

Actividad 1: Explotación y Mitigación de Broken Authentication

Tema: Ataques contra autenticación débil

Objetivo: Explorar cómo una mala autenticación permite acceso no autorizado y aplicar soluciones

¿Qué es Broken Authentication?

Broken Authentication ocurre cuando un atacante puede eludir o forzar los mecanismos de autenticación debido a debilidades en la implementación del sistema. Esto puede incluir credenciales débiles, almacenamiento inseguro de contraseñas, gestión inadecuada de sesiones y falta de protección contra ataques de fuerza bruta.

Creamos la BD de prueba

Iniciar el servicio *MySQL*

Antes de acceder a MySQL, asegurarse de que el servicio está en ejecución: `sudo systemctl start mysql`

Si se desea verificar el estado del servicio: `sudo systemctl status mysql`

Acceder a MySQL como *root*

Si MySQL está en ejecución, iniciar sesión con: `sudo mysql -u root`

Si MySQL tiene una contraseña configurada, usar: `sudo mysql -u root -p`

Seguidamente ingresar la contraseña cuando se solicite.

Crear la base de datos

```
CREATE DATABASE testdb;
```

Usar la base de datos

```
USE testdb;
```

Crear la tabla de usuarios

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50) UNIQUE NOT NULL,  
    password VARCHAR(50) NOT NULL  
);
```

Insertar usuarios de prueba

```
INSERT INTO users (username, password) VALUES
('admin', '123456');

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('root');
```

Verificar que los datos se han insertado correctamente

```
SELECT * FROM users;

EXIT;
```

Código Vulnerable: *login_weak.php*

Crear el archivo **vulnerable** *login_weak.php*. El código contiene varias vulnerabilidades que pueden ser explotadas para realizar ataques de autenticación rota.

```
<?php

$conn = new mysqli("localhost", "root", "root", "testdb");

if ($conn->connect_error) {
    die("Error de conexión: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST" || $_SERVER["REQUEST_METHOD"] == "GET") {
    $username = $_REQUEST["username"];
    $password = $_REQUEST["password"];

    print("Usuario: " . $username . "<br>");
    print("Contraseña: " . $password . "<br>");

    $query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
    print("Consulta SQL: " . $query . "<br>");

    $result = $conn->query($query);

    if ($result->num_rows > 0) {
        echo "Inicio de sesión exitoso";
    } else {
        echo "Usuario o contraseña incorrectos";
    }
}

?>
```

Iniciar el servicio *Apache*

Antes de acceder la página web, asegurarse de que el servicio está en ejecución:

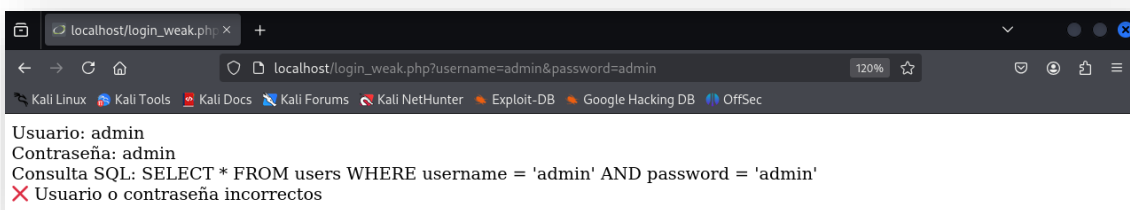
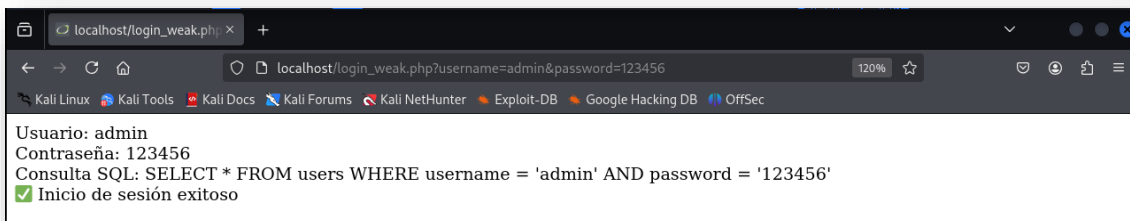
```
sudo systemctl start apache2
```

Acceder a la página web con el siguiente enlace:

