Vulnerability Report

Archivo: index.php

Code Analyzed:

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css">


    <title>Hello, world!</title>
</head>

<body>
    <script>
        function eliminar(){
            var respuesta  = confirm("Estas seguro que deseas eliminar");
            return respuesta;
        }
    </script>
    <div class="container-fluid-row ">

        <div class="row px-4">
            <h1>Prueba Tecnica</h1>

            <div class="col-4">
                <div class="card">
                    <div class="card-body">
                        <form action="" method="post">
                            <!-- Mostrar Error     -->
                            <?php
                            include "models/conexion.php";
                            include "controller/registro.php";

                            ?>
                            <h3>Registro de Persona</h3>
                            <div class="mb-3">
                                <label for="nombre" class="form-label">Nombre</label>
                                <input type="text" class="form-control" id="nombre"
name="nombre">

                            </div>
                            <div class="mb-3">
                                <label for="apellido" class="form-label">Apellido</label>
                                <input type="text" class="form-control" id="apellido"
name="apellido">
                            </div>
                            <div class="mb-3">
                                <label for="dni" class="form-label">Cedula</label>
                                <input type="text" class="form-control" id="dni" name="dni">
                            </div>
                            <div class="mb-3">
                                <label for="fecha" class="form-label">Fecha Nacimiento</label>
                                <input type="date" class="form-control" id="fecha"
name="fecha">
                            </div>
                            <div class="mb-3">
                                <label for="email" class="form-label">Email</label>
                                <input type="email" class="form-control" id="email"
name="email">
                            </div>
                            <button type="submit" name="btnregistrar" class="btn btn-success"
value="ok">Registrar</button>
```

```
                        </form>
                    </div>
                </div>
            </div>
            <div class="col-8">
                <?php include "controller/eliminar.php"; ?>
                <table class="table table-success table-striped">
                    <thead>
                        <tr>
                            <th scope="col">#</th>
                            <th scope="col">Nombre</th>
                            <th scope="col">Apellido</th>
                            <th scope="col">Cedula</th>
                            <th scope="col">Fecha Nacimiento</th>
                            <th scope="col">Correo</th>
                            <th scope="col">Acciones</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php
                        include("models/conexion.php");
                        $registrosPorPagina = 5;
                        $pagina = isset($_GET['pagina']) ? (int)$_GET['pagina'] : 1;
                        if ($pagina < 1) $pagina = 1;

                        $offset = ($pagina - 1) * $registrosPorPagina;

                        // Total de registros y páginas
                        $resultadoTotal = $conexion->query("SELECT COUNT(*) AS total FROM
personas");
                        $totalRegistros = $resultadoTotal->fetch_object()->total;
                        $totalPaginas = ceil($totalRegistros / $registrosPorPagina);

                        // Consulta paginada
                        $sql = $conexion->query("SELECT * FROM personas LIMIT
$registrosPorPagina OFFSET $offset");
                        while ($a = $sql->fetch_object()) { ?>
                            <tr>
                                <td><?= $a->id ?></td>
                                <td><?= htmlspecialchars($a->nombre) ?></td>
                                <td><?= htmlspecialchars($a->apellido) ?></td>
                                <td><?= htmlspecialchars($a->cedula) ?></td>
                                <td><?= htmlspecialchars($a->fecha_nacimiento) ?></td>
                                <td><?= htmlspecialchars($a->correo) ?></td>
                                <td>
                                    <a class="btn btn-warning" href="edit.php?id=<?= $a->id
?>"><i class="fa-solid fa-pencil"></i></a>

                                    <a onclick="return eliminar()" class="btn btn-danger"
href="index.php?id=<?= $a->id ?>"><i class="fa-solid fa-trash"></i></a>
                                </td>
                            </tr>
                        <?php } ?>
                    </tbody>
                </table>
                <!-- Paginación -->
                <nav>
                    <ul class="pagination">
                        <?php if ($pagina > 1): ?>
                            <li class="page-item"><a class="page-link" href="?pagina=<?=
$pagina - 1 ?>">Anterior</a></li>
                        <?php endif; ?>

                        <?php for ($i = 1; $i <= $totalPaginas; $i++): ?>
                            <li class="page-item <?= $i == $pagina ? 'active' : '' ?>">
                                <a class="page-link" href="?pagina=<?= $i ?>"><?= $i ?></a>
                            </li>
                        <?php endfor; ?>

                        <?php if ($pagina < $totalPaginas): ?>
                            <li class="page-item"><a class="page-link" href="?pagina=<?=
$pagina + 1 ?>">Siguiente</a></li>
                        <?php endif; ?>
                    </ul>
                </nav>
            </div>
        </div>
    </div>

