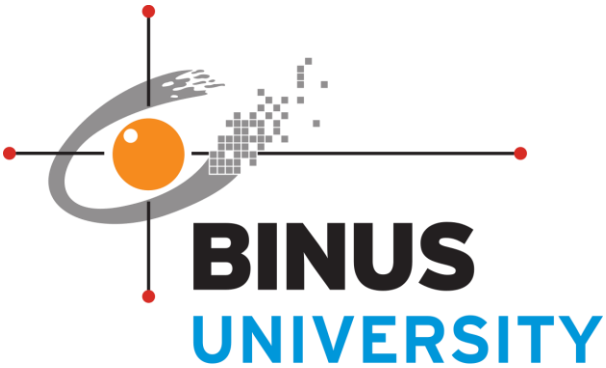


Code Audit Secure Programming Website CyberCourse

Secure Programming - Code Auditing Report
December 16th 2024

Prepared By
Axel Kuarsani - 2602054713
Daniel Rafael Ayorbaba - 2540120723
Muhammad Fariz Alfaresa - 2502018013
Matthew Devensen - 2602080251
Raditya Rahman Maulana - 2602213593
Aminah Kaityln Latifah - 2602203144
Binus University
Kelompok 1

Executive Summary

Dalam rangka memenuhi tugas akhir dari mata kuliah Secure Programming, kami ditugaskan untuk melakukan code auditing untuk mencari celah keamanan dari codingan yang dibuat oleh kelompok lain serta memberikan solusi terhadap celah keamanan yang ditemukan. Kami dari kelompok 1 mendapatkan kesempatan untuk melakukan code audit dari kelompok 2 dengan judul project mereka CyberCourse.

CyberCourse merupakan sebuah platform pembelajaran online yang terdiri dari dua roles yaitu student dan tutor. Student hanya dapat mengakses courses dan mereka bisa menjadi tutor dengan cara mendaftarkan diri pada menu Teachers. Jika student mendaftarkan diri untuk menjadi salah satu teachers, maka student dapat memilih salah satu profesi seperti designer, developer, dan sebagainya.

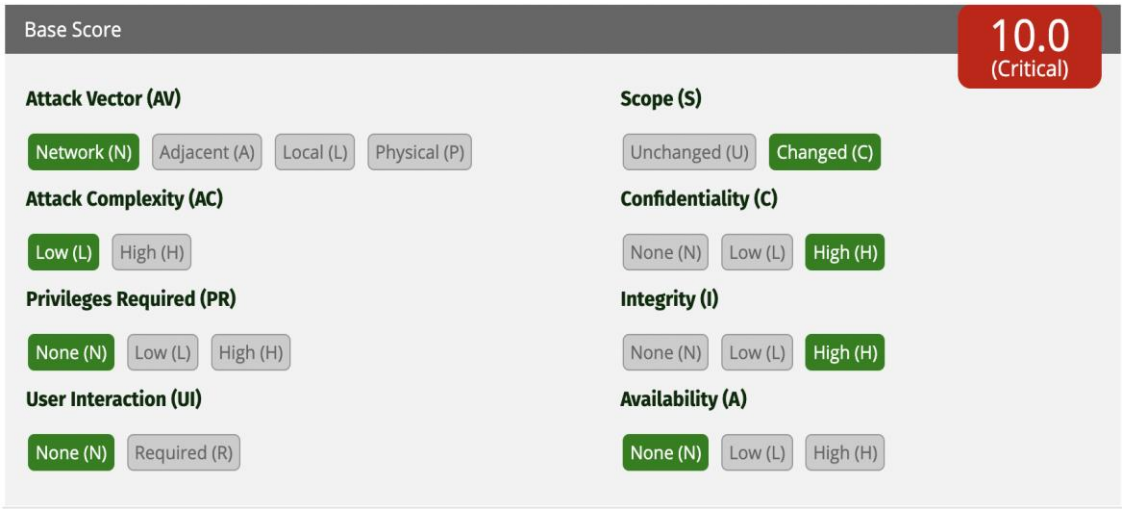
Selain itu, user dengan role student dapat mengakses video yang telah disediakan pada menu courses, jika student suka video yang diberikan maka mereka bisa memberikan rating serta comment pada video tersebut. Jika user telah mendapatkan role sebagai tutor *(mari kita berasumsi bahwa user memilih profesi sebagai Developer/Lecturer)* maka ia dapat mengakses admin dashboard sebagai developer serta membuat playlist dan mengupload video materi yang telah dibuat.

