snyk.exe code test "C:\xampp\htdocs\58.online-education-institutes-managing-system"

Testing C:\xampp\htdocs\58.online-education-institutes-managing-system ...

✗ [Low] Use of Hardcoded Credentials

Path: contact.php, line 7

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

✗ [High] SQL Injection

Path: contact.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\58.online-education-institutes-managing-system

Summary:

2 Code issues found

1 [High] 1 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\59.simple-real-time-chatbox\chat"

Testing C:\xampp\htdocs\59.simple-real-time-chatbox\chat ...

✗ [Low] Use of Hardcoded Credentials

Path: db.php, line 4

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

✗ [Low] Use of Hardcoded Credentials

Path: db.php, line 10

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

✗ [Medium] Privacy Leak

Path: register.php, line 56

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

✗ [High] SQL Injection

Path: login.php, line 67

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

✗ [High] Cross-site Scripting (XSS)

Path: register.php, line 56

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

✗ [High] Cross-site Scripting (XSS)

Path: insert.php, line 4

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

✗ [High] Cross-site Scripting (XSS)

Path: insert.php, line 5

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\59.simple-real-time-chatbox\chat

Summary:

7 Code issues found

4 [High] 1 [Medium] 2 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\60.online-food-ordering-system"

Testing C:\xampp\htdocs\60.online-food-ordering-system ...

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

Path: dashboard.php, line 18

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 Without 'HttpOnly' Flag

Path: dashboard.php, line 18

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] Use of Hardcoded Credentials

Path: connection.php, line 3

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

✗ [Low] Use of Hardcoded Credentials

Path: connection.php, line 6

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

✗ [Low] Use of Hardcoded Credentials

Path: fetch.php, line 2

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

✗ [Low] Use of Hardcoded Credentials

Path: fetch2.php, line 2

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

✗ [Medium] Use of Hardcoded Credentials

Path: connection.php, line 4

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli\_connect.

✗ [Medium] Use of Hardcoded Credentials

Path: connection.php, line 6

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli\_connect.

✗ [Medium] Use of Hardcoded Credentials

Path: fetch.php, line 2

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli\_connect.

✗ [Medium] Use of Hardcoded Credentials

Path: fetch2.php, line 2

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli\_connect.

✗ [High] Path Traversal

Path: deletefood.php, line 11

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: deleteVendor.php, line 16

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: deleteVendor.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: deleteVendor.php, line 33

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: vendor\_delete\_food.php, line 12

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: update.php, line 84

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: deleteVendor.php, line 34

Info: Unsanitized input from an HTTP parameter flows into rmdir, 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: deleteVendor.php, line 35

Info: Unsanitized input from an HTTP parameter flows into rmdir, 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: food.php, line 49

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: food.php, line 94

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: update.php, line 83

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: vendor-new.php, line 29

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

✗ [High] SQL Injection

Path: food.php, line 45

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: food.php, line 91

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: update.php, line 79

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: vendor-new.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: deletefood.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: deletefood.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: deletefood.php, line 41

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: deleteVendor.php, line 5

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: deleteVendor.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: deleteVendor.php, line 20

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

✗ [High] SQL Injection

Path: deleteVendor.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: deleteVendor.php, line 28

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

✗ [High] SQL Injection

Path: deleteVendor.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: vendor\_delete\_food.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: vendor\_delete\_food.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: update.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: update.php, line 57

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: form/cart.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: form/cart.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: form/cart.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: form/cart.php, line 75

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: form/cart.php, line 81

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: form/cart.php, line 208

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: form/cart.php, line 215

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: form/cart.php, line 318

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: form/cart.php, line 322

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: form/order.php, line 5

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: form/order.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: form/order.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: form/cancelorder.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] Cross-site Scripting (XSS)

Path: form/cart.php, line 249

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: form/order.php, line 3

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

✗ [High] Cross-site Scripting (XSS)

Path: form/index.php, line 31

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: form/index.php, line 155

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: form/forms.php, line 24

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: update.php, line 78

Info: Unsanitized input from an uploaded file 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: deletefood.php, line 11

Info: Unsanitized input from an HTTP parameter 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: deleteVendor.php, line 16

Info: Unsanitized input from an HTTP parameter 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: deleteVendor.php, line 18

Info: Unsanitized input from an HTTP parameter 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: deleteVendor.php, line 33

Info: Unsanitized input from an HTTP parameter 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: vendor\_delete\_food.php, line 12

Info: Unsanitized input from an HTTP parameter 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: update.php, line 84

Info: Unsanitized input from an HTTP parameter flows into unlink, 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\60.online-food-ordering-system

Summary:

64 Code issues found

54 [High] 4 [Medium] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\61.online-voting-system"

Testing C:\xampp\htdocs\61.online-voting-system ...

