could not screenshot.

Install Apache (webserver), MySQL (or MariaDB), and PHP:

Installing Apache:

```
labuser1@ML-RefVm-535928:~$ sudo apt install apache2 -y
[sudo] password for labuser1:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Installing MySQL:

```
labuser1@ML-RefVm-535928:~$ sudo apt install mysql-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Installing PHP:

```
labuser1@ML-RefVm-535928:~$ sudo apt install php libapache2-mod-php php-mysql -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Download DVWA: Clone DVWA from GitHub:

```
labuser1@ML-RefVm-535928:/var/www/html$ sudo git clone https://github.com/digininja/DVWA.git
Cloning into 'DVWA'...
remote: Enumerating objects: 4857, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 4857 (delta 5), reused 9 (delta 3), pack-reused 4840 (from 1)
Receiving objects: 100% (4857/4857), 2.43 MiB | 7.31 MiB/s, done.
Resolving deltas: 100% (2342/2342), done.
labuser1@ML-RefVm-535928:/var/www/html$
```

Set Permissions: Set appropriate permissions for the DVWA folder:

```
labuser1@ML-RefVm-535928:/var/www/html$ sudo chown -R www-data:www-data /var/www/html/DVWA
labuser1@ML-RefVm-535928:/var/www/html$
```

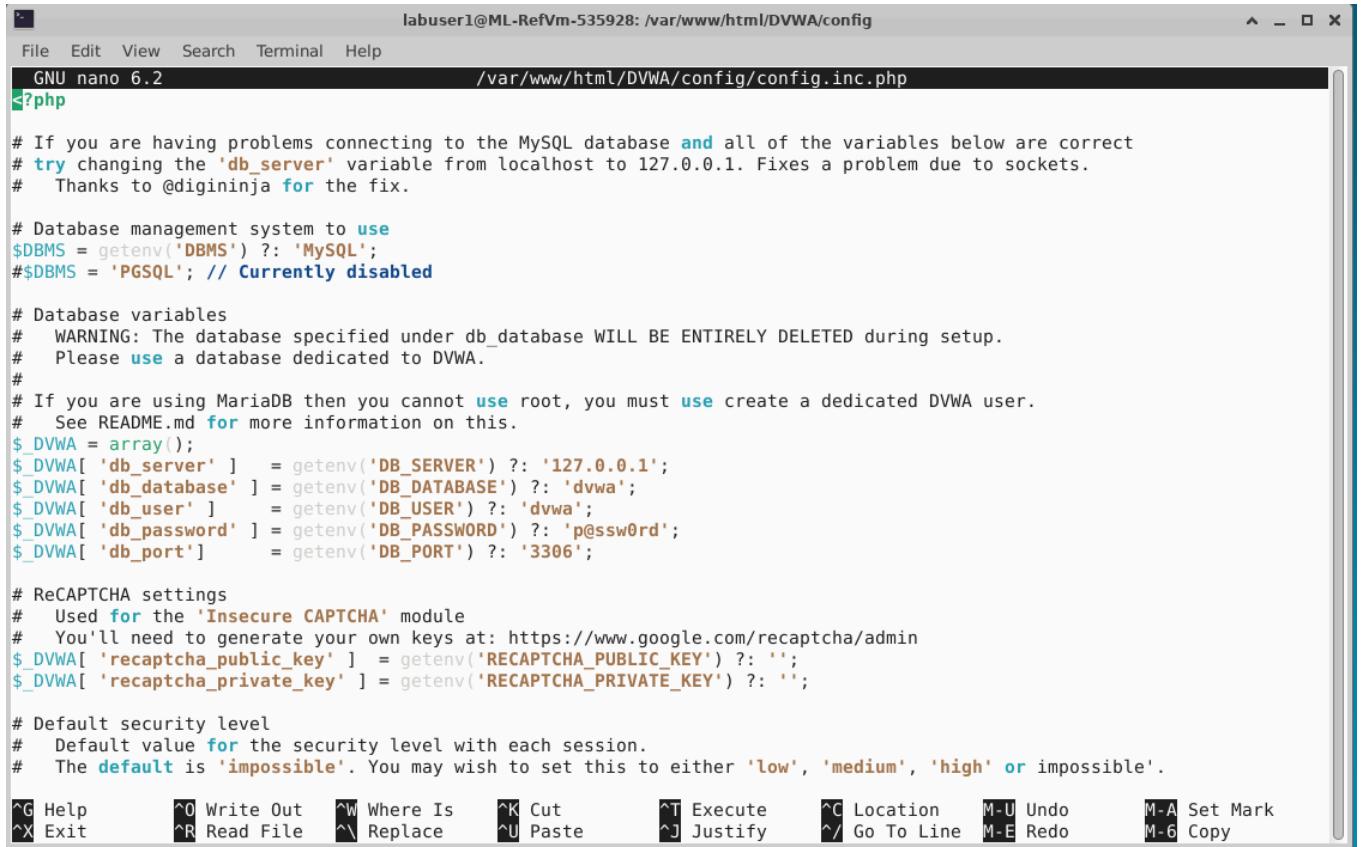
This gives the Apache webserver permission to access the DVWA files.

Copy config.inc.php.dist into config.inc.php:

```
labuser1@ML-RefVm-535928:/var/www/html$ cd /var/www/html/DVWA/config/
labuser1@ML-RefVm-535928:/var/www/html/DVWA/config$ sudo cp config.inc.php.dist config.inc.php
labuser1@ML-RefVm-535928:/var/www/html/DVWA/config$
```

The source code is copied to the active configuration file.

Open the DVWA configuration file to set up database credentials:



The screenshot shows a terminal window titled "labuser1@ML-RefVm-535928: /var/www/html/DVWA/config". The window contains the DVWA configuration file, specifically config.inc.php. The code is written in PHP and defines database connection variables. It includes comments explaining the setup, such as changing the db_server variable from localhost to 127.0.0.1. The nano editor interface is visible at the bottom, showing standard keyboard shortcuts for various operations like Help, Write Out, Cut, Paste, Execute, and Undo.