```
http://localhost/login_weak.php?username=admin&password=123456
```

Vulnerabilidades del código:

1. *Inyección SQL*: La consulta SQL usa variables sin validación, lo que permite ataques de inyección.
2. *Uso de contraseñas en texto plano*: No se usa hashing para almacenar las contraseñas, lo que facilita su robo en caso de acceso a la base de datos.
3. *Falta de control de intentos de inicio de sesión*: No hay mecanismos de protección contra ataques de fuerza bruta.
4. *Falta de gestión segura de sesiones*: No se generan tokens de sesión seguros tras un inicio de sesión exitoso.



Explotación de Broken Authentication

Si el usuario root de MySQL no tiene una contraseña asignada, estableced una para evitar posibles inconvenientes al trabajar con MySQL.

Ataque de fuerza bruta con Hydra

Si el sistema no tiene un límite de intentos fallidos, se puede usar Hydra para adivinar contraseñas:

```
gunzip /usr/share/wordlists/rockyou.txt.gz

hydra -l admin -P /usr/share/wordlists/rockyou.txt localhost http-post-form
"/login_weak.php:username=^USER^&password=^PASS^:Usuario o contraseña incorrectos" -V
```

Explicación de los parámetros:

- *http-post-form*: Indica que estás atacando un formulario de autenticación con método POST.
- *"/login_weak.php:username=^USER^&password=^PASS^:Fallo"*:
 - */login_weak.php* → Ruta de la página de inicio de sesión.

- `username=^USER^&password=^PASS^` → Parámetros que se envían en la solicitud POST. Hydra reemplazará `^USER^` y `^PASS^` con los valores de la lista de usuarios y contraseñas.
- *Fallo* → Texto que aparece en la respuesta cuando el inicio de sesión falla. Se puede cambiar por el mensaje real de error que muestra la página cuando una contraseña es incorrecta (por ejemplo, *"Usuario o contraseña incorrectos"*).

```
(root@kali) ~ # hydra -l admin -P /usr/share/wordlists/rockyou.txt localhost http-post-form "/login_weak.php:username='USER'&password='PASS':Usuario o contraseña incorrectos" -V
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-24 04:52:54
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://localhost:80/login_weak.php:username='USER'&password='PASS':Usuario o contraseña incorrectos
[ATTEMPT] target localhost - login "admin" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "12345" - 2 of 14344399 [child 1] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "123456789" - 3 of 14344399 [child 2] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "password" - 4 of 14344399 [child 3] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "iloveyou" - 5 of 14344399 [child 4] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "princess" - 6 of 14344399 [child 5] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "1234567" - 7 of 14344399 [child 6] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "rockyou" - 8 of 14344399 [child 7] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "12345678" - 9 of 14344399 [child 8] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "abc123" - 10 of 14344399 [child 9] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "nicole" - 11 of 14344399 [child 10] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "daniel" - 12 of 14344399 [child 11] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "babygirl" - 13 of 14344399 [child 12] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "monkey" - 14 of 14344399 [child 13] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "lovely" - 15 of 14344399 [child 14] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "jessica" - 16 of 14344399 [child 15] (0/0)
[80][http-post-form] host: localhost login: admin password: 123456
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-24 04:52:56
```

Explotación de SQL Injection

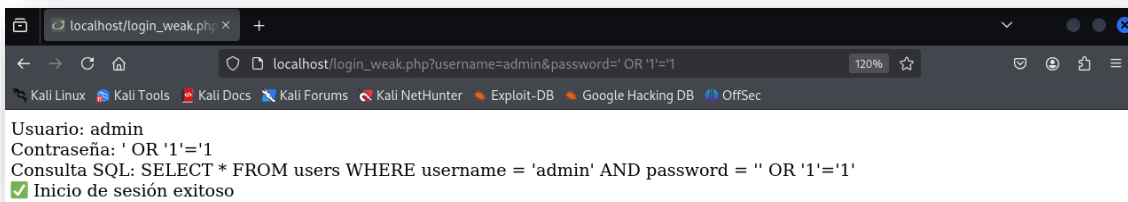
El atacante puede intentar un *payload malicioso* en el campo de contraseña:

