Task 3 : Source Code Review

```
<?php

// Poorly secured code

$user_input = $_POST['username'];

$password_input = $_POST['password'];


// Query to authenticate user

$query = "SELECT * FROM users WHERE username='$user_input' AND password='$password_input'";

$result = mysqli_query($connection, $query);


if(mysqli_num_rows($result) > 0) {

   echo "Login successful!";

} else {

   echo "Invalid username or password!";

}

?>
```

Now, let's review and recommend secure coding practices:

1. **SQL Injection Vulnerability**: This code is highly susceptible to SQL injection attacks since it directly concatenates user input into the SQL query. To mitigate this, use prepared statements with parameterized queries.
2. **Password Storage**: Storing passwords in plain text is a significant security risk. Always hash passwords before storing them in the database using secure hashing algorithms like bcrypt or Argon2.
3. **Input Sanitization**: User inputs should be sanitized to prevent malicious input. Use functions like **mysqli_real_escape_string()** or better yet, utilize parameterized queries to prevent SQL injection.
4. **Error Handling**: Error messages should not reveal sensitive information about the system. Instead of directly outputting SQL errors, handle them gracefully and log them securely.
5. **Authentication**: Implement secure authentication mechanisms like multi-factor authentication (MFA) or token-based authentication to enhance security.

Here's the same code with these secure practices implemented:

```php
<?php
// Secure code
$user_input = mysqli_real_escape_string($connection, $_POST['username']);
$password_input = mysqli_real_escape_string($connection, $_POST['password']);

// Hashing the password
$password_hash = password_hash($password_input, PASSWORD_DEFAULT);

// Prepared statement to authenticate user
$query = "SELECT * FROM users WHERE username=?";
$stmt = mysqli_prepare($connection, $query);
mysqli_stmt_bind_param($stmt, "s", $user_input);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
if(mysqli_num_rows($result) > 0) {
   $user = mysqli_fetch_assoc($result);
   if(password_verify($password_input, $user['password'])) {
      echo "Login successful!";
   } else {
      echo "Invalid username or password!";
   }
} else {
   echo "Invalid username or password!";
}
// Close statement and connection
mysqli_stmt_close($stmt);
mysqli_close($connection);
?>
```

In this updated code:

- User inputs are sanitized using **mysqli_real_escape_string()**.
- Passwords are hashed using **password_hash()** before storing them in the database.
- Prepared statements with parameterized queries are used to prevent SQL injection.
- Password verification is done using **password_verify()** function.
- Proper error handling is not shown here, but it's crucial to handle errors securely, such as logging them instead of displaying them to users.