```
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ] = getenv('DB_USER') ?: 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'p@ssw0rd';
$_DVWA[ 'db_port' ] = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
$_DVWA[ 'recaptcha_private_key' ] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';

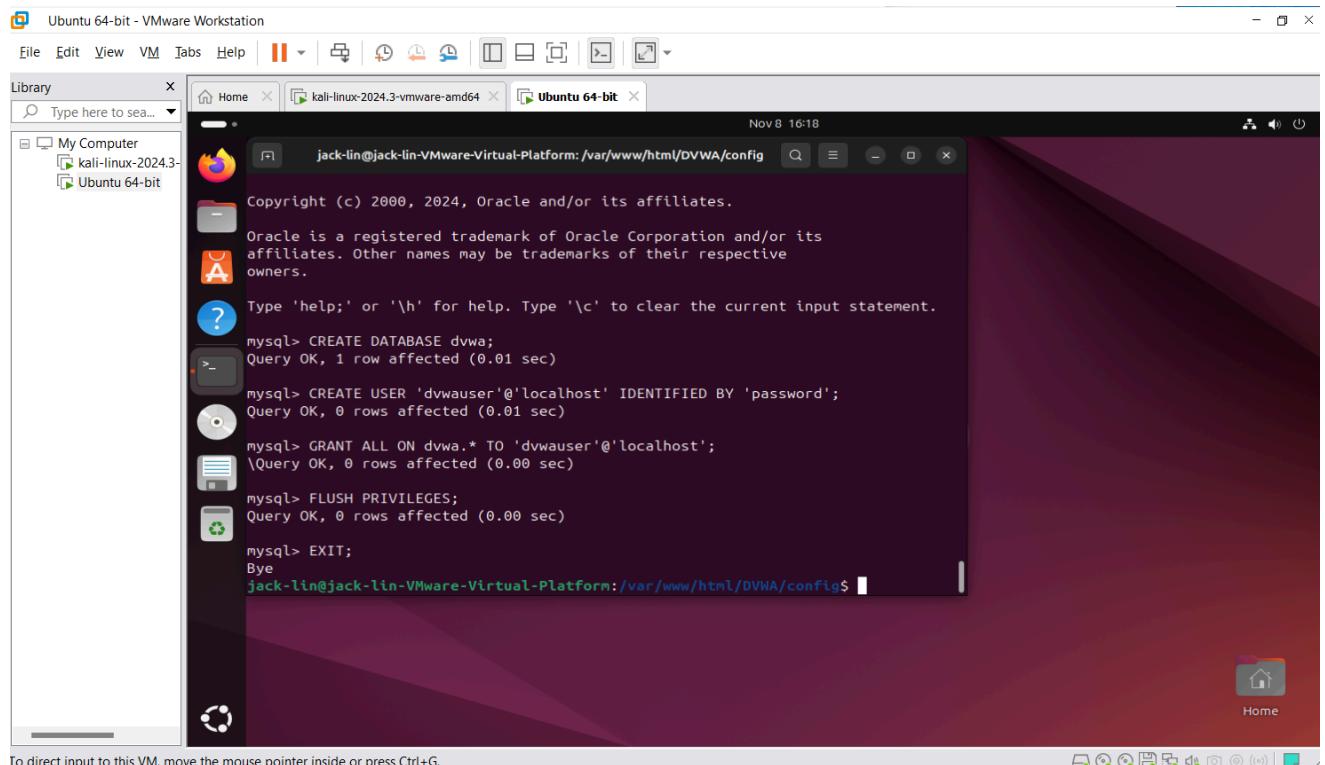
# Default security level
# Default value for the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible'.
```

Change these lines to match your MySQL/MariaDB setup:

```
$_DVWA = array();
$_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'dvwauser';
$_DVWA[ 'db_password' ] = 'password';
$_DVWA[ 'db_port' ] = '3306';
```

Create DVWA Database:

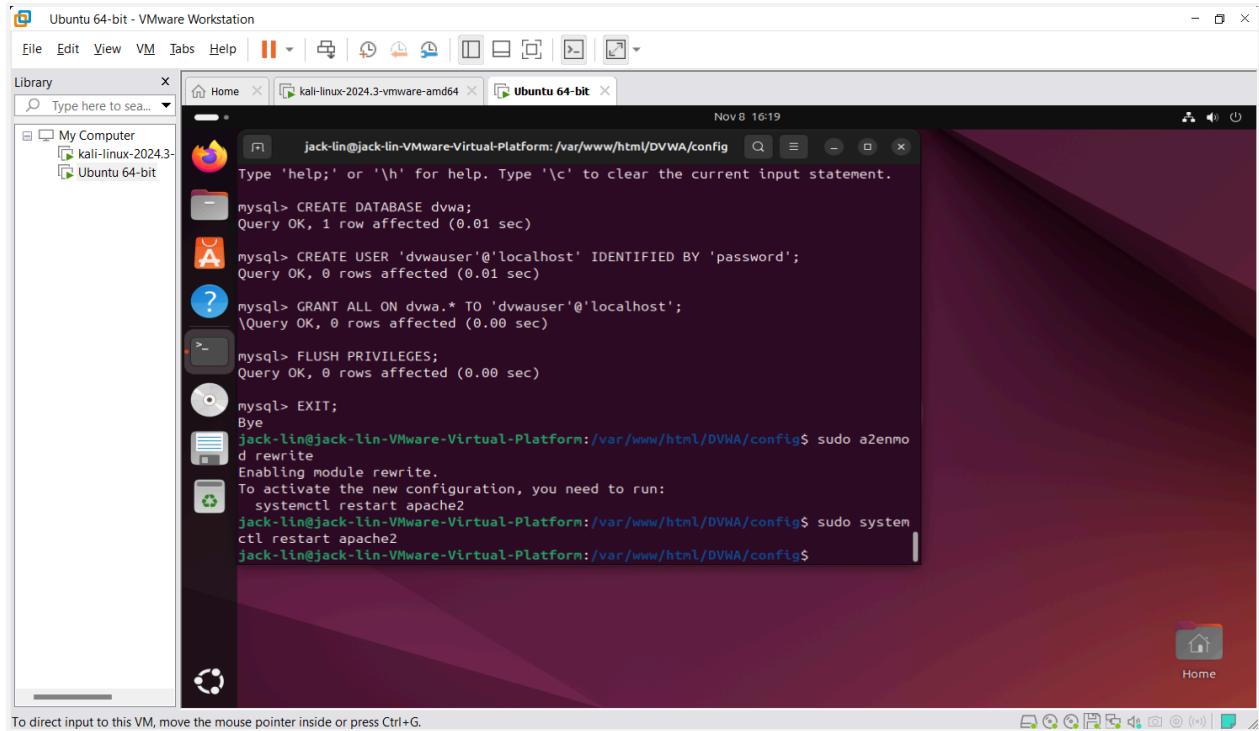
Open MySQL and create a database for DVWA:



```
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> CREATE DATABASE dvwa;  
Query OK, 1 row affected (0.01 sec)  
mysql> CREATE USER 'dvwauser'@'localhost' IDENTIFIED BY 'password';  
Query OK, 0 rows affected (0.01 sec)  
mysql> GRANT ALL ON dvwa.* TO 'dvwauser'@'localhost';  
Query OK, 0 rows affected (0.00 sec)  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)  
mysql> EXIT;  
Bye  
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$
```

A database and a database user are created. All permissions are then granted to the database and the user and the changes are applied.

Enable mod_rewrite and Restart Apache:
This helps allow Apache to access DVWA.