```
username: admin
password: ' OR '1'='1
```

Esto convertiría la consulta en:

```
SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1'='1';
```

Debido a que `'1'='1'` es siempre *verdadero*, el atacante obtendría acceso.



localhost/login_weak.php: x +

localhost/login_weak.php?username=admin&password=' OR '1'='1' 120%

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

Usuario: admin
Contraseña: ' OR '1'='1
Consulta SQL: SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1'='1'
✓ Inicio de sesión exitoso

Mitigación: Código Seguro en PHP

* Uso de contraseñas cifradas con password_hash

La función password_hash() con PASSWORD_BCRYPT genera un hash de hasta **60 caracteres**, y con PASSWORD_ARGON2ID, incluso más (hasta 255). Por eso, se necesita que la columna pueda almacenarlos adecuadamente.

Sentencia SQL para modificar la tabla:

```
ALTER TABLE users MODIFY password VARCHAR(255) NOT NULL;
```

Esto permite almacenar cualquier hash generado con los algoritmos recomendados en PHP.

Insertar usuarios con contraseña 'hasheada'

A partir de ahora, no se puede seguir insertando contraseñas en texto plano ('123456'). Se debe usar password_hash() en PHP al registrar al usuario.

Ejemplo de código para registrar un usuario 'raul' con password '123456': *add_users.php*

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

// Conexión a la base de datos
$conn = new mysqli("localhost", "root", "root", "testdb");

if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}

// Usuario de prueba
$username = "raul";
$password = "123456";
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Inserción
$stmt = $conn->prepare("INSERT INTO users (username, password) VALUES (?, ?)");
$stmt->bind_param("ss", $username, $hashed_password);

if ($stmt->execute()) {
    echo "Usuario insertado correctamente";
} else {
    echo "Error al insertar usuario: " . $stmt->error;
}

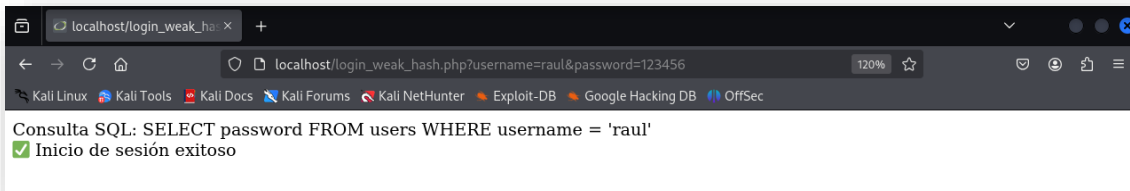
$stmt->close();
$conn->close();
?>
```

PASSWORD_DEFAULT usa actualmente BCrypt, pero se actualizará automáticamente en versiones futuras de PHP. Si deseas más control, puedes usar PASSWORD_BCRYPT o PASSWORD_ARGON2ID.

```
MariaDB [testdb]> select * from users;
```

id	username	password
1	admin	123456
2	raul	\$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtb1rSsu1GFV5/tPa0

```
2 rows in set (0.000 sec)
```