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

✗ [Low] Use of Hardcoded Credentials

Path: db\_connect.php, line 3

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

✗ [Medium] Cross-site Scripting (XSS)

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

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

✗ [Medium] Cross-site Scripting (XSS)

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

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

✗ [Medium] Path Traversal

Path: admin\_class.php, line 170

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

✗ [Medium] Path Traversal

Path: admin\_class.php, line 189

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

✗ [High] Cross-site Scripting (XSS)

Path: navbar.php, line 20

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: manage\_catset.php, line 4

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

✗ [High] Cross-site Scripting (XSS)

Path: manage\_catset.php, line 5

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: manage\_catset.php, line 6

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: manage\_catset.php, line 7

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

✗ [High] Cross-site Scripting (XSS)

Path: manage\_catset.php, line 10

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: files.php, line 169

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

✗ [High] Cross-site Scripting (XSS)

Path: files.php, line 172

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: files.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] Cross-site Scripting (XSS)

Path: files.php, line 233

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: files.php, line 256

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: manage\_voting.php, line 106

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: manage\_voting.php, line 109

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: manage\_voting.php, line 112

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: manage\_opt.php, line 13

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\_class.php, line 51

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

✗ [High] SQL Injection

Path: files.php, line 4

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

✗ [High] SQL Injection

Path: files.php, line 7

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

✗ [High] SQL Injection

Path: files.php, line 61

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

✗ [High] SQL Injection

Path: manage\_voting.php, line 4

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

✗ [High] SQL Injection

Path: manage\_voting.php, line 8

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

✗ [High] SQL Injection

Path: manage\_voting.php, line 18

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

✗ [High] SQL Injection

Path: manage\_voting.php, line 65

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

✗ [High] SQL Injection

Path: manage\_opt.php, line 4

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

✗ [High] SQL Injection

Path: manage\_user.php, line 4

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

✗ [High] File Inclusion

Path: voting.php, line 38

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

✗ [High] File Inclusion

Path: index.php, line 35

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

✗ [High] Path Traversal

Path: admin\_class.php, line 46

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\_class.php, line 164

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\61.online-voting-system

Summary:

40 Code issues found

29 [High] 4 [Medium] 7 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\62.online-cake-shop"

Testing C:\xampp\htdocs\62.online-cake-shop ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/connection.php, line 2

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

✗ [Medium] Open Redirect

Path: includes/signup.php, line 26

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

✗ [High] Cross-site Scripting (XSS)

Path: 404.php, line 17

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: updateproduct1.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: addtransacdetail.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: updatesupplier.php, line 24

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: insertsupplier.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: deleteproduct.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: updatecustomer1.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: transactionsave.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: deletecustomer.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: inserttrans.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: confirm.php, line 5

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: confirm.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: inserttransac.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: detailtransac.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: detailtransac.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: detailtransac.php, line 70

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: detailtransac.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: updatecustomer.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: admindetail.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: insertproduct.php, line 34

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: insertcustomer.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: orderdetail.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: orderdetail.php, line 65

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: orderdetail.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: orderdetail.php, line 109

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: deletetransac.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: updateproduct.php, line 24

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: updatesupplier1.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\62.online-cake-shop

Summary:

30 Code issues found

28 [High] 1 [Medium] 1 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\63.events-management-system"

Testing C:\xampp\htdocs\63.events-management-system ...

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

Path: chklogin.php, line 26

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/viewuser.php, line 59

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/viewuser.php, line 64

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/reguser.php, line 59

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/reguser.php, line 64

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/viewevent.php, line 59

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/viewevent.php, line 64

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

✗ [Low] Use of Hardcoded Credentials

Path: conn.php, line 6

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/conn.php, line 2

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

✗ [Medium] Use of Hardcoded Credentials

Path: admin/viewuser.php, line 60

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/viewuser.php, line 64

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/reguser.php, line 60

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/reguser.php, line 64

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/viewevent.php, line 60

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/viewevent.php, line 64

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysqli.