```
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ mysql> CREATE DATABASE dvwa;
Query OK, 1 row affected (0.01 sec)

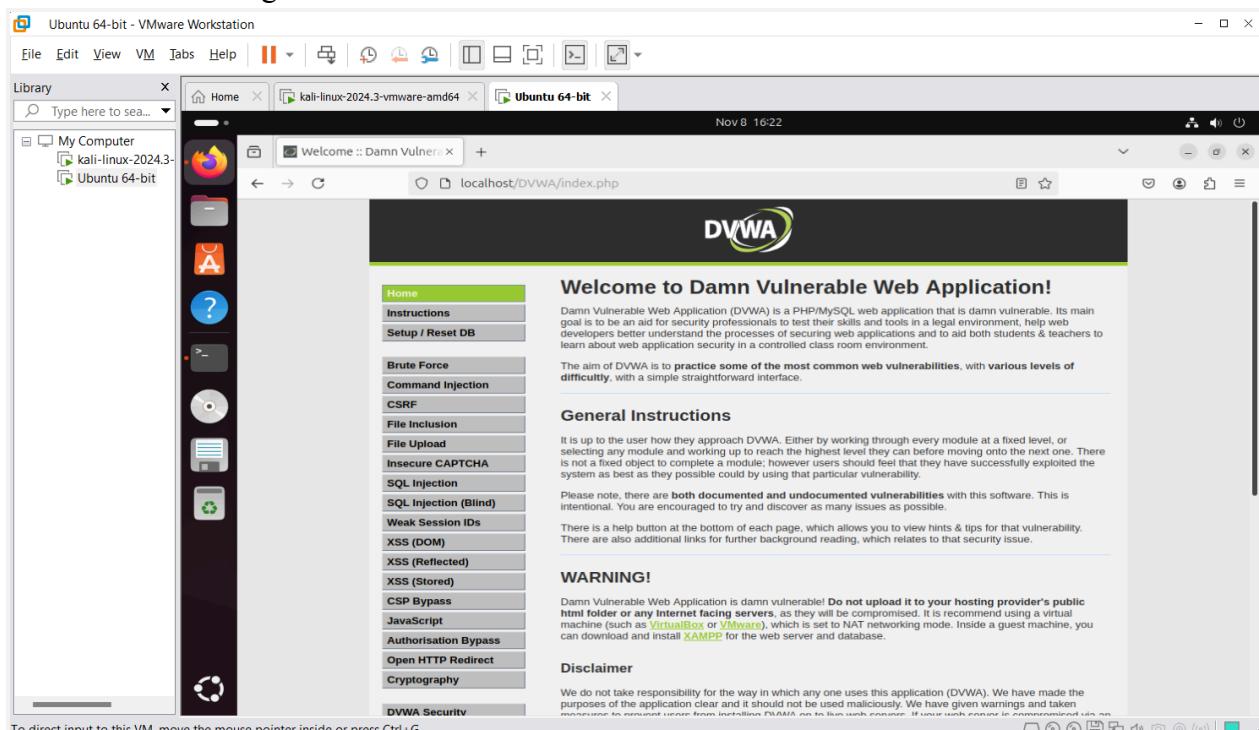
mysql> CREATE USER 'dvwauser'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL ON dvwa.* TO 'dvwauser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> EXIT;
Bye
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ sudo a2enmod
d rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
    systemctl restart apache2
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ sudo system
ctl restart apache2
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$
```

Set Up DVWA:
Access DVWA using Firefox:



Part 2: Conducting an SQL Injection Attack

The security level in DVWA is set to low:

A screenshot of a VMware Workstation window titled "Ubuntu 64-bit - VMware Workstation". Inside, a Firefox browser window shows the DVWA (Damn Vulnerable Web Application) interface. The main page title is "DVWA Security :: Damn V.". The left sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, Cryptography, and DVWA Security. The "SQL Injection" item is highlighted. The main content area is titled "Security Level". It states: "Security level is currently: **low**". Below this, it says: "You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:". A numbered list follows: 1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques. 2. Medium - The security level mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques. 3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation. Similar to Insecure CAPTCHA, The Flags (CTFs) competitions. 4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'. A dropdown menu shows "Low" is selected, with a "Submit" button next to it. Below the dropdown is a text input field containing "Security level set to low". At the bottom of the page, there is a footer with links to DVWA documentation and other resources.

Step 2: Access SQL Injection Page

This result is given by clicking on SQL Injection in the left column and entering a User ID of 1. The expected result occurs, which gives information on the user with this ID number:

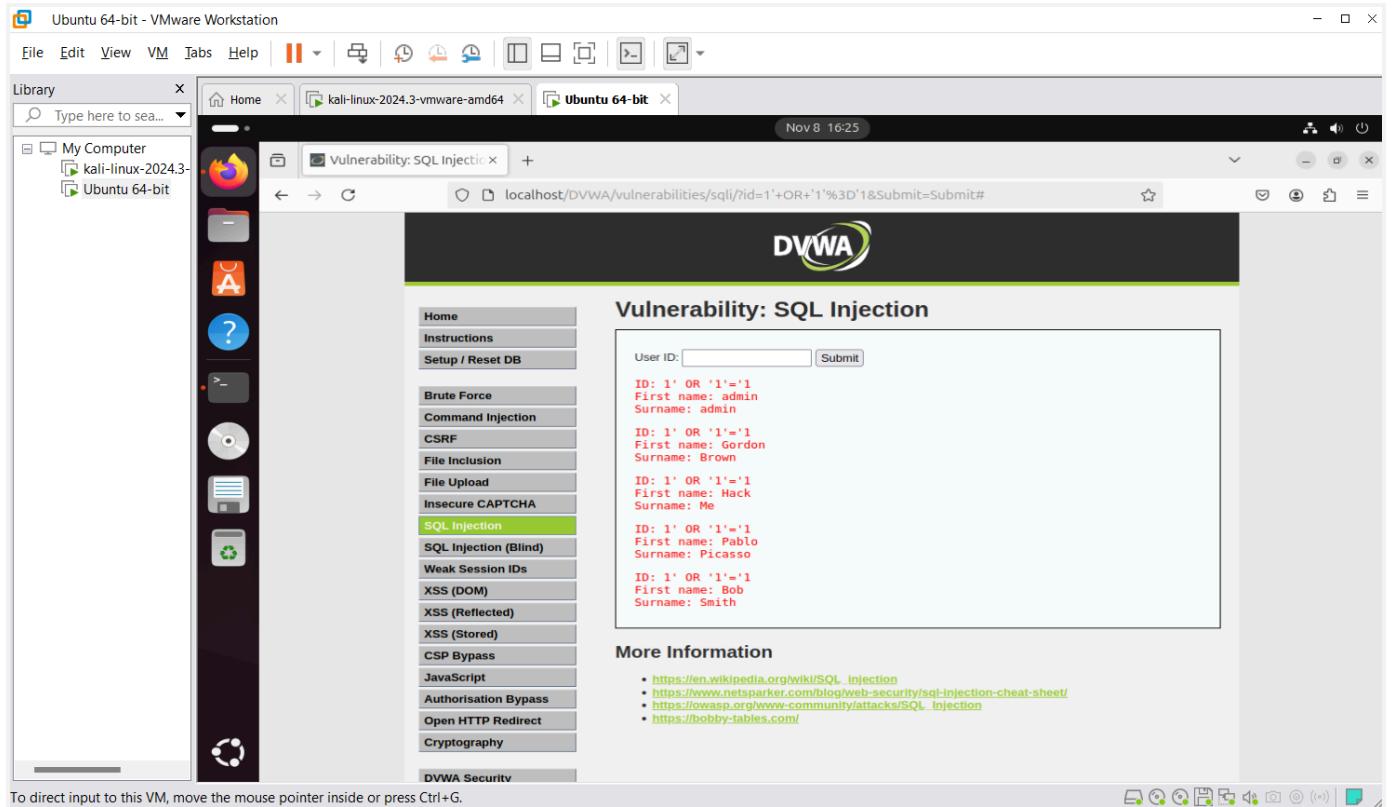
A screenshot of a VMware Workstation window titled "Ubuntu 64-bit - VMware Workstation". Inside, a Firefox browser window shows the DVWA interface. The main content area is titled "Vulnerability: SQL Injection". On the left, the sidebar shows the "SQL Injection" item is selected. The main form has a "User ID:" input field containing "1" and a "Submit" button. Below the form, the output shows: "ID: 1", "First name: admin", and "Surname: admin". Under "More Information", there is a list of 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
- <https://bobby-tables.com>

 At the bottom of the page, there is a footer with links to DVWA documentation and other resources.

Step 3: Perform SQL Injection Attack

Now, we enter the payload of: 1' OR '1'='1



All user information is displayed because the payload of 1 or 1 = 1 will always be evaluated as true. Since there was no password parameter, the payload did not even need to comment anything out.

Part 3: Understanding SQL Injection Attack

When we entered

1' OR '1'='1, it modified the SQL query to:

```
SELECT * FROM users WHERE id = '1' OR '1'='1';
```

The OR '1'='1' clause always evaluates to true, returning all rows in the database instead of just one specific user, indicating a successful SQL injection attack.

Part 4: SQL Injection Mitigation Techniques

Here are some commonly used techniques to mitigate SQL injection vulnerabilities:

1. Parameterized Queries:

- Use prepared statements and parameterized queries to separate data from code.
- This method prevents user input from being executed as SQL code.

2. Escaping User Input:

- Escaping special characters in SQL queries can reduce the risk of SQL Injection.
 - This method should only be used in combination with other techniques.
3. Input Validation:
 - Validate and sanitize user inputs. For instance, ensure numerical fields contain only numbers.
 4. Using ORM or Stored Procedures:
 - Using Object-Relational Mapping (ORM) libraries or stored procedures helps prevent SQL injection by abstracting SQL queries.

Part 5: Demonstrating SQL Injection Mitigation

1. Example Code for a Parameterized Query (PHP):
 - In a real-world scenario, code like the following would replace vulnerable queries:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");
$stmt->bind_param("i", $id);
$stmt->execute();
$result = $stmt->get_result();
```

- This code binds user input (\$id) as a parameter, which prevents the input from being interpreted as part of the SQL code.
2. Testing the Mitigation:
 - In a fully implemented environment, if you attempt the SQL injection payload (1' OR '1'='1), it will be treated as a string and not executed as SQL code.
 - Since this is a hypothetical scenario, the DVWA app itself would need to be rewritten to reflect this approach.
 3. Setting DVWA Security to High:
 - Go to DVWA Security settings and set the security level to High.
 - Now, try the same SQL injection payload. The app should prevent this attack because of more secure code practices.

Now, the security level in DVWA is set to high:

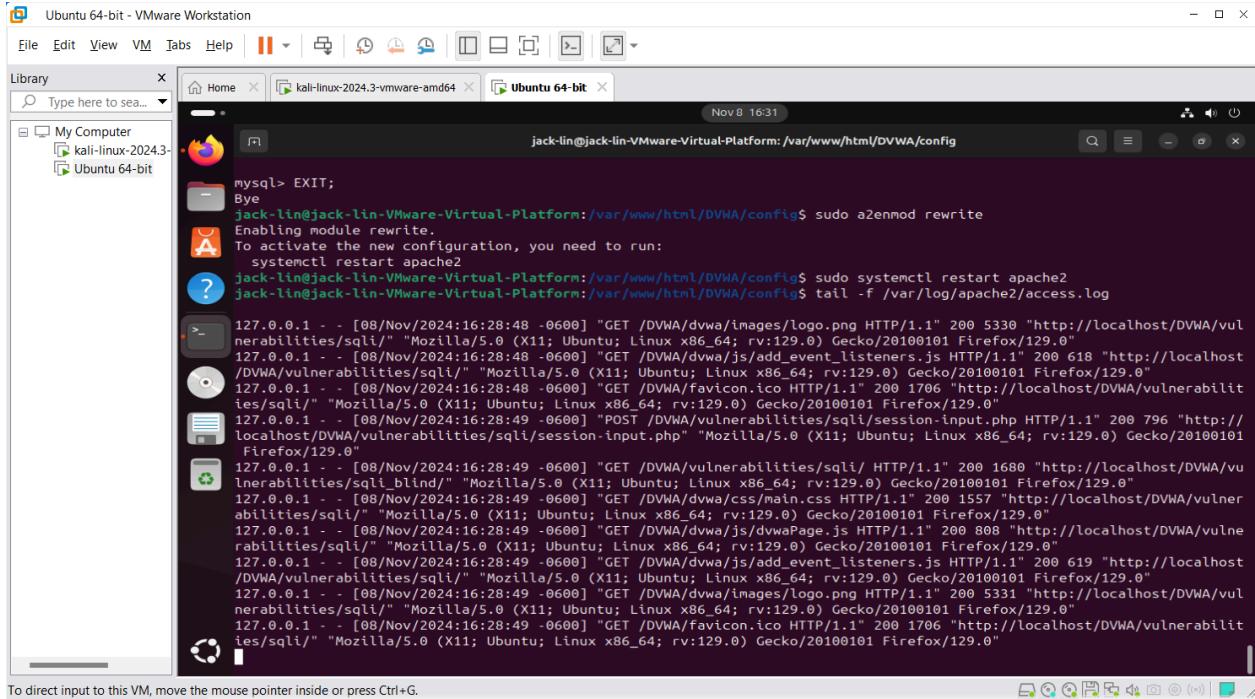
The screenshot shows a Firefox browser window running on a Kali Linux VM. The URL is `localhost/DVWA/security.php`. The DVWA logo is at the top. The main content area is titled "DVWA Security" with a lock icon. A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute 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, Authorisation Bypass, Open HTTP Redirect, and Cryptography. The "SQL Injection" option is highlighted. Below the sidebar, a section titled "Security Level" says "Security level is currently: high". It explains that users can set the security level to low, medium, high, or impossible. A dropdown menu is set to "High" and a "Submit" button is present. A message box below says "Security level set to high".