<?php

```
$conn = new mysqli("localhost", "root", "root", "testdb");

if ($conn->connect_error) {
    die("Error de conexión: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST" || $_SERVER["REQUEST_METHOD"] == "GET") {
    $username = $_REQUEST["username"];
    $password = $_REQUEST["password"];

    // NO PREVENIMOS SQL INJECTION, SOLO SE AGREGA PASSWORD_HASH
    $query = "SELECT password FROM users WHERE username = '$username'";
    print("Consulta SQL: " . $query . "<br>");

    $result = $conn->query($query);

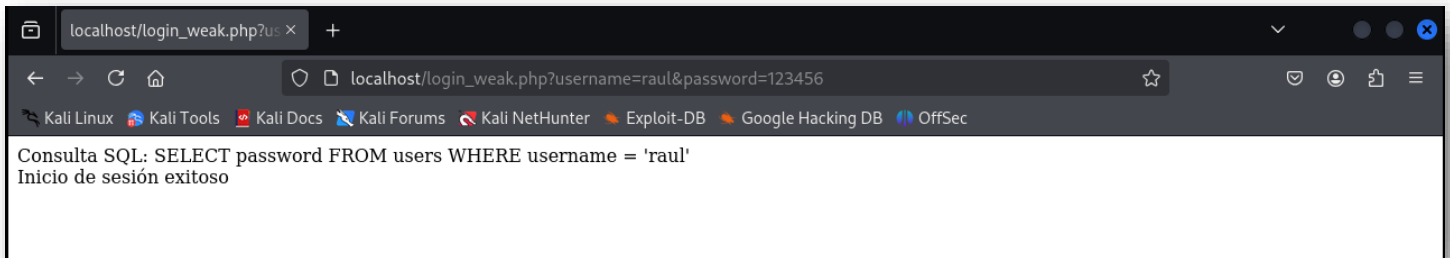
    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
        $hashed_password = $row["password"];

        // Verificación de contraseña hasheada
        if (password_verify($password, $hashed_password)) {
            echo "Inicio de sesión exitoso";
        } else {
            echo "Usuario o contraseña incorrectos";
        }
    } else {
        echo "Usuario no encontrado";
    }
}
```

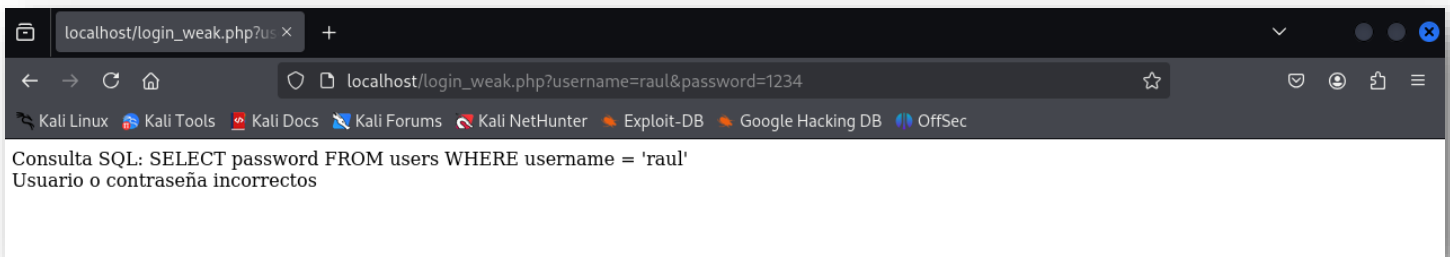
```
$conn->close();  
?>
```

Acceder a la página web con el siguiente enlace:

http://localhost/login_weak.php?username=raul&password=123456



http://localhost/login_weak.php?username=raul&password=1234



* Uso de consultas preparadas para prevenir SQL Injection

```
<?php  
  
$conn = new mysqli("localhost", "root", "root", "testdb");  
  
// Verificar conexión  
if ($conn->connect_error) {  
    die("Error de conexión: " . $conn->connect_error);  
}  
  
// Solo permitir POST o GET  
if ($_SERVER["REQUEST_METHOD"] == "POST" || $_SERVER["REQUEST_METHOD"] == "GET") {  
    $username = $_REQUEST["username"];  
    $password = $_REQUEST["password"];  
  
    // Usar sentencias preparadas para evitar inyección SQL  
    $query = "SELECT password FROM users WHERE username = ?";
```