Setelah kami melakukan static analysis terhadap project yang telah dibuat oleh kelompok 2, jenis serangan yang ditemukan yaitu Cookie Tampering, IDOR (Insecure Direct Object Reference), XSS, CSRF Token, Low Password Security, dan File Upload. Tujuh model serangan tersebut ditemukan hampir pada semua code dari CyberCourse dan total vulnerability yang kami temukan yaitu sebanyak 32. Setelah itu, kami melakukan scoring severity menggunakan CVSS v3.1.

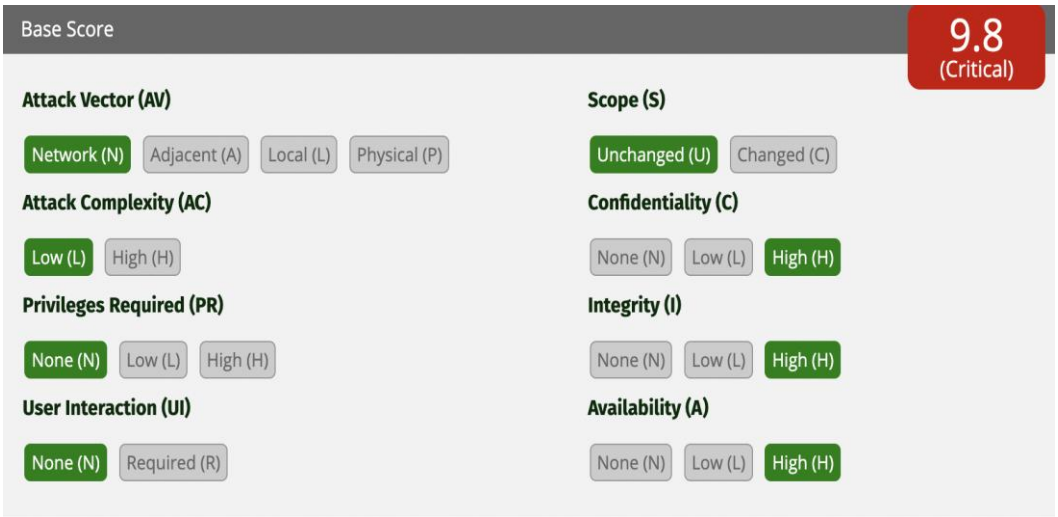
Code Vulnerabilities

Berisi dari vulnerability yang ditemukan serta lokasi ditemukannya vulnerability serta severity score dari setiap attack:

- **Cookie Tampering**
  - Severity score:



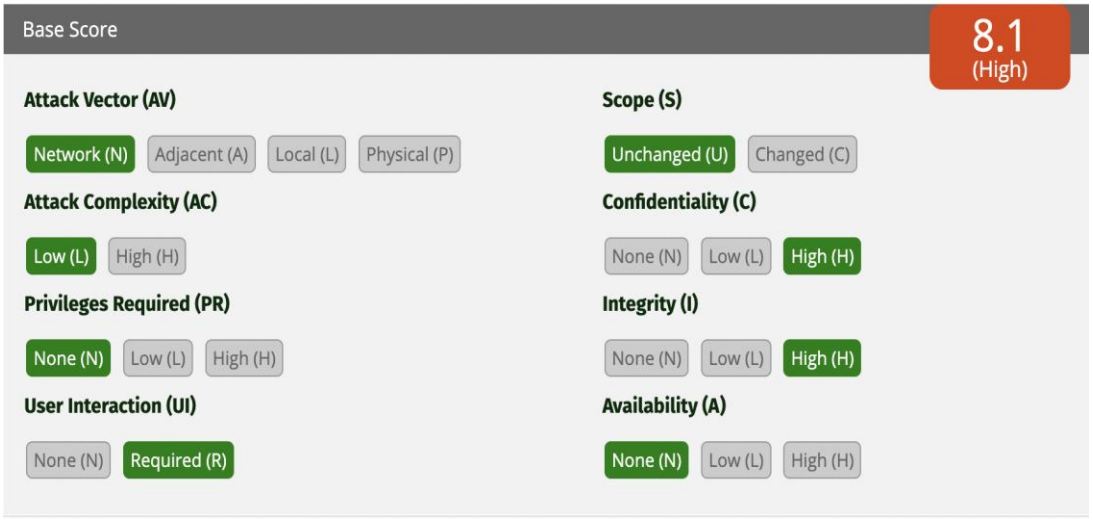
  - Location: All of the files, except:
    - Login user and admin
    - Register user and admin
    - user and admin headers
- **IDOR**
  - Severity score:



  - Location:
    - Courses 53
    - Home 143
    - Likes 80
    - Playlist 133
    - Search Course 54

- Tutor Profile 99
- Watch Video 145,193
- Comments Admin 58
- Contents Admin 79, 82
- Playlist Admin 83, 86
- Search Page Admin 110, 113, 158, 161
- Update Content Admin 177
- Update Playlist Admin 125
- VIew Content Admin 104
- View Playlist 99, 132

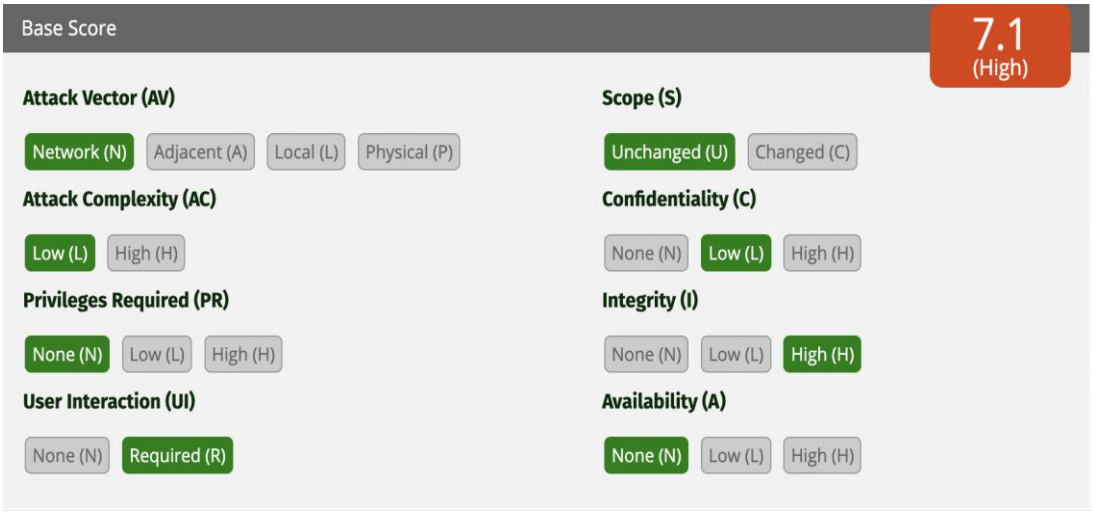
- **XSS**
  - Severity score:



  - Location:
    - Bookmark 51-52
    - Comments 104
    - Courses 47-48
    - Home 137-138, 142
    - Likes 72-73, 77
    - Playlist 104-105, 109-110, 136
    - Profile 46
    - Search Course 48-49, 53
    - Search Tutor 68-69
    - Teachers 71-72
    - Tutor Profile 62-63, 93,98
    - Update user 113-115
    - Watch Video 179, 187-188,239-240,243
    - Add Content Admin 90
    - Comments Admin 58
    - Dashboard Admin 50
    - Playlist Admin 79-80
    - Profile Admin 49-50
    - Search Page Admin 107, 154-155