The same payload of `1' OR '1'='1` only gives the result for one user:

The screenshot shows a Firefox browser window running on a Kali Linux VM. The URL is `localhost/DVWA/vulnerabilities/sql/`. The DVWA logo is at the top. The main content area is titled "Vulnerability: SQL Injection". A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, and Cryptography. The "SQL Injection" option is highlighted. Below the sidebar, a message box says "Click here to change your ID." followed by "ID: 1' OR '1'='1", "First name: admin", and "Surname: admin". A "More Information" section lists several links about SQL injection. A smaller window titled "SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA)" is open, showing the session ID input field with the value "Session ID: 1' OR '1'='1".

All entries are not shown. Instead, only the user information for ID 1 is shown because, in the payload, 1 was the first character. This security level uses better coding practices, such as parameterized queries and input sanitization. Thus, the entry for the given ID was displayed as it was supposed to, not the entire list.

Part 6: Securing DVWA and Reviewing Logs



The screenshot shows a terminal window titled "Ubuntu 64-bit - VMware Workstation". The terminal is running on a Kali Linux host. The user has run several commands to manage Apache modules and then tail the Apache access log:

```
mysql> EXIT;
Bye
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ sudo systemctl restart apache2
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ tail -f /var/log/apache2/access.log

127.0.0.1 - - [08/Nov/2024:16:28:48 -0600] "GET /DVWA/dvwa/images/logo.png HTTP/1.1" 200 5330 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:48 -0600] "GET /DVWA/dvwa/js/add_event_listeners.js HTTP/1.1" 200 618 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:48 -0600] "GET /DVWA/favicon.ico HTTP/1.1" 200 1706 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "POST /DVWA/vulnerabilities/sql/session-input.php HTTP/1.1" 200 796 "http://localhost/DVWA/vulnerabilities/sql/session-input.php" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "GET /DVWA/vulnerabilities/sql/_blind/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "GET /DVWA/dvwa/css/main.css HTTP/1.1" 200 1557 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "GET /DVWA/dvwa/js/dvwaPage.js HTTP/1.1" 200 808 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "GET /DVWA/dvwa/images/logo.png HTTP/1.1" 200 5331 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
127.0.0.1 - - [08/Nov/2024:16:28:49 -0600] "GET /DVWA/favicon.ico HTTP/1.1" 200 1706 "http://localhost/DVWA/vulnerabilities/sql/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:129.0) Gecko/20100101 Firefox/129.0"
```

These are the most recent entries in the Apache access log, which show the actions we took using DVWA on the Apache server just before running this command. This log could be used for troubleshooting or security monitoring on the Apache server.

```

jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ tail -f /var/log/mysql/error.log
2024-11-08T22:09:59.47188Z 7 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.39-0ubuntu0.24.04.2).
2024-11-08T22:09:59.47351Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '127.0.0.1' port: 33060, socket: /var/run/mysqld/mysqld.sock
2024-11-08T22:10:00.99152Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.39-0ubuntu0.24.04.2) (Ubuntu).
2024-11-08T22:10:01.594894Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.39-0ubuntu0.24.04.2) starting as process 5348
2024-11-08T22:10:01.599389Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2024-11-08T22:10:01.682179Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2024-11-08T22:10:01.761095Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2024-11-08T22:10:01.761115Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2024-11-08T22:10:01.772130Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '127.0.0.1' port: 33060, socket: /var/run/mysqld/mysqld.sock
2024-11-08T22:10:01.772256Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.39-0ubuntu0.24.04.2' socket: '/var/run/mysqld/mysqld.sock' port: 3306 (Ubuntu).
^C
jack-lin@jack-lin-VMware-Virtual-Platform:/var/www/html/DVWA/config$ S

```

These are the most recent entries in the error log for My SQL on our virtual machine. This could be used to view errors during connection and operation or by another user.

2. Disable DVWA or Set the Security Level to High:

DVWA Security :: Damn V.9

localhost/DVWA/security.php

DVWA Security

Security level is currently: **high**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA.

1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.

2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploit and increase their skills.

3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation similar in various Capture The Flags (CTFs) competitions.

4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.

Prior to DVWA v9.3, this level was known as "high".

High Security level set to high

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

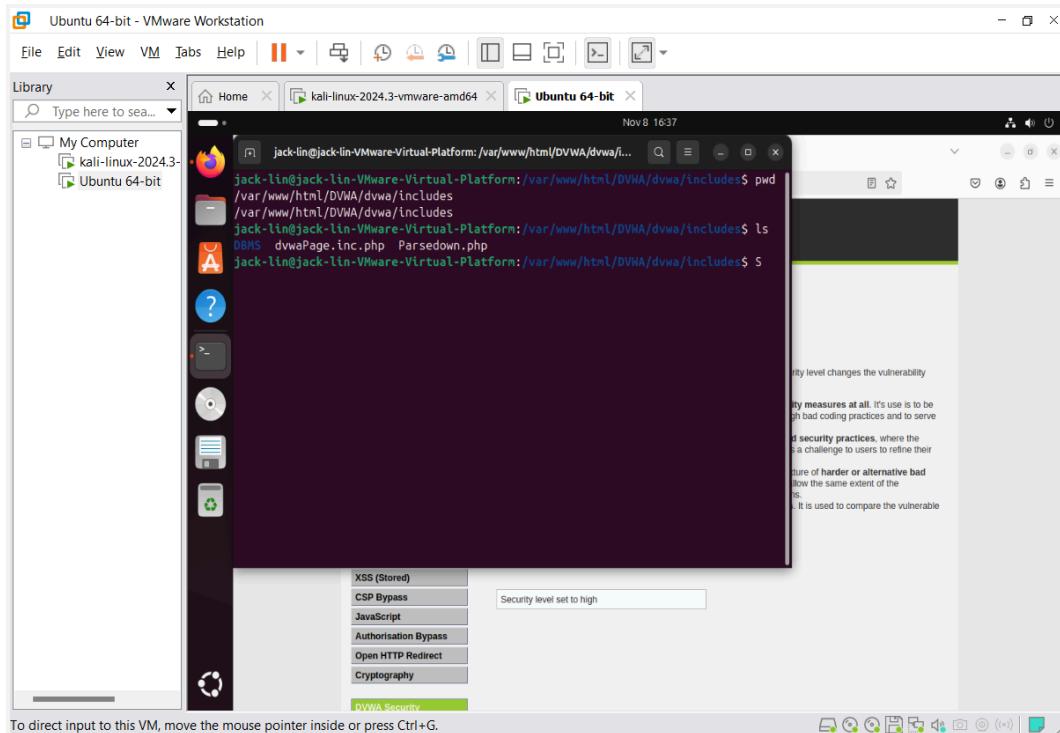
The security level in DVWA is now set to high.