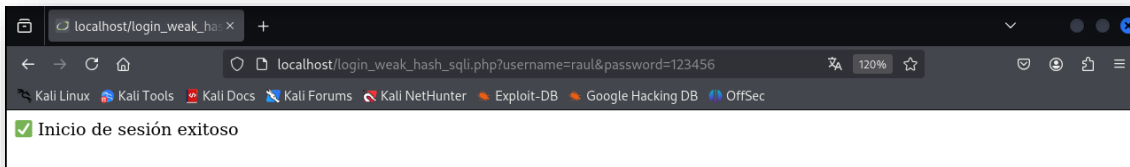
```
$stmt = $conn->prepare($query);
$stmt->bind_param("s", $username);
$stmt->execute();
$stmt->store_result();

// Verificar si el usuario existe
if ($stmt->num_rows > 0) {
    $stmt->bind_result($hashed_password);
    $stmt->fetch();

    // Verificar el password hasheado
    if (password_verify($password, $hashed_password)) {
        echo "Inicio de sesión exitoso";
    } else {
        echo "Usuario o contraseña incorrectos";
    }
} else {
    echo "Usuario no encontrado";
}

$stmt->close();
}

$conn->close();
?>
```



* Implementar bloqueo de cuenta tras varios intentos fallidos

Para bloquear la cuenta después de **3 intentos fallidos**, podemos hacer lo siguiente:

1. Añadir un campo *failed_attempts* en la base de datos para contar los intentos fallidos.
2. Registrar el *timestamp* del último intento fallido con un campo *last_attempt* para poder restablecer los intentos después de un tiempo.
3. Modificar la lógica del login:
 - o Si el usuario tiene 3 intentos fallidos, *bloquear la cuenta*.
 - o Si han pasado, por ejemplo, 15 minutos desde el último intento, *restablecer los intentos fallidos*.
 - o Si el login es exitoso, *reiniciar los intentos fallidos a 0*.

Modificación en la Base de Datos

Ejecuta este SQL para agregar los campos en la tabla *users*:

```
ALTER TABLE users ADD failed_attempts INT DEFAULT 0;
ALTER TABLE users ADD last_attempt TIMESTAMP NULL DEFAULT NULL;
```



```
Database changed
MariaDB [testdb]> ALTER TABLE users ADD failed_attempts INT DEFAULT 0;
Query OK, 0 rows affected (0.013 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [testdb]> ALTER TABLE users ADD last_attempt TIMESTAMP NULL DEFAULT NULL;
Query OK, 0 rows affected (0.012 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [testdb]> select * from users;
+-----+-----+-----+-----+-----+
| id | username | password | failed_attempts | last_attempt |
+-----+-----+-----+-----+-----+
| 1 | admin | 123456 | 0 | NULL |
| 5 | raul | $2y$10$rDVGu6SumOhqEnoM6Ppa0.zPMUxh5sDw1Jb.Xyloupc/er59tkK52 | 0 | NULL |
+-----+-----+-----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [testdb]>
```

Código PHP Mejorado con 'Bloqueo de Cuenta' tras 3 intentos fallidos

```
<?php

$conn = new mysqli("localhost", "root", "root", "testdb");

// Verificar conexión
if ($conn->connect_error) {
    die("Error de conexión: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST" || $_SERVER["REQUEST_METHOD"] == "GET") {
    $username = $_REQUEST["username"];
    $password = $_REQUEST["password"];

    // Buscar usuario y su estado de intentos fallidos
    $query = "SELECT password, failed_attempts, last_attempt FROM users WHERE username = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $stmt->bind_result($hashed_password, $failed_attempts, $last_attempt);
        $stmt->fetch();
        $stmt->close();

        // Verificar si la cuenta está bloqueada
        $current_time = time();
        $lockout_duration = 900; // 15 minutos (900 segundos)

        if ($failed_attempts >= 3) {
            $last_attempt_time = strtotime($last_attempt);
            if (($current_time - $last_attempt_time) < $lockout_duration) {
                die("Cuenta bloqueada. Inténtalo de nuevo más tarde.");
            } else {
                // Reiniciar intentos fallidos después de 15 minutos
            }
        }
    }
}
```

```
$reset_attempts_query = "UPDATE users SET failed_attempts = 0 WHERE
username = ?";
$stmt = $conn->prepare($reset_attempts_query);
$stmt->bind_param("s", $username);
$stmt->execute();
$stmt->close();
$failed_attempts = 0;
}
}

// Verificación de contraseña
if (password_verify($password, $hashed_password)) {
    echo "Inicio de sesión exitoso";

    // Reiniciar intentos fallidos en caso de éxito
    $reset_attempts_query = "UPDATE users SET failed_attempts = 0 WHERE
username = ?";
    $stmt = $conn->prepare($reset_attempts_query);
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->close();
} else {
    // Aumentar intentos fallidos
    $failed_attempts++;
    $update_attempts_query = "UPDATE users SET failed_attempts = ?,
last_attempt = NOW() WHERE username = ?";
    $stmt = $conn->prepare($update_attempts_query);
    $stmt->bind_param("is", $failed_attempts, $username);
    $stmt->execute();
    $stmt->close();

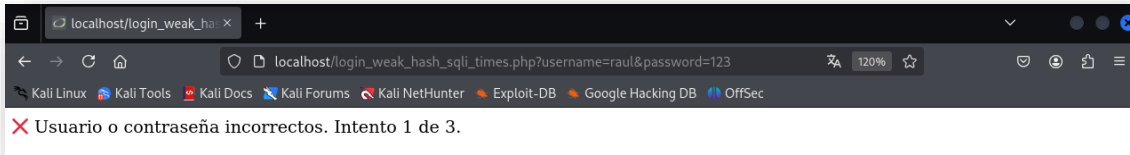
    if ($failed_attempts >= 3) {
        die("Cuenta bloqueada por demasiados intentos. Inténtalo de nuevo en
15 minutos.");
    }

    echo "Usuario o contraseña incorrectos. Intento $failed_attempts de 3.";
}
} else {
    echo "Usuario no encontrado";
}
}

$conn->close();
?>
```