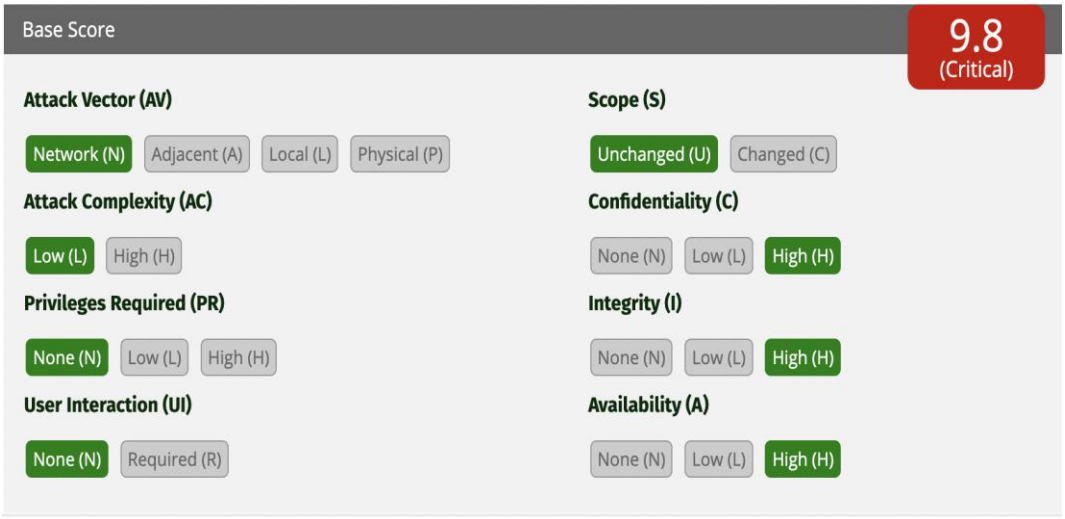
- Update Content Admin 136
- Update Playlist Admin 113,115
- Update Admin 120, 137
- View Content 95, 100,135-136,141
- View Playlist 94,96,129
- Admin Header 39-40,76-77
- User Header 38, 74

- **CSRF Token**
  - Severity score:

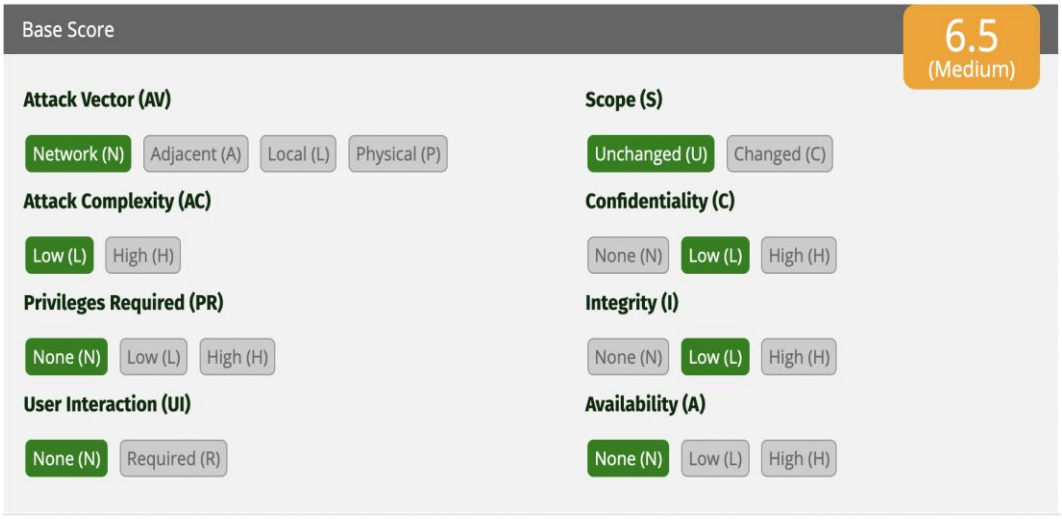


  - Location:
    - Comments 73-80, 108-112
    - Contact 56-63
    - Likes 78-82
    - Login User 61-69
    - Playlist 80-93
    - Register User 80-100
    - Teachers 79-82
    - Update user 108-129
    - Watch Video 141-148, 191-205,218-222,247-251
    - Add Content Admin 68-106
    - Add Playlist Admin 55-70
    - Comments Admin 60-63
    - Contents Admin 77-81
    - Login Admin 68-76
    - Playlist Admin 81-85
    - Register Admin 76-113
    - Search Page Admin 108-112, 156-160
    - Update Content Admin 134-180
    - Update Playlist Admin 104-127
    - View Content Admin 140-143,
    - View Playlist Admin 97-101, 130-134
    - Admin Header 18-21

