E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\47.Online-Shopping-Portal-project\shopping"

Testing C:\xampp\htdocs\47.Online-Shopping-Portal-project\shopping ...

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

Path: 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/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.

✗ [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: 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: login.php, line 25

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 28

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 32

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

✗ [Low] Path Traversal

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 10

Info: Unsanitized input from the request URL flows into fs.readFile, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

✗ [Low] Python 2 source code

Path: admin/assets/plugins/dynatree/doc/dynatree\_server.py, line 85

Info: This source file appears to be in Python 2. The Python 2 interpreter has been unsupported without security updates since January 2020. Consider porting this code to Python 3.

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 806

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 842

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1099

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1145

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1162

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1254

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 806

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 842

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1099

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1145

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1162

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1254

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/dynatree/tests/qunit.js, line 545

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-mockjax/test/jquery.js, line 23

Info: Unsanitized input from the document location 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/jquery.js, line 23

Info: Unsanitized input from the document location 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).

✗ [Low] Allocation of Resources Without Limits or Throttling

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 6

Info: This endpoint handler performs a file system operation and does not use a rate-limiting mechanism. It may enable the attackers to perform Denial-of-service attacks. Consider using a rate-limiting middleware such as express-limit.

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/js\_data.php, line 87

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dymanic\_table.php, line 30

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data\_two\_headers.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/6776.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/deferred\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data\_th.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/2512.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/js\_data\_mixed\_types.php, line 87

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/-complex\_header.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/two\_tables.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/complex\_header\_2.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/html\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/empty\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data.php, line 25

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).

✗ [Low] Cleartext Transmission of Sensitive Information

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 6

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Medium] Cleartext Transmission of Sensitive Information

Path: admin/assets/plugins/jQuery-File-Upload/server/node/server.js, line 290

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Medium] Debug Mode Enabled

Path: admin/assets/plugins/jQuery-File-Upload/server/gae-python/main.py, line 159

Info: Running the application in debug mode (debug flag is set to True in webapp2.WSGIApplication) is a security risk if the application is accessible by untrusted parties.

✗ [Medium] Permissive Cross-domain Policy

Path: admin/assets/plugins/ckeditor/plugins/wsc/dialogs/tmp.html, line 63

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [Medium] Open Redirect

Path: admin/assets/plugins/jQuery-File-Upload/server/gae-go/app/main.go, line 229

Info: Unsanitized input from a web form flows into net.http.Redirect, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] Open Redirect

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 755

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

✗ [Medium] Open Redirect

Path: forgot-password.php, line 19

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] Open Redirect

Path: forgot-password.php, line 28

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] Open Redirect

Path: admin/index.php, line 18

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] Open Redirect

Path: admin/index.php, line 27

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] Open Redirect

Path: login.php, line 39

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] Open Redirect

Path: login.php, line 51

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] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 806

Info: Unsanitized input from an exception 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 842

Info: Unsanitized input from an exception 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1099

Info: Unsanitized input from an exception 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1145

Info: Unsanitized input from an exception 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1162

Info: Unsanitized input from an exception 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1254

Info: Unsanitized input from an exception 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] Regular Expression Denial of Service (ReDoS)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 433

Info: Unsanitized user input from an HTTP header flows into preg\_match, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 719

Info: Unsanitized user input from an HTTP header flows into preg\_match, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 852

Info: Unsanitized user input from an HTTP header flows into preg\_match, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Path Traversal

Path: admin/insert-product.php, line 34

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/insert-product.php, line 35

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/insert-product.php, line 36

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/update-image1.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: admin/update-image2.php, line 20

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/update-image3.php, line 18

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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 295

Info: Unsanitized input from an HTTP header 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: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 308

Info: Unsanitized input from an HTTP header flows into mkdir, 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 690

Info: Unsanitized input from an HTTP header flows into mkdir, 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 330

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 698

Info: Unsanitized input from an HTTP header 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 708

Info: Unsanitized input from an HTTP header 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 700

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 738

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 704

Info: Unsanitized input from an HTTP header 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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 725

Info: Unsanitized input from an HTTP header 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.