¿Cómo Funciona?

1. Verifica si el usuario está bloqueado (*failed_attempts* >= 3).
 - Si el último intento fue hace menos de 15 minutos, bloquea la cuenta.
 - Si han pasado más de 15 minutos, restablece los intentos a 0.
2. Si el login es exitoso, reinicia los intentos fallidos.
3. Si el login falla, incrementa los intentos fallidos y bloquea la cuenta en el intento 3.



```
Database changed
MariaDB [testdb]> select * from users;
```

| id | username | password | failed_attempts | last_attempt |
|----|----------|---|-----------------|--------------|
| 1 | admin | 123456 | 0 | NULL |
| 2 | raul | \$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtblrSsu1GFV5/tPa0 | 0 | NULL |

```
2 rows in set (0.001 sec)

MariaDB [testdb]> select * from users;
```

| id | username | password | failed_attempts | last_attempt |
|----|----------|---|-----------------|---------------------|
| 1 | admin | 123456 | 0 | NULL |
| 2 | raul | \$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtblrSsu1GFV5/tPa0 | 1 | 2025-02-24 05:27:56 |

```
2 rows in set (0.001 sec)
```

```
MariaDB [testdb]> select * from users;
```

| id | username | password | failed_attempts | last_attempt |
|----|----------|---|-----------------|---------------------|
| 1 | admin | 123456 | 0 | NULL |
| 2 | raul | \$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtblrSsu1GFV5/tPa0 | 1 | 2025-02-24 05:27:56 |

```
2 rows in set (0.001 sec)

MariaDB [testdb]> select * from users;
```

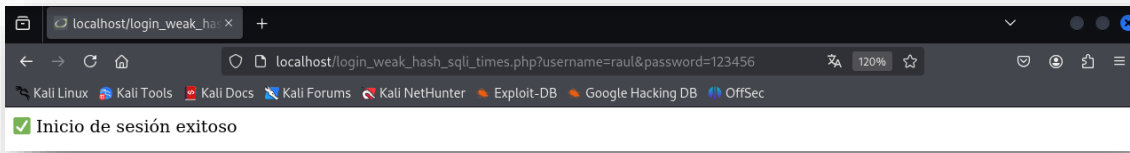
| id | username | password | failed_attempts | last_attempt |
|----|----------|---|-----------------|---------------------|
| 1 | admin | 123456 | 0 | NULL |
| 2 | raul | \$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtblrSsu1GFV5/tPa0 | 2 | 2025-02-24 05:28:41 |

```
2 rows in set (0.001 sec)

MariaDB [testdb]> select * from users;
```

| id | username | password | failed_attempts | last_attempt |
|----|----------|---|-----------------|---------------------|
| 1 | admin | 123456 | 0 | NULL |
| 2 | raul | \$2y\$10\$N35bt3uLFQ9IYv/uQieIDeZhn8jJLa5zQ6nqtblrSsu1GFV5/tPa0 | 0 | 2025-02-24 05:28:41 |

```
2 rows in set (0.001 sec)
```