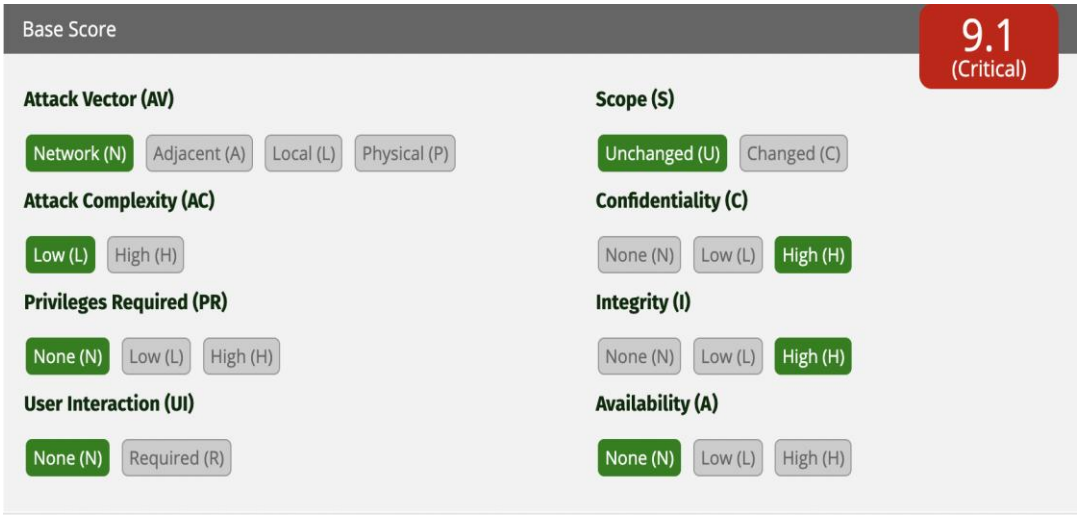
    - User Header 18-21
- **SQL Injection**
    - Severity score:



    - Location:
        - Search Course 33
        - Search Tutor 40
- **Low Password Security**
    - Severity score:



    - Location:
        - Register User 18, 20
        - Register Admin 14 16
- **File Upload**
    - Severity score:



- Location:
  - Register User 32-33
  - Update user 47-48
  - Add Content Admin 27-28,35-36
  - Add Playlist Admin 21-22
  - Register Admin 19-20
  - Update Content Admin 39-47, 62-69
  - Update Playlist Admin 33-39

Explanation

Cookie Tampering

Sebuah serangan yang terjadi karena cookie yang diberi oleh browser tidak aman. Penemuan

```
if($select_user->rowCount() > 0){
    setcookie(name: 'user_id', value: $row['id'], expires_or_options: time() + 60*60*24*30, path: '/');
    header(header: 'location:home.php');
} else {
```

Karena cookie yang tidak diberi secure, dan HTTP only flag cookie dapat diubah, Ini berdampak sangat besar karena dalam aplikasi CyberCourse terdapat banyak penggunaan cookie seperti:

```
if(isset($_COOKIE['user_id'])){
   $user_id = $_COOKIE['user_id'];
}else{
   $user_id = '';
}
```

ada juga yang digunakan untuk sebuah query ke dalam database

```
<div class="box-container">
   <?php
      $select_bookmark = $conn->prepare(query: "SELECT * FROM `bookmark` WHERE user_id = ?");
      $select_bookmark->execute(params: [$user_id]);
      if($select_bookmark->rowCount() > 0){
```

Jika cookie tampering dilakukan, user id dapat diubah dan kita dapat melihat data orang lain, dan hal yang sama terjadi pada page admin ini membuat dalam Cyber Course mayoritas dari page yang ada dapat dilakukan Cookie Tampering

Recommendation

Untuk memperbaiki ini saat pembuatan cookie perlu penambahan 2 flag akan terlihat seperti ini

```
setcookie(name: 'user_id', value: $row['id'], expires_or_options: time() + 60*60*24*30, path: '/', doma…'', true, true);
header(header: 'location:home.php');
```