✗ [Medium] Use of Hardcoded Credentials

Path: conn.php, line 6

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysql\_connect.

✗ [Medium] Use of Hardcoded Credentials

Path: admin/conn.php, line 2

Info: Do not hardcode passwords in code. Found a hardcoded password used in mysql\_connect.

✗ [High] SQL Injection

Path: admin/chkpostannounce.php, line 50

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

✗ [High] SQL Injection

Path: admin/chkbanuser.php, line 6

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

✗ [High] SQL Injection

Path: admin/chkactiveuser.php, line 7

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

✗ [High] SQL Injection

Path: chkforgot.php, line 47

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

✗ [High] SQL Injection

Path: eventreg.php, line 53

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

✗ [High] SQL Injection

Path: admin/chkchangepass.php, line 54

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

✗ [High] SQL Injection

Path: admin/chkchangepass.php, line 58

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

✗ [High] SQL Injection

Path: chkchangepass.php, line 30

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

✗ [High] SQL Injection

Path: chkchangepass.php, line 34

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

✗ [High] SQL Injection

Path: admin/chkdaterep.php, line 79

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

✗ [High] SQL Injection

Path: admin/chkgalupload.php, line 60

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

✗ [High] Path Traversal

Path: admin/chkgalupload.php, line 53

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: chklogin.php, line 35

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: eventreg.php, line 85

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\63.events-management-system

Summary:

31 Code issues found

14 [High] 8 [Medium] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\ajms"

Testing C:\xampp\htdocs\ajms ...

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

Path: classes/Users.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: classes/Users.php, line 37

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

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

Path: classes/Users.php, line 130

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

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

Path: classes/Users.php, line 147

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

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

Path: inc/packages.php, line 30

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

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

Path: classes/Login.php, line 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: classes/Login.php, line 49

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

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

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

✗ [Medium] Path Traversal

Path: classes/Users.php, line 119

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

✗ [Medium] Path Traversal

Path: classes/Users.php, line 230

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

✗ [Medium] SQL Injection

Path: admin/journals/index.php, line 67

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

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 44

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

✗ [High] SQL Injection

Path: admin/reports/working\_trial\_balance.php, line 63

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

✗ [High] SQL Injection

Path: admin/reports/working\_trial\_balance.php, line 68

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

✗ [High] SQL Injection

Path: admin/reports/working\_trial\_balance.php, line 70

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

✗ [High] SQL Injection

Path: admin/groups/view\_group.php, line 4

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

✗ [High] SQL Injection

Path: admin/reports/trial\_balance.php, line 74

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

✗ [High] SQL Injection

Path: admin/reports/trial\_balance.php, line 87

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

✗ [High] SQL Injection

Path: admin/groups/manage\_group.php, line 4

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

✗ [High] SQL Injection

Path: admin/accounts/view\_account.php, line 4

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

✗ [High] SQL Injection

Path: admin/journals/view\_details.php, line 4

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

✗ [High] SQL Injection

Path: admin/journals/manage\_journal.php, line 6

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/accounts/manage\_account.php, line 4

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/reports/working\_trial\_balance.php, line 29

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/reports/working\_trial\_balance.php, line 33

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/reports/trial\_balance.php, line 24

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/reports/trial\_balance.php, line 28

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/journals/update\_status.php, line 3

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 118

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

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 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] File Inclusion

Path: admin/index.php, line 26

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

✗ [High] File Inclusion

Path: admin/index.php, line 28

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

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\ajms

Summary:

33 Code issues found

22 [High] 4 [Medium] 7 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\lms"

Testing C:\xampp\htdocs\lms ...

