E:\Thesis\web app PT\Static-tools>snyk auth

­­

Now redirecting you to our auth page, go ahead and log in,

and once the auth is complete, return to this prompt and you'll

be ready to start using snyk.

If you can't wait use this url:

https://app.snyk.io/login?token=3cf6ceca-b342-4e44-af87-96dc3f5459e4&utm\_medium=cli&utm\_source=cli&utm\_campaign=CLI\_V1\_PLUGIN&utm\_campaign\_content=1.1205.0&os=windows\_nt&docker=false

Your account has been authenticated. Snyk is now ready to be used.

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\1.HOTEL\_managment\HOTEL

Testing C:\xampp\htdocs\1.HOTEL\_managment\HOTEL ...

✗ [Low] Use of Hardcoded Credentials

Path: loginsession.php, line 4

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: userdetailinsert.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: connection.php, line 8

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [High] SQL Injection

Path: loginsession.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: editres.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: tariadd1.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: tariadd1.php, line 21

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: userdetailinsert.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: deleres.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: deleres.php, line 15

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: addres.php, line 22

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: addres.php, line 41

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: tariedit.php, line 4

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: edres.php, line 97

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: edres.php, line 259

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: feedback.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: taried.php, line 15

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: addres.php, line 40

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: tariedit.php, line 11

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: edres.php, line 32

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\1.HOTEL\_managment\HOTEL

Summary:

20 Code issues found

17 [High] 3 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\2.online-pizza-ordering-system\php-opos

Testing C:\xampp\htdocs\2.online-pizza-ordering-system\php-opos ...

✗ [Low] Use of Hardcoded Credentials

Path: admin/db\_connect.php, line 3

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 521

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 565

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 562

Info: MD5 hash (used in hexMD5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 587

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 589

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/assets/font-awesome/js/conflict-detection.js, line 592

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Open Redirect

Path: admin/admin\_class.php, line 188

Info: Unsanitized input from an HTTP header flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] SQL Injection

Path: admin/admin\_class.php, line 246

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/admin\_class.php, line 248

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/vendor/venobox/venobox.js, line 664

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/vendor/php-email-form/validate.js, line 142

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: admin/header.php, line 11

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: header.php, line 11

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [High] Hardcoded Secret

Path: admin/header.php, line 11

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: header.php, line 11

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: admin/header.php, line 15

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] Hardcoded Secret

Path: header.php, line 15

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] SQL Injection

Path: admin/admin\_class.php, line 215

Info: Unsanitized input from an HTTP header flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cart\_list.php, line 35

Info: Unsanitized input from an HTTP header flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: category.php, line 6

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: category.php, line 43

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_prod.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/manage\_user.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/view\_order.php, line 15

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/admin\_class.php, line 119

Info: Unsanitized input from an uploaded file flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: view\_prod.php, line 57

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/view\_order.php, line 51

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: login.php, line 43

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: signup.php, line 54

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/navbar.php, line 19

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] File Inclusion

Path: admin/index.php, line 36

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 81

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] Path Traversal

Path: admin/admin\_class.php, line 111

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: admin/admin\_class.php, line 165

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\2.online-pizza-ordering-system\php-opos

Summary:

35 Code issues found

21 [High] 7 [Medium] 7 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\3.online-computer-and-laptop-store\php-ocls

Testing C:\xampp\htdocs\3.online-computer-and-laptop-store\php-ocls ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: view\_product.php, line 300

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: products.php, line 56

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: products.php, line 56

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: products.php, line 94

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: home.php, line 97

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: my\_account.php, line 37

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: view\_categories.php, line 51

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Master.php, line 372

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Master.php, line 575

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Master.php, line 576

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 137

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Path Traversal

Path: view\_product.php, line 287

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Path Traversal

Path: cart.php, line 22

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] SQL Injection

Path: view\_product.php, line 292

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: classes/Master.php, line 525

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/inventory/index.php, line 36

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] File Inclusion

Path: index.php, line 20

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 22

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 22

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 24

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] Path Traversal

Path: products.php, line 84

Info: Unsanitized input from an HTTP parameter flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [High] Path Traversal

Path: home.php, line 86

Info: Unsanitized input from an HTTP parameter flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [High] Path Traversal

Path: classes/Users.php, line 30

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/Users.php, line 98

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/Users.php, line 141

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 50

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 60

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] SQL Injection

Path: view\_product.php, line 2

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 5

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 11

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 19

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 27

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 53

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products.php, line 79

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: home.php, line 81

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: home.php, line 94

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_categories.php, line 5

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_categories.php, line 9

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_categories.php, line 15

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_categories.php, line 21

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/sales/index.php, line 69

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/sales/index.php, line 71

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_category.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_brand.php, line 5

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/manage\_inventory.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/manage\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_sub\_category.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/view\_order.php, line 14

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/product/manage\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: products.php, line 51

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products.php, line 56

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products.php, line 65

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: home.php, line 141

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/sales/index.php, line 42

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/sales/index.php, line 42

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 104

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 105

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/topBarNav.php, line 11

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/orders/update\_status.php, line 3