Part 7: Exploring the source code of DVWA

Step 1: Locate the DVWA Directory

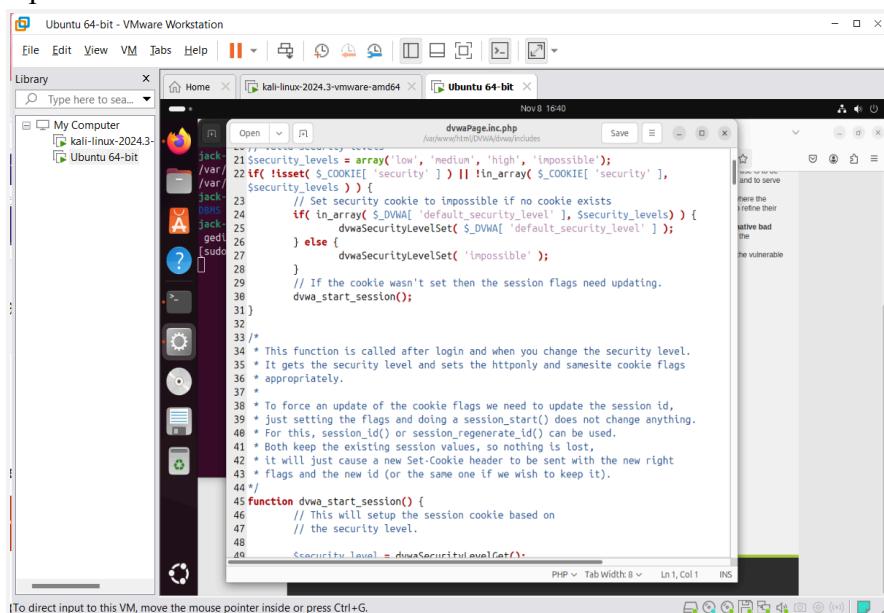
Step 2: Find the Security Level File

After changing to the correct directory (`/var/www/html/DVWA`), we can see the file that controls the security level: `dvwa/includes/dvwaPage.inc.php`



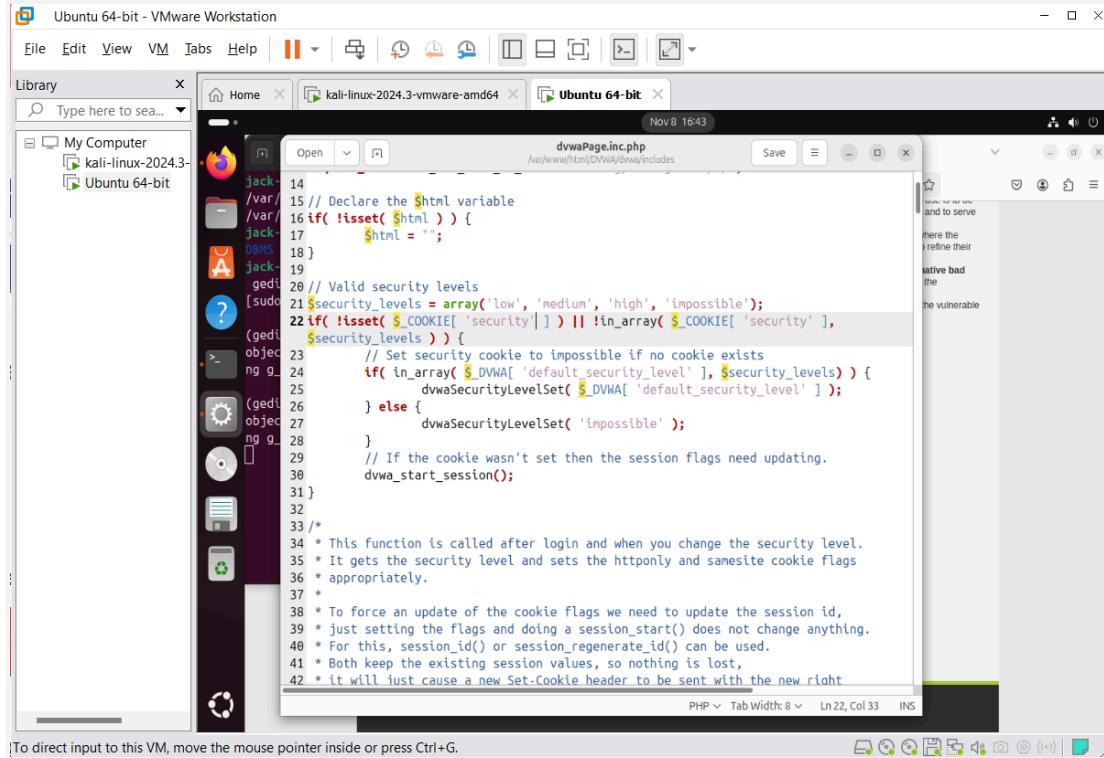
Step 3: Check dvwaPage.inc.php for Security Level Code

Open the File:



Search for the Security Level Logic: Look for the following lines or similar code within this file:

```
$security_level = isset($_SESSION['security'])? $_SESSION['security'] : 'impossible';
```



```
14 // Declare the $html variable
15 if( !isset( $html ) ) {
16     $html = "";
17 }
18 }
19
20 // Valid security levels
21 $security_levels = array( 'low', 'medium', 'high', 'impossible' );
22 if( !isset( $_COOKIE[ 'security' ] ) || !in_array( $_COOKIE[ 'security' ],
23     $security_levels ) ) {
24     // Set security cookie to impossible if no cookie exists
25     if( in_array( $_DWA[ 'default_security_level' ], $security_levels ) ) {
26         dwaSecurityLevelSet( $_DWA[ 'default_security_level' ] );
27     } else {
28         dwaSecurityLevelSet( 'impossible' );
29     }
30     // If the cookie wasn't set then the session flags need updating.
31     dwa_start_session();
32 }
33 /*
34 * This function is called after login and when you change the security level.
35 * It gets the security level and sets the httponly and samesite cookie flags
36 * appropriately.
37 *
38 * To force an update of the cookie flags we need to update the session id,
39 * just setting the flags and doing a session_start() does not change anything.
40 * For this, session_id() or session_regenerate_id() can be used.
41 * Both keep the existing session values, so nothing is lost,
42 * it will just cause a new Set-Cookie header to be sent with the new right
```