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 975

Info: Unsanitized input from an HTTP header 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.

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 981

Info: Unsanitized input from an HTTP header 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.

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 747

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

✗ [High] SQL Injection

Path: 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: forgot-password.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/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: login.php, line 12

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: login.php, line 26

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: login.php, line 48

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-account.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/insert-product.php, line 37

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/subcategory.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/subcategory.php, line 21

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: payment-method.php, line 12

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-users.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: 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/updateorder.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/updateorder.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-products.php, line 23

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/category.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/category.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: pending-orders.php, line 12

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: bill-ship-addresses.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: bill-ship-addresses.php, line 32

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: search-result.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: search-result.php, line 195

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: order-details.php, line 106

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: order-details.php, line 110

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: product-details.php, line 44

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: category.php, line 32

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-subcategory.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: sub-category.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: admin/edit-category.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/manage-products.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: my-cart.php, line 51

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-cart.php, line 64

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-cart.php, line 79

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/update-image1.php, line 20

Info: Unsanitized input from an uploaded file 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/update-image2.php, line 21

Info: Unsanitized input from an uploaded file 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/update-image3.php, line 19

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/lib/jquery.js, line 25

Info: Unsanitized input from the document location 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).

✗ [High] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 740

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/assets/plugins/DataTables/examples/server\_side/scripts/objects\_jsonp.php, line 192

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/assets/plugins/DataTables/examples/examples\_support/editable\_ajax.php, line 2

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/assets/plugins/DataTables/examples/server\_side/scripts/jsonp.php, line 190

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/assets/plugins/jQuery-Smart-Wizard/more\_examples/services/service.php, line 62

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/assets/plugins/DataTables/examples/server\_side/scripts/id\_jsonp.php, line 195

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/assets/plugins/jquery-validation/demo/captcha/index.php, line 55

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] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 700

Info: Unsanitized input from an HTTP header flows into fopen, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 738

Info: Unsanitized input from an HTTP header flows into fopen, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 725

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 975

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 981

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 747

Info: Unsanitized input from an HTTP header flows into readfile, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\47.Online-Shopping-Portal-project\shopping

Summary:

131 Code issues found

73 [High] 17 [Medium] 41 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\49.Online-Fire-Reporting-System\ofrs"

Testing C:\xampp\htdocs\49.Online-Fire-Reporting-System\ofrs ...

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

Path: admin/vendor/fontawesome-free/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/vendor/fontawesome-free/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/vendor/fontawesome-free/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/vendor/fontawesome-free/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/vendor/fontawesome-free/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/vendor/fontawesome-free/js/conflict-detection.js, line 592

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 3

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/config.php, line 3

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

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

Path: admin/index.php, line 6

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-site.php, line 36

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: admin/change-password.php, line 14

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

✗ [High] SQL Injection

Path: admin/edit-team.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/edit-team.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/add-team.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/team-ontheway-requests.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: details.php, line 45

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: details.php, line 95

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: details.php, line 143

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/new-requests.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/all-requests.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/index.php, line 8

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-report-result.php, line 100

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: search-report-result.php, line 62

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/workin-progress-requests.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/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: admin/assigned-requests.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/request-details.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/request-details.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: admin/request-details.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: admin/request-details.php, line 110

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/request-details.php, line 172

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/request-details.php, line 220

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/request-details.php, line 335

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-report-result.php, line 101

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/completed-requests.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: reporting.php, line 7

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-teams.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/profile.php, line 24

Info: Unsanitized input from an HTTP header 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/profile.php, line 30

Info: Unsanitized input from an HTTP header 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-site.php, line 30

Info: Unsanitized input from an HTTP header 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-site.php, line 43

Info: Unsanitized input from an HTTP header 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/change-password.php, line 23

Info: Unsanitized input from an HTTP header 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/change-password.php, line 28

Info: Unsanitized input from an HTTP header 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-site.php, line 40

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/edit-team.php, line 80

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-report-result.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: search-report-result.php, line 34

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-report-result.php, line 66

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-report-result.php, line 66

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] Path Traversal

Path: admin/manage-site.php, line 38

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/manage-site.php, line 41

Info: Unsanitized input from an HTTP parameter 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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\49.Online-Fire-Reporting-System\ofrs

Summary:

52 Code issues found

40 [High] 12 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\50.Online-Nurse-Hiring-System\onhs"

Testing C:\xampp\htdocs\50.Online-Nurse-Hiring-System\onhs ...