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

Path: classes/Users.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: classes/Users.php, line 37

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

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

Path: classes/Users.php, line 128

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

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

Path: inc/packages.php, line 30

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

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

Path: classes/Login.php, line 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: classes/Login.php, line 49

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

✗ [Medium] SQL Injection

Path: classes/Master.php, line 210

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

✗ [Medium] SQL Injection

Path: classes/Master.php, line 263

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

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

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

✗ [Medium] Path Traversal

Path: classes/Users.php, line 119

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

✗ [Medium] Path Traversal

Path: classes/Users.php, line 208

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

✗ [High] File Inclusion

Path: admin/view\_lead/index.php, line 63

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

✗ [High] File Inclusion

Path: admin/index.php, line 26

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

✗ [High] File Inclusion

Path: admin/index.php, line 28

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

✗ [High] SQL Injection

Path: admin/view\_lead/index.php, line 3

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

✗ [High] SQL Injection

Path: admin/clients/view\_client.php, line 3

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/transactions/manage\_payment.php, line 4

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

✗ [High] SQL Injection

Path: admin/transactions/manage\_payment.php, line 17

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

✗ [High] SQL Injection

Path: admin/view\_lead/manage\_note.php, line 5

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

✗ [High] SQL Injection

Path: admin/clients/manage\_client.php, line 4

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

✗ [High] SQL Injection

Path: admin/leads/manage\_lead.php, line 3

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/sources/view\_source.php, line 4

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

✗ [High] SQL Injection

Path: admin/view\_lead/view\_note.php, line 4

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

✗ [High] SQL Injection

Path: admin/transactions/manage\_transaction.php, line 3

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

✗ [High] SQL Injection

Path: admin/view\_lead/view\_log.php, line 4

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/view\_lead/manage\_log.php, line 5

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

✗ [High] SQL Injection

Path: admin/opportunities/manage\_opportunity.php, line 3

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

✗ [High] SQL Injection

Path: admin/transactions/view\_transaction.php, line 3

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

✗ [High] SQL Injection

Path: admin/sources/manage\_source.php, line 4

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

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 44

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/transactions/manage\_payment.php, line 43

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/view\_lead/manage\_note.php, line 18

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/view\_lead/manage\_log.php, line 18

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 101

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

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\lms

Summary:

38 Code issues found

27 [High] 5 [Medium] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\66.quiz-web-application"

Testing C:\xampp\htdocs\66.quiz-web-application ...

✗ [Low] Use of Hardcoded Credentials

Path: connection.php, line 2

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

✗ [High] Cross-site Scripting (XSS)

Path: question.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: question.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] SQL Injection

Path: process.php, line 24

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\66.quiz-web-application

Summary:

4 Code issues found

3 [High] 1 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\67.helping-hand-ngo"

Testing C:\xampp\htdocs\67.helping-hand-ngo ...

✗ [Low] Use of Hardcoded Credentials

Path: user1/common/connect.php, line 4

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/common/connect.php, line 4

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

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

Path: admin/index.php, line 22

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 23

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: user1/login.php, line 20

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: user1/login.php, line 21

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 Without 'HttpOnly' Flag

Path: admin/index.php, line 22

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 23

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: user1/login.php, line 20

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: user1/login.php, line 21

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

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/js/chartinator.js, line 528

Info: Unsanitized input from data from a remote resource flows into replaceWith, 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] SQL Injection

Path: user1/post\_donation.php, line 21

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

✗ [Medium] Path Traversal

Path: user1/m\_profile.php, line 54

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

✗ [Medium] Path Traversal

Path: user1/m\_volunteer\_profile.php, line 54

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

✗ [Medium] Path Traversal

Path: admin/edit\_profile.php, line 73

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

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 111

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 116

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 121

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 125

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 129

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 133

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 137

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 141

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 145

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 149

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 153

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_v.php, line 157

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 109

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 114

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 119

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 124

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 128

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 132

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 136

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 140

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 144

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 148

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 152

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 156

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_user.php, line 160

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 75

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 80

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 85

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 90

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 94

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 98

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 102

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 110

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 114

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 118

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 122

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 126

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 130

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/more\_postdonation.php, line 134

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit.php, line 90

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 93

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 98

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 108

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 112

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 116

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 120

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 125

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_event.php, line 126

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 92

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 97

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 102

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 111

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 115

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 119

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 123

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 127

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 131

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 135

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 139

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 143

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/more\_donation.php, line 147

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/post\_donation.php, line 84

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/post\_donation.php, line 85

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/post\_donation.php, line 87