    <!-- Optional JavaScript; choose one of the two! -->
```

```
    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
    -->
</body>

</html>
```

Analysis: ```html
Security Vulnerabilities

SQL Injection

Approximate Line: The included files "controller/eliminar.php" and "controller/registro.php" are likely vulnerable. Also, any query within "models/conexion.php" using unsanitized user input. Specifically, the delete functionality and data insertion.

Description: If the `eliminar.php` script uses the `$_GET['id']` parameter directly in a SQL query without proper sanitization/parameterization, it's vulnerable to SQL injection. Similarly, `registro.php` is vulnerable if it's not sanitizing the POST data. An attacker could manipulate the ID or other input fields to execute arbitrary SQL commands.

Mitigation: Use prepared statements with parameterized queries for all database interactions. Never directly embed user input into SQL queries. For example, using PDO:

```
$stmt = $pdo->prepare("SELECT * FROM personas WHERE id = ?");
$stmt->execute([$_GET['id']]);
$data = $stmt->fetch();
```

Code Quality Improvement: Introduce dependency injection to manage the database connection. This reduces coupling and improves testability.

Security Vulnerabilities

Cross-Site Scripting (XSS)

Approximate Line: Although `htmlspecialchars()` is used, it's not consistently applied everywhere.

Description: If data retrieved from the database (e.g., `nombre`, `apellido`, `correo`) contains malicious JavaScript, it could be executed in the user's browser when displayed in the table.

Mitigation: Ensure all data retrieved from the database and displayed in the HTML is properly escaped using `htmlspecialchars()` with the correct encoding (e.g., `ENT_QUOTES`, `UTF-8`). Consider using a templating engine which automatically escapes output.

Code Quality Improvement: Employ a templating engine like Twig or Blade which can handle escaping by default, increasing readability and consistency.

Security Vulnerabilities

Missing Input Validation

Approximate Line: The registration form inputs.

Description: There's no server-side validation on the form data (name, email, DNI, date). An attacker can submit invalid or malicious data, potentially causing issues in the database or application logic.

Mitigation: Implement server-side validation for all form inputs. Check for data types, lengths, formats (e.g., email validity, date format), and acceptable ranges. Use a validation library to simplify this process.

Code Quality Improvement: Centralize validation logic into reusable functions or classes. This improves maintainability and reduces code duplication.

Security Vulnerabilities

Information Disclosure

Approximate Line: Pagination logic.

Description: While not a direct security vulnerability, the page displays all records even if there is an attempt to navigate outside of the range of records. This can be improved by validating that a user doesn't enter a page # that would cause an error or simply return no results.

Mitigation: Implement a redirect or display an error message to prevent accessing invalid page numbers. Limit the range of pagination links displayed. Validate user-provided values like $_GET['pagina'] to ensure they are within the expected range.

Code Quality Improvement: Abstract the pagination logic into a reusable component.

Code Quality Metrics

Complexity

The PHP code within the HTML mixes presentation logic with database interaction. This increases complexity and reduces readability.

Improvement: Separate the business logic from the presentation using the MVC (Model-View-Controller) pattern or a similar architectural pattern. Use a templating engine for the view layer.

Coupling

The code has high coupling between different parts (e.g., HTML, PHP, database). Changes in one part might require changes in other parts.

Improvement: Use interfaces and abstract classes to reduce dependencies between components. Dependency injection is beneficial here.

Duplication

The database connection code is likely duplicated in multiple files (`conexion.php` is included in `registro.php` and the main page).

Improvement: Create a single, reusable function or class to manage the database connection. Use a database abstraction layer (like PDO) to further reduce dependencies.

Readability

The code can be difficult to read and understand due to the mixing of HTML and PHP code and the lack of clear separation of concerns.

Improvement: Follow coding standards (e.g., PSR standards), use meaningful variable names, add comments to explain complex logic, and break down large functions into smaller, more manageable ones. Use a code formatter/linter to enforce consistent style. Using a templating engine greatly improves this.

Proposed Solution

1. **Implement MVC pattern:** Separate presentation, data access, and business logic.
2. **Use Prepared Statements:** Protect against SQL injection.
3. **Input Validation:** Validate data on both client and server sides.
4. **Output Encoding:** Use `htmlspecialchars()` or a templating engine with auto-escaping.
5. **Error Handling:** Implement proper error handling and logging.
6. **Dependency Injection:** Manage dependencies for better testability and maintainability.
7. **Code Standards:** Adhere to coding standards (PSR) for readability.

```
```