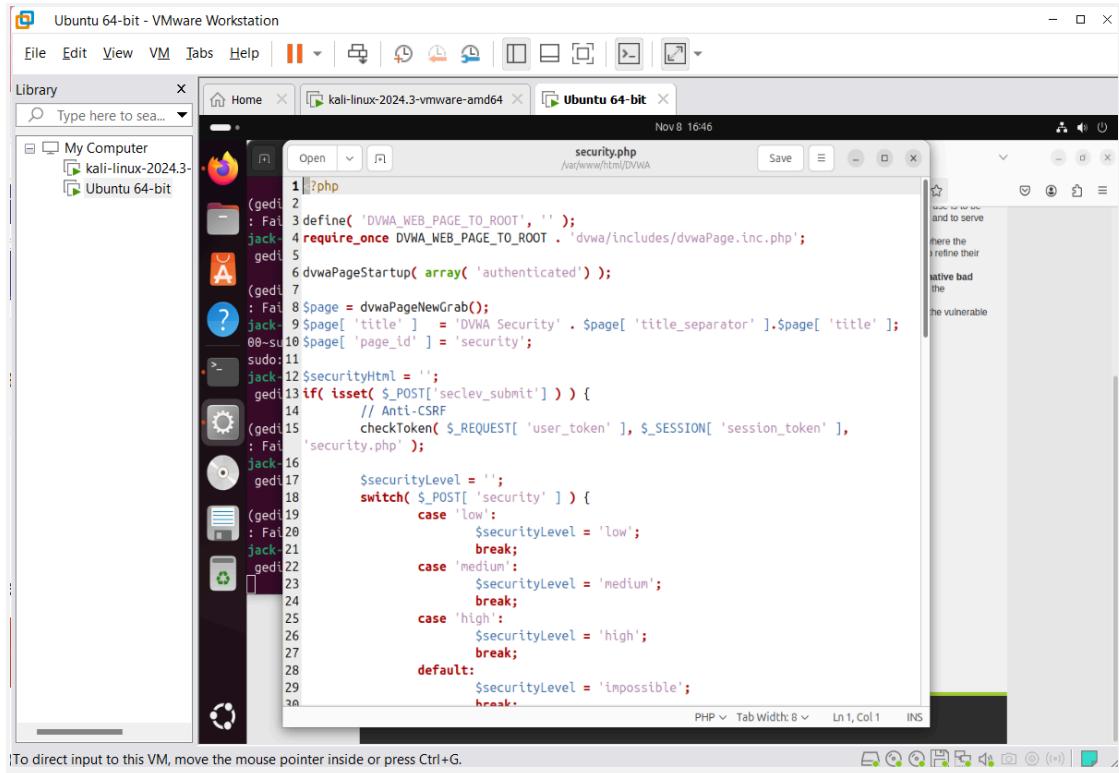
This code sets the default security level to impossible if the security level is not otherwise set.

Identify the Security Levels:

DVWA defines several security levels (low, medium, high, impossible) that are stored in PHP sessions. Changing the security level in the web interface updates this session variable.

Step 4: Explore security.php for Security Level Setting

Open security.php:



```
Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help | Home | kali-linux-2024.3-vmware-amd64 | Ubuntu 64-bit | Nov 8 16:46
Library Type here to search... security.php /var/www/html/DVWA
1 //php
2
3 define( 'DVWA_WEB_PAGE_TO_ROOT', '' );
4 require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';
5
6 dvwaPageStartup( array( 'authenticated' ) );
7
8 $page = dvwaPageNewGrab();
9 $page[ 'title' ] = 'DVWA Security' . $page[ 'title_separator' ].$page[ 'title' ];
10 $page[ 'page_id' ] = 'security';
11
12 $SecurityHTML = '';
13 if( isset( $_POST['seclev_submit'] ) ) {
14     // Anti-CSRF
15     checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ],
16     'security.php' );
17
18     $SecurityLevel = '';
19     switch( $_POST[ 'security' ] ) {
20         case 'low':
21             $SecurityLevel = 'low';
22             break;
23         case 'medium':
24             $SecurityLevel = 'medium';
25             break;
26         case 'high':
27             $SecurityLevel = 'high';
28             break;
29         default:
30             $SecurityLevel = 'impossible';
31             break;
32     }
33
34     header('Location: ' . $_SERVER['SCRIPT_NAME']);
35     exit;
36 }
```

This code determines the security level for the session.

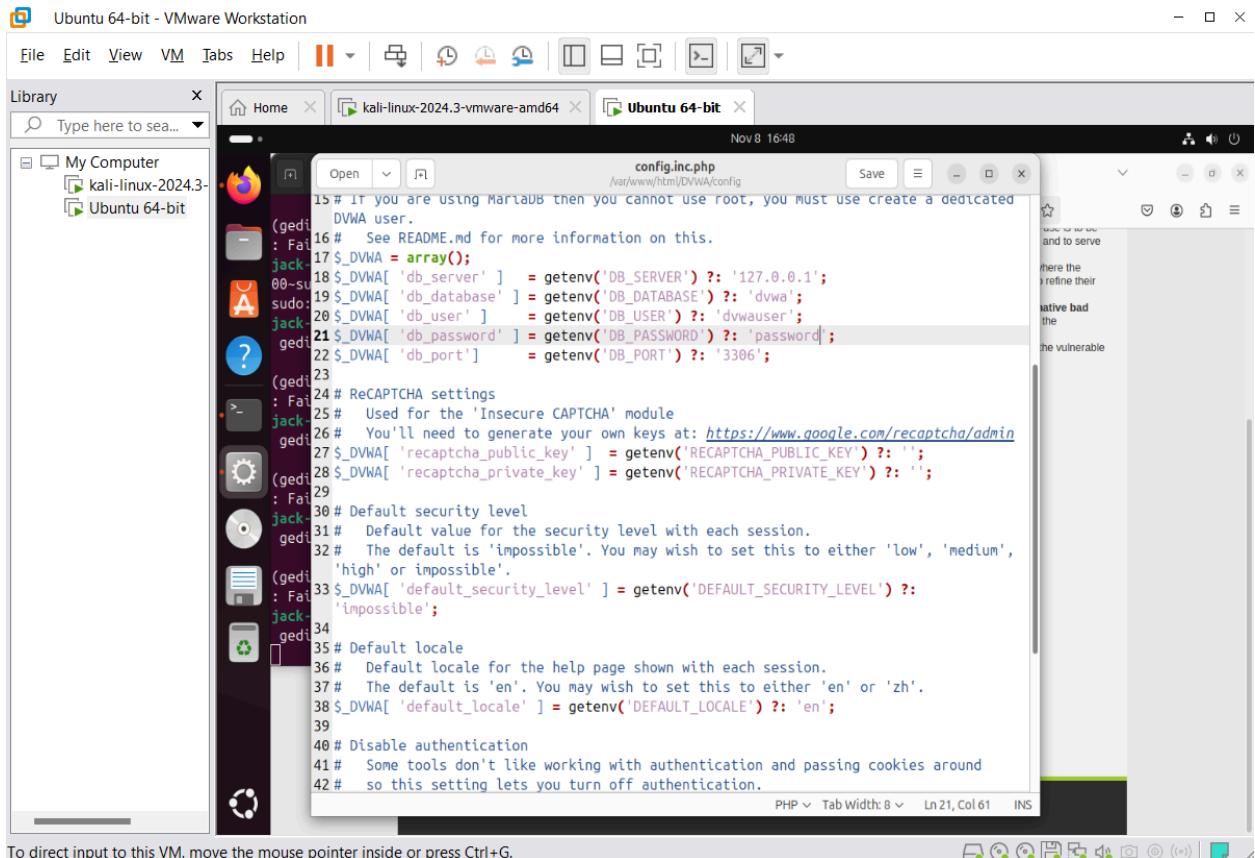
Examine the Code: Look for the form handler code that saves the selected security level to the session:

```
if(isset($_POST['security']) ) {
    $_SESSION['security']= $_POST['security'];
    header('Location: ' . $_SERVER['SCRIPT_NAME']);
    exit;
}
```

This code checks if the form to change the security level has been submitted. If so, it updates the session variable `$_SESSION['security']` with the new value selected by the user (low, medium, high, or impossible).