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 121

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 122

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\3.online-computer-and-laptop-store\php-ocls

Summary:

64 Code issues found

46 [High] 6 [Medium] 12 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\4.quality-beauty-parlour-management-system\bea

Testing C:\xampp\htdocs\4.quality-beauty-parlour-management-system\bea ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 11

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 12

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/reset-password.php, line 11

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/index.php, line 9

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/dbconnection.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [High] Cross-site Scripting (XSS)

Path: admin/search-invoices.php, line 81

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/search-appointment.php, line 82

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/bwdates-reports-details.php, line 68

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/bwdates-reports-details.php, line 68

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: admin/search-invoices.php, line 92

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/search-appointment.php, line 86

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/bwdates-reports-details.php, line 80

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/index.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/sales-reports-detail.php, line 88

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/sales-reports-detail.php, line 128

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/contact-us.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/admin-profile.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-appointment.php, line 68

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: index.php, line 17

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: index.php, line 19

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/about-us.php, line 15

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/forgot-password.php, line 11

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/add-customer.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/view-appointment.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/view-appointment.php, line 88

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/add-services.php, line 16

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/add-customer-services.php, line 16

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-services.php, line 16

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-services.php, line 85

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-customer-detailed.php, line 19

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-customer-detailed.php, line 88

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\4.quality-beauty-parlour-management-system\bea

Summary:

32 Code issues found

26 [High] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\5.book-management-system\BMS

Testing C:\xampp\htdocs\5.book-management-system\BMS ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/connection.php, line 3

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [High] Path Traversal

Path: admin/process\_book\_edit.php, line 59

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: admin/process\_book\_add.php, line 58

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Cross-site Scripting (XSS)

Path: book\_list.php, line 7

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: search.php, line 7

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: book\_list.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: search.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_users\_del.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/book\_edit.php, line 8

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: book\_detail.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_category\_del.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_book\_del.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_contact\_del.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/category\_edit.php, line 11

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: addtocart.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_book\_edit.php, line 64

Info: Unsanitized input from an uploaded file flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/process\_book\_add.php, line 63

Info: Unsanitized input from an uploaded file flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\5.book-management-system\BMS

Summary:

17 Code issues found

16 [High] 1 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\6.online-eyewear-shop-application\oews

Testing C:\xampp\htdocs\6.online-eyewear-shop-application\oews ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 17

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 132

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 22

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 47

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Path Traversal

Path: classes/Users.php, line 222

Info: Unsanitized input from a database flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: classes/SystemSettings.php, line 239

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [Medium] SQL Injection

Path: classes/Master.php, line 332

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 47

Info: Unsanitized input from an HTTP parameter flows into file\_put\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 239

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 243

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] File Inclusion

Path: admin/index.php, line 26

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 28

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 20

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 22

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] Cross-site Scripting (XSS)

Path: admin/reports/index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 169

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 170

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inventory/manage\_stock.php, line 15

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 91

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 92

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: report/list.php, line 4

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: admin/reports/index.php, line 64

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/manage\_stock.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: report/list.php, line 36

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/products/manage\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/customers/manage\_customer.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/categories/manage\_category.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inquiries/view\_inquiry.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products/index.php, line 22

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products/index.php, line 49

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: orders/view\_order.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/update\_status.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/categories/view\_category.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: report/view\_report.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/assign\_team.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/view\_inventory.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/view\_order.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/manage\_request.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/products/view\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products/view\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\6.online-eyewear-shop-application\oews

Summary:

43 Code issues found

34 [High] 4 [Medium] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\7.computer-service-management-system\LabManagement

Testing C:\xampp\htdocs\7.computer-service-management-system\LabManagement ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: class/edit/edit.php, line 641

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: class/login/login.php, line 61

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: class/add/add.php, line 405

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: class/add/add.php, line 417

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: class/add/add.php, line 468

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: assets/amcharts/plugins/export/libs/xlsx/xlsx.js, line 1796

Info: md5 hash (used in createHash) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/edit.js, line 918

Info: Unsanitized input from data from a remote resource flows into write, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/add.js, line 237

Info: Unsanitized input from data from a remote resource flows into write, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 178

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 179

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 180

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 181

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 182

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 183

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 184

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 185

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 186

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 187

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 188

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 189

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/custom.js, line 190

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/add.js, line 208

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/mycustom/js/add.js, line 209

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/items\_info.php, line 222

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/member\_profile.php, line 120

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/member\_profile.php, line 202

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/room\_info.php, line 120

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/room\_info.php, line 215

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/room\_info.php, line 216

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: views/footer.php, line 60

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: member/footer.php, line 39

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 309

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 329

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 351

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 544

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 576

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1000

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1000

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1046

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1081

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1129

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1136

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1137

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1140

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1145

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1150

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1268

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1306

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1351

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1352

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1353

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1682

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1772

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1772

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1802

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1934

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1934

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1937

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1961

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1994

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 1997

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2010

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2070

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2082

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2133

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2153

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2298

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2349

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2492

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2499

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2630

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2640

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2703

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2704

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2713

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2795

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2808

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2811

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2869

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2877

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 2928

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3110

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3112

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3114

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3115

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3116

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3117

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3118

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3124

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3130

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3168

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3218

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Cross-site Scripting (XSS)

Path: assets/amcharts/plugins/export/export.js, line 3229

Info: Unsanitized input from the document location flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\7.computer-service-management-system\LabManagement

Summary:

93 Code issues found

70 [High] 17 [Medium] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\8.Student-Management-System\studentms

Testing C:\xampp\htdocs\8.Student-Management-System\studentms ...