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/post\_donation.php, line 89

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/post\_donation.php, line 92

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/edit\_event.php, line 89

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/edit\_event.php, line 91

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/edit\_event.php, line 95

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 117

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 121

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 126

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 131

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 135

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_profile.php, line 161

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 117

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 121

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 126

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 131

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 135

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/m\_volunteer\_profile.php, line 161

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 165

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 171

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 177

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 183

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 189

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/edit\_profile.php, line 209

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_feedback.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_feedback.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_feedback.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_feedback.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_feedback.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 111

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 111

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 127

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 127

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 140

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/donations.php, line 140

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 80

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 81

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 82

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 83

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 93

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_post.php, line 111

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_donation.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_donation.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_donation.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_donation.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_donation.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 98

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 108

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 113

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 117

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 121

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 125

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 129

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 133

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 137

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 141

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 145

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/profile.php, line 152

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 108

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 109

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/v\_i.php, line 110

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_user.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_user.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_user.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_user.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_user.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/all\_cat.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/all\_cat.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/all\_cat.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/all\_cat.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 82

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 83

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 84

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 85

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 86

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 87

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 91

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/read\_event.php, line 92

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 107

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 108

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/donate.php, line 109

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_event.php, line 102

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_event.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_event.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_event.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_event.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/payment.php, line 97

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/payment.php, line 102

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/payment.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/payment.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 78

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 79

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 80

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 81

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 82

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 83

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_donation.php, line 84

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_money\_donation.php, line 82

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_money\_donation.php, line 83

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_money\_donation.php, line 84

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: user1/my\_money\_donation.php, line 85

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_volunteer.php, line 102

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_volunteer.php, line 103

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_volunteer.php, line 104

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_volunteer.php, line 105

Info: Unsanitized input from a database 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).

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/m\_volunteer.php, line 106

Info: Unsanitized input from a database 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).

✗ [Medium] Privacy Leak

Path: user1/m\_profile.php, line 126

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

✗ [Medium] Privacy Leak

Path: user1/m\_profile.php, line 131

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

✗ [Medium] Privacy Leak

Path: user1/m\_volunteer\_profile.php, line 126

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

✗ [Medium] Privacy Leak

Path: user1/m\_volunteer\_profile.php, line 131

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

✗ [Medium] Privacy Leak

Path: admin/edit\_profile.php, line 183

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

✗ [Medium] Privacy Leak

Path: admin/edit\_profile.php, line 189

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

✗ [High] SQL Injection

Path: admin/more\_v.php, line 25

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

✗ [High] SQL Injection

Path: admin/more\_v.php, line 31

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

✗ [High] SQL Injection

Path: admin/more\_v.php, line 36

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

✗ [High] SQL Injection

Path: admin/more\_v.php, line 41

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

✗ [High] SQL Injection

Path: admin/index.php, line 14

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

✗ [High] SQL Injection

Path: user1/login.php, line 12

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

✗ [High] SQL Injection

Path: admin/more\_user.php, line 25

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

✗ [High] SQL Injection

Path: admin/more\_user.php, line 32

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

✗ [High] SQL Injection

Path: admin/more\_user.php, line 37

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

✗ [High] SQL Injection

Path: admin/more\_user.php, line 42

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

✗ [High] SQL Injection

Path: user1/more\_postdonation.php, line 29

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

✗ [High] SQL Injection

Path: admin/edit.php, line 6

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

✗ [High] SQL Injection

Path: admin/more\_event.php, line 24

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

✗ [High] SQL Injection

Path: admin/more\_donation.php, line 24

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

✗ [High] SQL Injection

Path: user1/edit\_event.php, line 17

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

✗ [High] Path Traversal

Path: user1/m\_profile.php, line 55

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: user1/m\_volunteer\_profile.php, line 55

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: user1/register.php, line 45

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/edit\_profile.php, line 74

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: user1/login.php, line 94

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: user1/login.php, line 94

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: user1/m\_profile.php, line 177

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: user1/m\_profile.php, line 177

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: user1/m\_profile.php, line 193

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: user1/m\_profile.php, line 193

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: user1/m\_profile.php, line 209

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: user1/m\_profile.php, line 209

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: user1/m\_volunteer\_profile.php, line 177