Step 5: Review config.inc.php for Default Security Level

Open the Config File:



```
config.inc.php
/var/www/html/DVWA/config
Nov 8 16:48
15 # If you are using MariaDB then you cannot use root, you MUST use create a dedicated DVWA user.
16 # See README.md for more information on this.
17 $_DVWA = array();
18 $_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
19 $_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
20 $_DVWA[ 'db_user' ] = getenv('DB_USER') ?: 'dvwauser';
21 $_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'password';
22 $_DVWA[ 'db_port' ] = getenv('DB_PORT') ?: '3306';

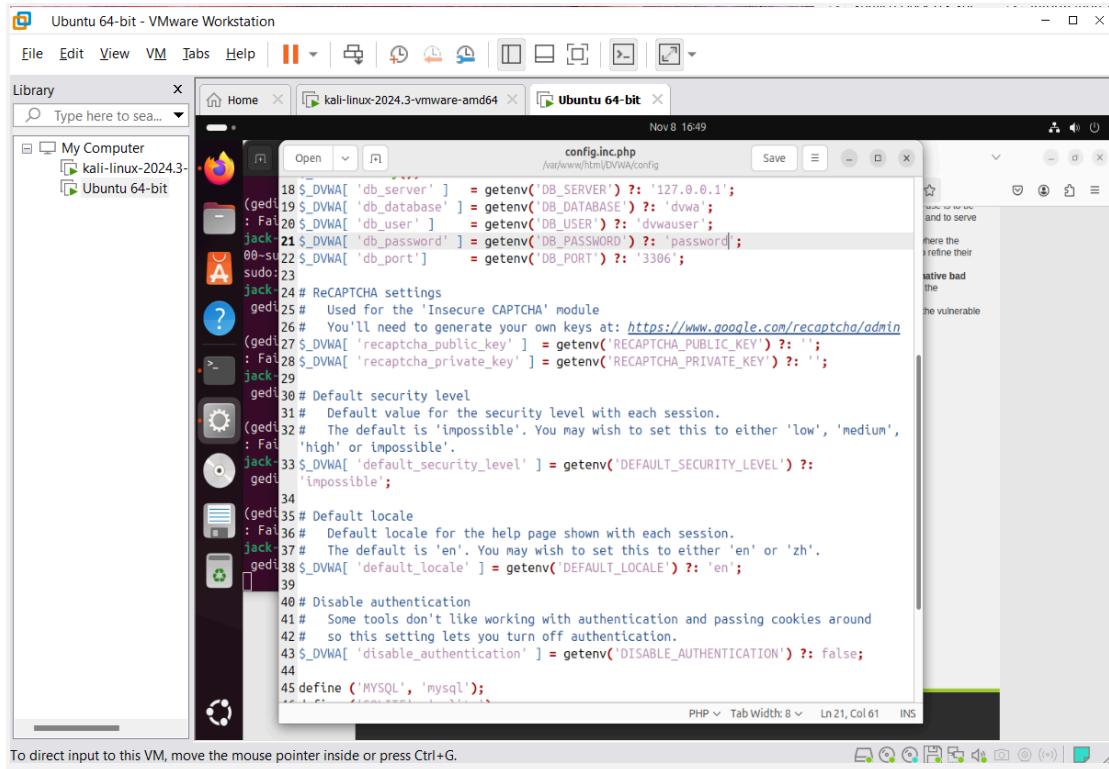
23
24 # ReCAPTCHA settings
25 # Used for the 'Insecure CAPTCHA' module
26 # You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
27 $_DVWA[ 'recaptcha_public_key' ] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
28 $_DVWA[ 'recaptcha_private_key' ] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';
29
30 # Default security level
31 # Default value for the security level with each session.
32 # The default is 'impossible'. You may wish to set this to either 'low', 'medium',
# 'high' or 'impossible'.
33 $_DVWA[ 'default_security_level' ] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';
34
35 # Default locale
36 # Default locale for the help page shown with each session.
37 # The default is 'en'. You may wish to set this to either 'en' or 'zh'.
38 $_DVWA[ 'default_locale' ] = getenv('DEFAULT_LOCALE') ?: 'en';
39
40 # Disable authentication
41 # Some tools don't like working with authentication and passing cookies around
42 # so this setting lets you turn off authentication.
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Our configuration file looks slightly different than it did before because these settings worked better for the virtual machine we were using.

Look for Security Level:

In this file, you may find a line that sets the default security level:



```
config.inc.php
/var/www/html/DVWA/config
Nov 8 16:49

18 $_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
19 $_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dwa';
20 $_DVWA[ 'db_user' ] = getenv('DB_USER') ?: 'dwawuser';
21 $_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'password';
22 $_DVWA[ 'db_port' ] = getenv('DB_PORT') ?: '3306';
23
24 # ReCAPTCHA settings
25 # Used for the 'Insecure CAPTCHA' module
26 # You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
27 $_DVWA[ 'recaptcha_public_key' ] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
28 $_DVWA[ 'recaptcha_private_key' ] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';
29
30 # Default security level
31 # Default value for the security level with each session.
32 # The default is 'impossible'. You may wish to set this to either 'low', 'medium',
33 $_DVWA[ 'default_security_level' ] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';
34
35 # Default locale
36 # Default locale for the help page shown with each session.
37 # The default is 'en'. You may wish to set this to either 'en' or 'zh'.
38 $_DVWA[ 'default_locale' ] = getenv('DEFAULT_LOCALE') ?: 'en';
39
40 # Disable authentication
41 # Some tools don't like working with authentication and passing cookies around.
42 # so this setting lets you turn off authentication.
43 $_DVWA[ 'disable_authentication' ] = getenv('DISABLE_AUTHENTICATION') ?: false;
44
45 define ('MYSQL', 'mysql');
```

Our default security level is currently set to impossible but can be easily changed.

Conclusions:

DVWA is an intentionally vulnerable web application that allows users to explore different types of attacks. In this project, we explored SQL injection, a vulnerability in weak web applications where user input is executed as part of a query and can be manipulated to display restricted database information. SQL injection can be prevented with secure coding practices, such as using parameterized queries so that user input is not executed as code, escaping user input so that users cannot input executable scripts, input validation to ensure users are only entering necessary input, and by using ORM or stored procedures that ensure SQL queries are abstracted. This project shows an example of an application with low security being vulnerable to SQL injection and how SQL injection can be prevented in that same application along with different logs and configuration files. The security level can also be easily adjusted in the configuration files for DVWA specifically.