Dengan penambahan ini cookie akan aman dan Cookie tampering tidak dapat terjadi

IDOR

Serangan ini terjadi karena menggunakan GET request untuk mengambil data-data. Penemuan

```
   <a href="playlist.php?get_id=<?= $course_id; ?>" class="inline-btn">View Playlist</a>
```

Karena penggunaan href yang langsung menuju get request saat membuka halaman link dapat diakses melalui URL

get_id= dapat diganti untuk membuka data yang berbeda, dan ini terjadi pada beberapa file

```
if(isset($_GET['get_id'])){
   $get_id = $_GET['get_id'];
}else{
   $get_id = '';
   header(header: 'location:home.php');
}
```

Snippet ini menyebabkan IDOR dengan menunjukkan get_id ke dalam URL

Recommendation

Get request tidak direkomendasikan untuk digunakan, lebih baik menggunakan POST request agar URL tidak dapat diganti

```php
if (isset($_POST['playlist_id'])) {
    $get_id = $_POST['playlist_id'];
} else {
    header(header: 'Location: playlist.php');
    exit();
}
if (!is_numeric(value: $get_id)) {
    header(header: 'Location: playlist.php');
    exit();
}

$user_id = $_COOKIE['user_id'];
$stmt = $conn->prepare(query: "SELECT * FROM playlists WHERE playlist_id = ? AND user_id = ?");
$stmt->execute(params: [$get_id, $user_id]);
$playlist = $stmt->fetch();
```

Dengan codingan ini kemungkinan untuk terjadinya IDOR berkurang dengan menggunakan POST request dan verifikasi user

XSS

Terjadi karena karakter-karakter yang berada di database diambil secara value ini berarti karakter spesial seperti “, <,> yang sudah di encode di dalam database menjadi &lt, &gt akan dikembalikan menjadi karakter aslinya. Penemuan

```php
    <h3><?= $fetch_playlist['title']; ?></h3>
    <p><?= $fetch_playlist['description']; ?></p>
```

dan ini terjadi pada banyak page yang membuat aplikasi sangat vulnerable

Recommendation

Dengan penambahan htmlspecialchars di awal

```php
<h3><?= htmlspecialchars(string: $fetch_playlist['title']) ?></h3>
<p><?= htmlspecialchars(string: $fetch_playlist['description']) ?></p>
```

Data yang sudah di decode akan tetap di decode saat tunjukan di dalam html

CSRF Token

Vulnerability ini terjadi karena tidak adanya CSRF token pada form tag

```php
<form action="" method="post" class="flex-btn">
   <input type="hidden" name="comment_id" value="<?= $fetch_comment['id']; ?>">
   <button type="submit" name="edit_comment" class="inline-option-btn">Edit Comment</button>
   <button type="submit" name="delete_comment" class="inline-delete-btn" onclick="return confirm('Delete this com
</form>
```

Akibat dari ini adalah sebuah attacker dapat mengirim form dengan data lain atas nama kita

Recommendation

Untuk memperbaiki vulnerability ini kita perlu menambahkan sebuah CSRF token saat pengiriman form

```php
$csrf_token = bin2hex(string: random_bytes(length: 32));
$csrf_token = $_COOKIE['csrf_token'];


if(isset($_POST['delete_comment'])){

    if ($_POST['csrf_token'] !== $csrf_token) {
        $message[] = 'Invalid CSRF token!';
```

untuk pembuatan dan validasi CSRF token

```php
<form action="" method="post" class="flex-btn">
   <input type="hidden" name="csrf_token" value="<?= $_COOKIE['csrf_token']; ?>">
   <input type="hidden" name="comment_id" value="<?= $fetch_comment['id']; ?>">
   <button type="submit" name="edit_comment" class="inline-option-btn">Edit Comment</button>
   <button type="submit" name="delete_comment" class="inline-delete-btn" onclick="return confirm('Delete this comment?');
</form>
```

dan untuk CSRF token yang akan di send, dengan mengimplementasikan kedua snippet ini kemungkinan untuk terjadinya CSRF berkurang

SQL Injection

Sebagian besar dari codingan sudah aman terhadap SQL Injection tetapi terdapat beberapa tempat dimana SQL Injection dapat terjadi. Penemuan

```php
$select_courses = $conn->prepare(query: "SELECT * FROM `playlist` WHERE title LIKE '%{$search_course}%' AND statu
```

dalam snippet ini query menggunakan user input secara langsung menyebabkan SQL Injection dapat terjadi.

