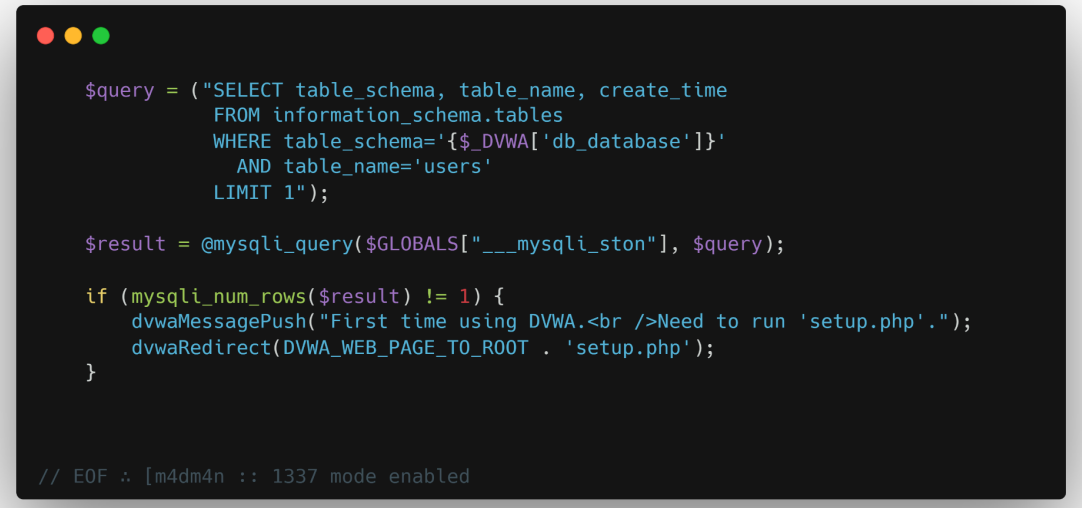
```php
if (isset($_POST['Login'])) {

    // Anti-CSRF token retrieval from session
    $session_token = $_SESSION['session_token'] ?? "";

    // CSRF protection — verifies the submitted token matches session token
    checkToken($_REQUEST['user_token'], $session_token, 'login.php');

// EOF ∴ [m4dm4n :: 1337 mode enabled
```

✅ **Secure Reasoning**: Prevents CSRF attacks by verifying the submitted token against a session-stored token.

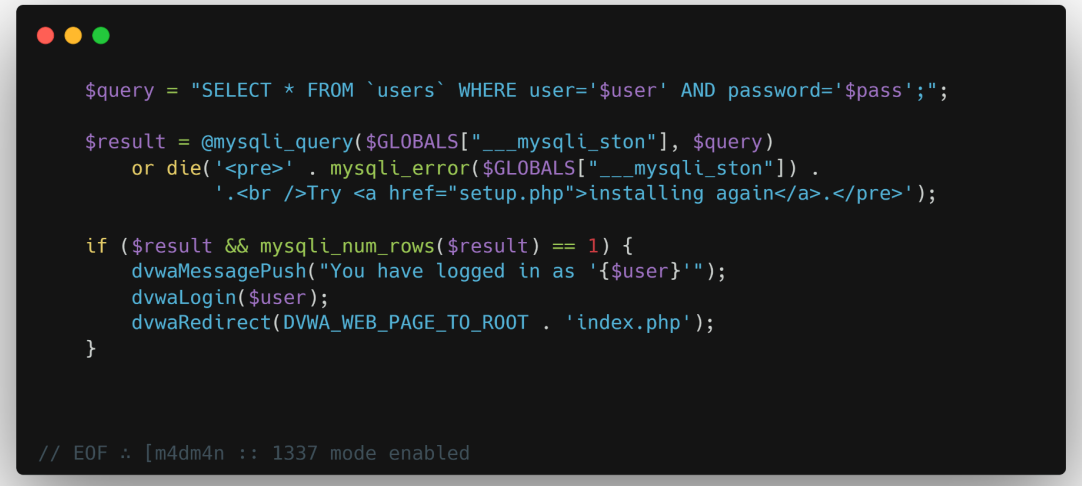
```php
    // Extract and clean the username
    $user = stripslashes($_POST['username']);
    $user = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $user);

    // Extract and clean the password
    $pass = stripslashes($_POST['password']);
    $pass = mysqli_real_escape_string($GLOBALS["___mysqli_ston"], $pass);

    // Hashing the password ( ⚠ insecure method: MD5)
    $pass = md5($pass);


// EOF ∴ [m4dm4n :: 1337 mode enabled
```