* Implementar autenticación multifactor (MFA)

Para añadir MFA (*Autenticación Multifactor*) al sistema de login, seguiremos estos pasos:

Pasos para Implementar MFA

1. Generar un código de verificación temporal (OTP) de 6 dígitos.
2. Enviar el código OTP al usuario mediante correo electrónico o SMS (en este caso, usaremos correo simulado, ya que almacenará el código generado en un fichero txt).
3. Crear un formulario para que el usuario ingrese el código OTP después de iniciar sesión.
4. Verificar el código OTP antes de permitir el acceso.

Modificación en la Base de Datos

Agregar una nueva columna para almacenar el OTP temporal y su tiempo de expiración:

```
ALTER TABLE users ADD otp_code VARCHAR(6) NULL;
ALTER TABLE users ADD otp_expires TIMESTAMP NULL DEFAULT NULL;
ALTER TABLE users ADD email VARCHAR(255) NULL;
```

```
MariaDB [testdb]> select * from users;
+----+-----+-----+-----+-----+-----+
| id | username | password | failed_attempts | last_attempt | otp_code | otp_expires | email |
+----+-----+-----+-----+-----+-----+-----+
| 1 | admin | 123456 | 0 | NULL | NULL | NULL | NULL |
| 5 | raul | $2y$10$rDVGu6Sum0hqEnoM6Ppa0.zPMUxh5sDw1Jb.Xyloupc/er59tkK52 | 1 | 2025-03-25 11:57:40 | NULL | NULL | NULL |
+----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [testdb]>
```

Código PHP Mejorado con MFA (*Login con generación de OTP login_otp.php*)

```
<?php

$conn = new mysqli("localhost", "root", "root", "testdb");

// Verificar conexión
if ($conn->connect_error) {
    die("Error de conexión: " . $conn->connect_error);
}
if ($_SERVER["REQUEST_METHOD"] == "POST" || $_SERVER["REQUEST_METHOD"] == "GET") {
    // Permite GET y POST
    $username = $_REQUEST["username"];
```

```

$password = $_REQUEST["password"];

// Buscar usuario en la BD
$query = "SELECT id, password, email, failed_attempts, last_attempt FROM users WHERE
username = ?";
$stmt = $conn->prepare($query);
$stmt->bind_param("s", $username);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    $stmt->bind_result($user_id, $hashed_password, $email, $failed_attempts,
$last_attempt);
    $stmt->fetch();
    $stmt->close();

    if (password_verify($password, $hashed_password)) {
        // Generar código OTP de 6 dígitos
        $otp = mt_rand(100000, 999999);
        $otp_expires = date('Y-m-d H:i:s', strtotime('+10 minutes')); // Expira en 10
min

        // Guardar OTP en la BD
        $update_otp = "UPDATE users SET otp_code = ?, otp_expires = ? WHERE id = ?";
        $stmt = $conn->prepare($update_otp);
        $stmt->bind_param("ssi", $otp, $otp_expires, $user_id);
        $stmt->execute();
        $stmt->close();

        // Enviar OTP por correo
        enviarCorreo($email, $otp);

        // Redirigir al formulario de verificación OTP
        session_start();
        $_SESSION['user_id'] = $user_id;
        header("Location: verificar_otp.php");
        exit();
    } else {
        echo "Contraseña incorrecta";
    }
} else {
    echo "Usuario no encontrado";
}
}

$conn->close();

// Función para enviar correo con OTP
function enviarCorreo($email, $otp) {
    $to = $email;
    $subject = "Tu código OTP de acceso";
    $message = "Tu código de verificación es: $otp. Expira en 10 minutos.";
    $headers = "From: no-reply@tusitio.com";

    //mail($to, $subject, $message, $headers); // Se comenta para que no de errores si
no esta configurado el servidor SMTP
    // Simula el envío en fichero
    $logFile = "emails_simulados.txt";
    $message_file = "[" . date('Y-m-d H:i:s') . "] To: $email\nOTP: $otp\n\n";
    file_put_contents($logFile, $message_file, FILE_APPEND);
    echo "OTP guardado en emails_simulados.txt";
}
?>