✗ [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: 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: admin/change-password.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: admin/update-nurse-pic.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: admin/password-recovery.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/add-nurse.php, line 37

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/config.php, line 5

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 5

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/search-report-details.php, line 52

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: team.php, line 189

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] Path Traversal

Path: admin/update-nurse-pic.php, line 32

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/add-nurse.php, line 39

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/update-nurse-pic.php, line 40

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] Path Traversal

Path: admin/manage-nurse.php, line 18

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] SQL Injection

Path: admin/search-report-details.php, line 96

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] SQL Injection

Path: admin/password-recovery.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/password-recovery.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/add-nurse.php, line 45

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: book-nurse.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: book-nurse.php, line 21

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-request.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/view-request.php, line 94

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-report-details.php, line 102

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/profile.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-nurse.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/update-nurse-pic.php, line 38

Info: Unsanitized input from an uploaded file 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\50.Online-Nurse-Hiring-System\onhs

Summary:

27 Code issues found

19 [High] 8 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\51.Online-Security-Guard-Hiring-System\osghs"

Testing C:\xampp\htdocs\51.Online-Security-Guard-Hiring-System\osghs ...

✗ [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: admin/changeimage.php, line 21

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/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/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: admin/add-security-guard.php, line 25

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

✗ [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.

✗ [High] SQL Injection

Path: admin/changeimage.php, line 94

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-guard-detail.php, line 91

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/bwdates-reports-details.php, line 87

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 101

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: search-request.php, line 81

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] Path Traversal

Path: admin/changeimage.php, line 22

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/add-security-guard.php, line 26

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: admin/login.php, line 74

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 82

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/bwdates-reports-details.php, line 72

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 72

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 83

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-request.php, line 64

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\51.Online-Security-Guard-Hiring-System\osghs

Summary:

27 Code issues found

13 [High] 14 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\52.Pre-School-Enrollment-System\preschool"

Testing C:\xampp\htdocs\52.Pre-School-Enrollment-System\preschool ...

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

Path: admin/add-teacher.php, line 31

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: admin/password-recovery.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/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: admin/add-subadmin.php, line 15

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-subadmin-pwd.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/update-class-pic.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: admin/add-class.php, line 32

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/update-teacher-pic.php, line 30

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 5

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/config.php, line 5

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

✗ [High] SQL Injection

Path: admin/add-teacher.php, line 39

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] SQL Injection

Path: admin/password-recovery.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/password-recovery.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/add-subadmin.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/add-class.php, line 40

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 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/edit-class.php, line 21

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/aboutus.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/visitor-details.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-subadmin.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: visit.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/profile.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-teacher.php, line 21

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: enrollment.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/contact-us.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/enrollment-details.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/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/update-class-pic.php, line 38

Info: Unsanitized input from an uploaded file 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/update-teacher-pic.php, line 38

Info: Unsanitized input from an uploaded file 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/update-class-pic.php, line 40

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] Path Traversal

Path: admin/update-teacher-pic.php, line 40

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] Path Traversal

Path: admin/manage-teachers.php, line 18

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] Path Traversal

Path: admin/manage-classes.php, line 18

Info: Unsanitized input from an HTTP parameter 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.

✗ [High] Path Traversal

Path: admin/add-teacher.php, line 33

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/update-class-pic.php, line 32

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/add-class.php, line 34

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/update-teacher-pic.php, line 32

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\52.Pre-School-Enrollment-System\preschool

Summary:

40 Code issues found

28 [High] 12 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\53.Rail-Pass-Management-System\rpms"

Testing C:\xampp\htdocs\53.Rail-Pass-Management-System\rpms ...