⚠️ **Weak Hash**: MD5 is insecure — use `password_hash()` and `password_verify()` instead.
✅ **Escape**: Prevents basic SQL injection via escaping, but still vulnerable to logic flaws — **use prepared statements instead**.
Prevents **script injection via XSS** in login page by No unescaped user input in output



```
    $query = ("SELECT table_schema, table_name, create_time
              FROM information_schema.tables
              WHERE table_schema='{$_DVWA['db_database']}'
                AND table_name='users'
              LIMIT 1");

    $result = @mysqli_query($GLOBALS["___mysqli_ston"], $query);

    if (mysqli_num_rows($result) != 1) {
        dvwaMessagePush("First time using DVWA.<br />Need to run 'setup.php'.");
        dvwaRedirect(DVWA_WEB_PAGE_TO_ROOT . 'setup.php');
    }



// EOF ∴ [m4dm4n :: 1337 mode enabled
```

✅ **Safe UX Logic**: Verifies if the system is properly initialized before attempting login.



```
    $query = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";

    $result = @mysqli_query($GLOBALS["___mysqli_ston"], $query)
        or die('<pre>' . mysqli_error($GLOBALS["___mysqli_ston"]) .
               '.<br />Try <a href="setup.php">installing again</a>.</pre>');

    if ($result && mysqli_num_rows($result) == 1) {
        dvwaMessagePush("You have logged in as '{$user}'");
        dvwaLogin($user);
        dvwaRedirect(DVWA_WEB_PAGE_TO_ROOT . 'index.php');
    }



// EOF ∴ [m4dm4n :: 1337 mode enabled
```

⚠️ **Security Concern**:

- Raw SQL query is still vulnerable to logical attacks despite escaping.
- `die()` leaks internal errors to the user — dangerous in production.

✅ **Session Handling**: `dvwaLogin()` likely sets session securely.

```
    // Fallback for incorrect login attempt
    dvwaMessagePush('Login failed');
    dvwaRedirect('login.php');
}


// EOF ∴ [m4dm4n :: 1337 mode enabled
```

✅ **User Feedback**: Prevents username enumeration by using generic message.

```
$messagesHtml = messagesPopAllToHtml();

// No-cache headers
Header('Cache-Control: no-cache, must-revalidate');
Header('Content-Type: text/html;charset=utf-8');
Header('Expires: Tue, 23 Jun 2009 12:00:00 GMT');

// Generate CSRF token for login form
generateSessionToken();



// EOF ∴ [m4dm4n :: 1337 mode enabled
```

✅ **Security**: Disables caching and generates CSRF token server-side.

```php
<form action="login.php" method="post">
    <fieldset>
        <label for="user">Username</label>
        <input type="text" class="loginInput" name="username"><br />

        <label for="pass">Password</label>
        <input type="password" class="loginInput" name="password" autocomplete="off">
<br />

        <p class="submit">
            <input type="submit" value="Login" name="Login">
        </p>
    </fieldset>

    <?php echo tokenField(); ?> <!-- 🔐 CSRF token input -->
</form>



// EOF ∴ [m4dm4n :: 1337 mode enabled
```

✅ **Secure Form**: Embeds CSRF token into form. Password field disables autocomplete.

Feature	Description
CSRF Protection	Session token + token validation in `checkToken()` and `tokenField()`
SQL Injection Mitigation	Uses `mysqli_real_escape_string()` (better to use prepared statements)
Session Security	`dvwaLogin()` likely handles secure session cookies
Caching Controls	Prevents sensitive data caching in browser/proxies
Autocomplete Off	Password input disables browser remembering credentials

Issue	Description
❌ **MD5 Hashing**	MD5 is outdated and broken; use bcrypt or Argon2 with salt
❌ **Raw SQL Queries**	Escaping is used, but still vulnerable to logic flaws. Use **prepared statements**.
❌ **Error Leakage**	Raw DB error messages shown to user (`die()`) reveals internals
❌ **No Rate Limiting**	Brute-force login protection not enforced
❌ **No CAPTCHA**	Doesn't defend against automated login attempts
❌ **No Logging**	Failed login attempts aren't logged (no audit trail)
❌ **No Account Lockout**	No lockout mechanism after X failed attempts

How to Make it Production-Grade Secure

- Use **`password_hash()`** and **`password_verify()`** instead of MD5.
- Use **prepared statements** (`mysqli_prepare` / PDO).
- Implement **rate-limiting and account lockouts**.
- Apply **logging** for failed login attempts.
- Replace raw error messages with **generic failure messages**.
- Use **Content Security Policy (CSP)**, `HttpOnly`, `Secure`, and `SameSite` flags on session cookies.
- Add **CAPTCHA** to deter bots.
- Implement **2FA (Two-Factor Authentication)**.