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: user1/m\_volunteer\_profile.php, line 177

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: user1/m\_volunteer\_profile.php, line 193

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: user1/m\_volunteer\_profile.php, line 193

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: user1/m\_volunteer\_profile.php, line 209

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: user1/m\_volunteer\_profile.php, line 209

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: user1/register.php, line 168

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: user1/register.php, line 168

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: user1/register.php, line 184

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: user1/register.php, line 184

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: user1/register.php, line 200

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: user1/register.php, line 200

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/edit\_profile.php, line 220

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/edit\_profile.php, line 220

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/edit\_profile.php, line 237

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/edit\_profile.php, line 237

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/edit\_profile.php, line 254

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/edit\_profile.php, line 254

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: user1/f\_pass.php, line 90

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: user1/f\_pass.php, line 90

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/index.php, line 86

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 87

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: user1/login.php, line 78

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: user1/login.php, line 84

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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\67.helping-hand-ngo

Summary:

252 Code issues found

51 [High] 191 [Medium] 10 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\68.multi-vendor-online-groceries-management-system"

Testing C:\xampp\htdocs\68.multi-vendor-online-groceries-management-system ...

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

Path: classes/Login.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: classes/Login.php, line 45

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

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

Path: classes/Login.php, line 81

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

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

Path: inc/packages.php, line 30

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

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

Path: classes/Users.php, line 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: classes/Users.php, line 145

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

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

Path: classes/Users.php, line 164

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

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

Path: classes/Users.php, line 270

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

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

Path: classes/Users.php, line 289

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

✗ [Medium] SQL Injection

Path: classes/Master.php, line 300

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

✗ [Medium] SQL Injection

Path: classes/Master.php, line 306

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

✗ [Medium] SQL Injection

Path: classes/Master.php, line 314

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

✗ [Medium] SQL Injection

Path: orders/cart.php, line 29

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

✗ [Medium] SQL Injection

Path: orders/checkout.php, line 28

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

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

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

✗ [Medium] Path Traversal

Path: classes/Users.php, line 136

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

✗ [High] File Inclusion

Path: index.php, line 87

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

✗ [High] File Inclusion

Path: index.php, line 89

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

✗ [High] File Inclusion

Path: admin/index.php, line 27

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

✗ [High] File Inclusion

Path: admin/index.php, line 29

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

✗ [High] File Inclusion

Path: vendor/index.php, line 27

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

✗ [High] File Inclusion

Path: vendor/index.php, line 29

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

✗ [High] SQL Injection

Path: vendor/products/manage\_product.php, line 4

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

✗ [High] SQL Injection

Path: admin/vendors/manage\_vendor.php, line 3

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

✗ [High] SQL Injection

Path: vendor/categories/manage\_category.php, line 4

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

✗ [High] SQL Injection

Path: products/index.php, line 58

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

✗ [High] SQL Injection

Path: categories/view\_category.php, line 4

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

✗ [High] SQL Injection

Path: admin/products/view\_product.php, line 4

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

✗ [High] SQL Injection

Path: admin/shop\_types/manage\_shop\_type.php, line 4

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

✗ [High] SQL Injection

Path: vendor/reports/order\_reports.php, line 51

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/shop\_types/view\_shop\_type.php, line 4

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

✗ [High] SQL Injection

Path: vendor/products/view\_product.php, line 4

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

✗ [High] SQL Injection

Path: admin/reports/order\_reports.php, line 53

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

✗ [High] SQL Injection

Path: categories/manage\_category.php, line 4

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: admin/clients/manage\_client.php, line 3

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

✗ [High] SQL Injection

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

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

✗ [High] SQL Injection

Path: vendor/categories/view\_category.php, line 4

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

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 41

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

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 123

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

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 124

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: vendor/reports/order\_reports.php, line 15

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/reports/order\_reports.php, line 15

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

✗ [High] Cross-site Scripting (XSS)

Path: vendor/inc/navigation.php, line 90

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: vendor/inc/navigation.php, line 91

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

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 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/inc/navigation.php, line 124

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 125

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: classes/SystemSettings.php, line 44

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

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\68.multi-vendor-online-groceries-management-system

Summary:

53 Code issues found

37 [High] 7 [Medium] 9 [Low]