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: user/login.php, line 26

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: user/login.php, line 28

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: user/login.php, line 31

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: user/login.php, line 33

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 24

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 26

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 29

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 31

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: user/login.php, line 9

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/login.php, line 9

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 12

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 13

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: user/change-password.php, line 12

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: user/change-password.php, line 13

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/add-students.php, line 22

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/add-students.php, line 40

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: user/forgot-password.php, line 10

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/forgot-password.php, line 10

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: user/login.php, line 26

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: user/login.php, line 28

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: user/login.php, line 31

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: user/login.php, line 33

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 24

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 26

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 29

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 31

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Medium] Privacy Leak

Path: admin/edit-student-detail.php, line 176

Info: Sensitive data from a password flows into the echo statement, where it is leaked.

✗ [High] SQL Injection

Path: admin/between-date-reprtsdetails.php, line 108

Info: Unsanitized input from an HTTP parameter flows into prepare, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-class-detail.php, line 75

Info: Unsanitized input from an HTTP parameter flows into prepare, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/search.php, line 118

Info: Unsanitized input from an HTTP parameter flows into prepare, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: admin/between-date-reprtsdetails.php, line 73

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/between-date-reprtsdetails.php, line 73

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/between-date-reprtsdetails.php, line 140

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/search.php, line 84

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/search.php, line 157

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/manage-class.php, line 130

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/manage-notice.php, line 132

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/manage-students.php, line 134

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/manage-public-notice.php, line 128

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: user/login.php, line 77

Info: Unsanitized input from cookies flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: user/login.php, line 81

Info: Unsanitized input from cookies flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/login.php, line 75

Info: Unsanitized input from cookies flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/login.php, line 79

Info: Unsanitized input from cookies flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Path Traversal

Path: admin/add-students.php, line 41

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\8.Student-Management-System\studentms

Summary:

44 Code issues found

17 [High] 1 [Medium] 26 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\9.online-course-registration\onlinecourse

Testing C:\xampp\htdocs\9.online-course-registration\onlinecourse ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: index.php, line 8

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/manage-students.php, line 24

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/student-registration.php, line 14

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: news-details.php, line 8

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/index.php, line 8

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: change-password.php, line 18

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: change-password.php, line 19

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 16

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin/change-password.php, line 20

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/manage-students.php, line 23

Info: Do not hardcode passwords in code. Found a hardcoded password used in an SQL query.

✗ [High] SQL Injection

Path: admin/course.php, line 17

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/course.php, line 31

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: index.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-student-profile.php, line 20

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/edit-course.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/level.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/level.php, line 25

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/manage-students.php, line 15

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/manage-students.php, line 25

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/news.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/news.php, line 29

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: enroll.php, line 20

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/semester.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/semester.php, line 27

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/check\_availability.php, line 6

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/student-registration.php, line 16

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/department.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/department.php, line 27

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/session.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/session.php, line 28

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: check\_availability.php, line 6

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: check\_availability.php, line 17

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: check\_availability.php, line 19

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: my-profile.php, line 17

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: news-details.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: news-details.php, line 84

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: pincode-verification.php, line 15

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/index.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Path Traversal

Path: admin/edit-student-profile.php, line 19

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: my-profile.php, line 16

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\9.online-course-registration\onlinecourse

Summary:

40 Code issues found

30 [High] 1 [Medium] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test C:\xampp\htdocs\10.Student-Result-Management-System\srms

Testing C:\xampp\htdocs\10.Student-Result-Management-System\srms ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: admin-login.php, line 11

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: change-password.php, line 12

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: change-password.php, line 13

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: js/amcharts/plugins/export/libs/xlsx/xlsx.js, line 1796

Info: md5 hash (used in createHash) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 208

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 213

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 215

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 239

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 241

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 258

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 260

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 281

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 283

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 301

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 303

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 321

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 323

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 702

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 815

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 857

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/lobipanel/lobipanel.js, line 937

Info: Unsanitized input from browser storage flows into append, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: js/jquery-steps/jquery.steps.js, line 723

Info: Unsanitized input from data from a remote resource flows into html, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Code Injection

Path: js/lobipanel/lobipanel.js, line 570

Info: Unsanitized input from browser storage flows into eval, where it is executed as JavaScript code. This may result in a Code Injection vulnerability.

✗ [High] SQL Injection

Path: notice-details.php, line 44

Info: Unsanitized input from an HTTP parameter flows into prepare, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\10.Student-Result-Management-System\srms

Summary:

24 Code issues found

1 [High] 19 [Medium] 4 [Low]