Recommendation

Untuk memperbaiki ini kita perlu menggantikan query menggunakan ?

```php
$select_courses = $conn->prepare(query: "SELECT * FROM `playlist` WHERE title LIKE ? AND status = ?");
$select_courses->execute(params: ["%{$search_course}%",'active']);
```

Maka dengan ini SQL Injection tidak dapat terjadi

Low Password Security

Dalam code terdapat hashing algorithm yang lemah ya itu sha1. Penemuan

```php
$pass = sha1(string: $_POST['pass']);
```

Menggunakan hash method ini sangat vulnerable karena memungkinkan dapat dibobol menggunakan HashClash

Recommendation

Untuk memperbaiki vulnerability ini kita perlu mengganti algoritma hashing dengan algoritma yang lebih baru

```php
$hashed_password = password_hash(password: $_POST('pass'), algo: PASSWORD_BCRYPT);
```

dan juga menggunakan php function password_hash. Dengan ini password yang akan dimasukan ke dalam database akan lebih aman.

File Upload

Vulnerability ini ditemukan di setiap file upload dan dikarenakan tidak ada pengecekan mime type. Penemuan

```php
$image = $_FILES['image']['name'];
$image = filter_var(value: $image, filter: FILTER_SANITIZE_STRING);
$ext = pathinfo(path: $image, flags: PATHINFO_EXTENSION);
$rename = create_unique_id().'.'.$ext;
$image_size = $_FILES['image']['size'];
$image_tmp_name = $_FILES['image']['tmp_name'];
$image_folder = '../uploaded_files/'.$rename;
```

Dengan pengecekan ini file file yang memiliki double extension seperti payload.php.jpg masih dapat diupload dikarenakan adanya .jpg

Recommendation

Kami menyarankan penambahan pengecekan mime types

```php
$image = $_FILES['image']['name'];

$image = preg_replace(pattern: "/[^a-zA-Z0-9_\-\.]/", replacement: "", subject: $image);

if (substr_count(haystack: $image, needle: '.') > 1) {
    die("Invalid file format. Double extensions are not allowed.");
}

$ext = strtolower(string: pathinfo(path: $image, flags: PATHINFO_EXTENSION));

$allowed_extensions = ['jpg', 'jpeg', 'png', 'gif'];

if (!in_array(needle: $ext, haystack: $allowed_extensions)) {
    die("Invalid file type. Only JPG, JPEG, PNG, and GIF files are allowed.");
}

$rename = create_unique_id().'.'.$ext;

$image_size = $_FILES['image']['size'];
if ($image_size > 5000000) {
    die("File is too large.");
}

$image_tmp_name = $_FILES['image']['tmp_name'];
$image_folder = 'uploaded_files/'.$rename;
```

Dengan penambahan codingan ini double extension attack dapat dicegah

Additional Notes

Dalam codingan terdapat beberapa kesalahan yang ada, diantaranya adalah:

- Penggunaannya sebuah filter yang sudah deprecated atau tidak didukung oleh php.

```php
$edit_id = filter_var(value: $edit_id, filter: FILTER_SANITIZE_STRING);
```

- Terdapat banyak file yang sama menyebabkan banyak file memiliki vulnerability yang sama
- Terdapat logic error dimana password di hash lalu diberi filter yang menyebabkan redundancy.

```php
$pass = sha1(string: $_POST['pass']);
$pass = filter_var(value: $pass, filter: FILTER_SANITIZE_STRING);
$cpass = sha1(string: $_POST['cpass']);
$cpass = filter_var(value: $cpass, filter: FILTER_SANITIZE_STRING);
```

Selain ini kami juga menemukan bahwa codingan CyberCourse berasal dari template codingan, yang ada di youtube

https://youtube.com/playlist?list=PLSJxovi1IyDGkHNqlrPSU2kXu1aophIkG&si=JFyh84OC_ZK8BdR_

Conclusion

Dalam konteks CyberCourse kami menemukan bahwa aplikasi sangat tidak secure, dengan adanya vulnerability yang dapat di eksploit di setiap file.