✗ [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 Password Hash With Insufficient Computational Effort

Path: admin/changeimage.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-pass.php, line 36

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: 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: admin/index.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/index.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/index.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/index.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 in HTTPS Session Without 'Secure' Attribute

Path: admin/index.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/index.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/index.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/index.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.

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 76

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/index.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/pass-bwdates-reports-details.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: admin/pass-bwdates-reports-details.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: download-pass.php, line 62

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-pass.php, line 74

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/pass-bwdates-reports-details.php, line 82

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: download-pass.php, line 78

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-pass.php, line 90

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/view-pass-detail.php, line 66

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: view-pass-detail.php, line 63

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-pass-detail.php, line 102

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/changeimage.php, line 83

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/view-enquiry.php, line 66

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-category-detail.php, line 90

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] Path Traversal

Path: admin/changeimage.php, line 23

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/add-pass.php, line 37

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\53.Rail-Pass-Management-System\rpms

Summary:

31 Code issues found

17 [High] 14 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\54.Restaurant-Table-Booking-System\rtbs"

Testing C:\xampp\htdocs\54.Restaurant-Table-Booking-System\rtbs ...

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/config.php, line 5

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

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

Path: admin/bw-dates-report.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/bw-dates-report.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/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: 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: admin/reset-subadmin-pwd.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-subadmin.php, line 15

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/password-recovery.php, line 8

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

✗ [High] Cross-site Scripting (XSS)

Path: search-result.php, line 39

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/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/add-subadmin.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/password-recovery.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/password-recovery.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: search-result.php, line 79

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-status.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/bwdates-report-details.php, line 96

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/add-table.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/booking-details.php, line 21

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/booking-details.php, line 26

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-subadmin.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: index.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/profile.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\54.Restaurant-Table-Booking-System\rtbs

Summary:

24 Code issues found

15 [High] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\56.Tourism-Management-System\tms"

Testing C:\xampp\htdocs\56.Tourism-Management-System\tms ...

✗ [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: 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: admin/change-password.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: includes/signin.php, line 6

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 7

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: forgot-password.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: includes/signup.php, line 8

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

✗ [High] Cross-site Scripting (XSS)

Path: page.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] Path Traversal

Path: admin/create-package.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: admin/change-image.php, line 15

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\56.Tourism-Management-System\tms

Summary:

11 Code issues found

3 [High] 8 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\57.Zoo-Management System\zms"

Testing C:\xampp\htdocs\57.Zoo-Management System\zms ...

✗ [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 Password Hash With Insufficient Computational Effort

Path: admin/add-animals.php, line 25

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/changeimage.php, line 21

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 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 Hardcoded Credentials

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

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

✗ [High] SQL Injection

Path: admin/edit-animal-details.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-animal-details.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/view-normal-ticket.php, line 66

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-foreigner-ticket.php, line 66

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-foreigners-ticket.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/foreigner-search.php, line 90

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-normal-ticket.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/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/manage-normal-ticket.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-animals.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/add-animals.php, line 35

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/normal-search.php, line 90

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/profile.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/aboutus.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/changeimage.php, line 95

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/manage-animals.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/edit-ticket.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-ticket.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/contactus.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/normal-bwdates-reports-details.php, line 83

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: animal-detail.php, line 45

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/foreigner-bwdates-reports-details.php, line 83

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-foreigners-ticket.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/changeimage.php, line 23

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/foreigner-search.php, line 77

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/normal-search.php, line 77

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/normal-bwdates-reports-details.php, line 69

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/normal-bwdates-reports-details.php, line 69

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/foreigner-bwdates-reports-details.php, line 69

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/foreigner-bwdates-reports-details.php, line 69

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: animals.php, line 74

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] Path Traversal

Path: admin/add-animals.php, line 26

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/changeimage.php, line 22

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\57.Zoo-Management System\zms

Summary:

42 Code issues found

34 [High] 8 [Low]