```

Verificación de OTP (*verificar_otp.php*)

Verificamos el código OTP ingresado por el usuario.

```
<?php

session_start();
$conn = new mysqli("localhost", "root", "root", "testdb");

// Verificar conexión
if ($conn->connect_error) {
    die("Error de conexión: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user_id = $_SESSION['user_id'];
    $otp_ingresado = $_POST["otp"];

    // Obtener el OTP almacenado en la BD
    $query = "SELECT otp_code, otp_expires FROM users WHERE id = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $user_id);
    $stmt->execute();
    $stmt->bind_result($otp_almacenado, $otp_expires);
    $stmt->fetch();
    $stmt->close();

    // Verificar OTP y su expiración
    if ($otp_almacenado == $otp_ingresado && strtotime($otp_expires) > time()) {
        echo "Autenticación exitosa. Bienvenido!";
        // Opcional: Redirigir al panel de usuario
        // header("Location: dashboard.php");
    } else {
        echo "Código OTP inválido o expirado.";
    }
}

$conn->close();
?>

<!-- Formulario HTML para ingresar OTP -->
<form method="POST">
    <label for="otp">Código OTP:</label>
    <input type="text" name="otp" required>
    <button type="submit">Verificar</button>
</form>
```

Crear fichero **emails_simulados.txt** que almacenará el código OTP simulando el envío del correo

```
sudo touch emails_simulados.txt

sudo chmod 777 emails_simulados.txt
```

Acceder a la página web con el siguiente enlace:

```
http://localhost/login_weak.php?username=raul&password=123456
```

localhost/verificar_otp.php

Código OTP: Verificar

```
(root@kali)-[/var/www/html]
# cat emails_simulados.txt
To: rfuentes2002@gmail.com
OTP: 609845
```

```
MariaDB [testdb]> select * from users;
```

id	username	password	failed_attempts	last_attempt	otp_code	otp_expires	email
1	admin	123456	0	NULL	NULL	NULL	
2	raul	\$2y\$10\$N35bt3uLFQ9IVv/uQieIDeZhm8jJLa5zQ6nqtblrSsu1gFV5/tPa0	0	2025-02-24 05:28:41	609845	2025-02-24 11:46:32	rfuentes2002@gmail.com

2 rows in set (0.001 sec)

localhost/verificar_otp.php

✗ Código OTP inválido o expirado.

Código OTP: Verificar

localhost/verificar_otp.php

✓ Autenticación exitosa. Bienvenido!

Código OTP: Verificar

Si accedemos de nuevo observamos como cambia el Código OTP

```
MariaDB [testdb]> select * from users;
```

id	username	password	failed_attempts	last_attempt	otp_code	otp_expires	email
1	admin	123456	0	NULL	NULL	NULL	
2	raul	\$2y\$10\$N35bt3uLFQ9IVv/uQieIDeZhm8jJLa5zQ6nqtblrSsu1gFV5/tPa0	0	2025-02-24 05:28:41	802555	2025-02-24 11:40:37	rfuentes2002@gmail.com

2 rows in set (0.001 sec)

Flujo completo del *Login con MFA*

1. Usuario ingresa su usuario y contraseña.
2. Si las credenciales son correctas, se genera un código OTP y se guarda en la BD.
3. Se envía el código OTP al usuario por correo electrónico (fichero emails_simulados.txt).
4. Usuario ingresa el código OTP en un formulario.
5. El sistema verifica si el código es válido y no ha expirado.
6. Si es correcto, el usuario accede; si no, se muestra un error.

Beneficios de este *Sistema MFA*

- Mayor seguridad contra accesos no autorizados.
- Protege contra ataques de fuerza bruta, incluso si la contraseña es robada.
- Fácil de extender a SMS o aplicaciones como *Google Authenticator*.