Quick Scanner

Scanning report

Summary of alerts

Risk Level	Number of vulnerabilities
High	4
medium	3
Low	2
Information	0

Vulnerabilities Count

Vulnerability Name	Count
SQL Injection	2
Cross Site Injection	3
Local File Inclusion	2
Default Vulnerable pages	2

Server Information

Target link : http://172.17.0.2

Description	Information
Server Information	Apache/2.4.25 (Debian)
Connection Type	Keep-Alive
Content_Type	text/html;charset=utf-8

More Information

Description	Links
Target WEB Pages	Quick_Scanner_targeted_url
Local Photos Inside Target	Quick_Scanner_local_photos
Internet Photos Inside Target	Quick_Scanner_internet_photos

Alert Detail

High	SQL Injection
Description	SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack
URL	http://172.17.0.2/vulnerabilities/sqli/
Method	get
Attack Url	http://172.17.0.2/vulnerabilities/sqli/
Attack Parameters	{'id': "test'", 'Submit': "Submit'"}
Prevention Methods	• Validate User Inputs • Sanitize Data by Limiting Special Characters • Enforce Prepared Statements and Parameterization • Use Stored Procedures in the Database • Raise Virtual or Physical Firewalls

	• Use whitelists, not blacklists • Establish Appropriate Privileges and Strict Access
References	1. https://owasp.org/www-project-top-ten/2017/A1_2017-Injection 2. https://en.wikipedia.org/wiki/SQL_injection 3. https://portswigger.net/web-security/sql-injection

High	**SQL Injection**
Description	SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack
URL	http://172.17.0.2/vulnerabilities/brute/
Method	get
Attack Url	http://172.17.0.2/vulnerabilities/brute/
Attack Parameters	{'username': "test'", 'password': "test'", 'Login': "Login'"}
Prevention Methods	• Validate User Inputs • Sanitize Data by Limiting Special Characters • Enforce Prepared Statements and Parameterization • Use Stored Procedures in the Database • Raise Virtual or Physical Firewalls • Use whitelists, not blacklists • Establish Appropriate Privileges and Strict Access
References	1. https://owasp.org/www-project-top-ten/2017/A1_2017-Injection 2. https://en.wikipedia.org/wiki/SQL_injection 3. https://portswigger.net/web-security/sql-injection

Medium	**Cross Site Scripting**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross- site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site.
URL	http://172.17.0.2/vulnerabilities/csp/
Method	post
Form data	{'include': '<script>alert(1)</script>'}
Attack Parameters	<script>alert(1)</script>
Prevention Methods	• Sanitize or Filter input on arrival • HTML Encode Before Inserting Untrusted Data into HTML Element Conten • Whitelisting vs blacklisting • Use appropriate response headers • Use HTTPOnly cookie flag • Implement Content Security Policy
References	1. https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS) 2. https://en.wikipedia.org/wiki/Cross-site_scripting 3. https://portswigger.net/web-security/cross-site-scripting

Medium	**Cross Site Scripting**

Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross- site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site.
URL	http://172.17.0.2/vulnerabilities/xss_s/
Method	post
Form data	{'txtName': '<script>alert(1)</script>', 'btnSign': 'Sign Guestbook', 'btnClear': 'Clear Guestbook', 'mtxMessage': '<script>alert(1)</script>'}
Attack Parameters	<script>alert(1)</script>
Prevention Methods	• Sanitize or Filter input on arrival • HTML Encode Before Inserting Untrusted Data into HTML Element Conten • Whitelisting vs blacklisting • Use appropriate response headers • Use HTTPOnly cookie flag • Implement Content Security Policy
References	1. https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS) 2. https://en.wikipedia.org/wiki/Cross-site_scripting 3. https://portswigger.net/web-security/cross-site-scripting

Medium	Cross Site Scripting
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross- site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site.
URL	http://172.17.0.2/vulnerabilities/xss_r/
Method	get
Form data	{'name': '<script>alert(1)</script>'}
Attack Parameters	<script>alert(1)</script>
Prevention Methods	• Sanitize or Filter input on arrival • HTML Encode Before Inserting Untrusted Data into HTML Element Conten • Whitelisting vs blacklisting • Use appropriate response headers • Use HTTPOnly cookie flag • Implement Content Security Policy
References	1. https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS) 2. https://en.wikipedia.org/wiki/Cross-site_scripting 3. https://portswigger.net/web-security/cross-site-scripting

High	Local file Inclusion
Description	An attacker can use Local File Inclusion (LFI) to trick the web application into exposing or running files on the web server. An LFI attack may lead to information disclosure, remote code execution, or even Cross-site Scripting (XSS). Typically, LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement.Local File Inclusion is very similar to Remote File Inclusion (RFI). However, an attacker using LFI may only include local files (not remote files like in the case of RFI).

URL	1. http://172.17.0.2/vulnerabilities/fi/.?page=php://filter/resource=/etc/passwd 2. http://172.17.0.2/vulnerabilities/fi/?page=php://filter/resource=/etc/passwd
Prevention Methods	• To avoid LFI and many other vulnerabilities, never trust user input • you need to include local files in your website or web application code, use a whitelist of allowed file names and locations • Make sure that none of these files can be replaced by the attacker using file upload functions. • Use databases â€“ donâ€™t include files on a web server that can be compromised, use a database instead
References	1. https://owasp.org/www-community/vulnerabilities/PHP_File_Inclusion 2. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion 3. https://wiki.bi0s.in/web/lfi/ 4. https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/

Low	Server Misconfigurations | Default Pages
Description	Server misconfiguration attacks exploit configuration weaknesses found in web and application servers. Many servers come with unnecessary default and sample files, including applications, configuration files, scripts, and webpages. They may also have unnecessary services enabled, such as content management and remote administration functionality. Debugging functions may be enabled or administrative functions may be accessible to anonymous users. Servers may include well-known default accounts and passwords. Failure to fully lock down or harden the server can leave improperly set file and directory permissions. All of these server misconfiguration features can be used by attackers to bypass authentication methods and gain access to sensitive information, perhaps with elevated privileges. SSL vulnerabilities such as misconfigured certificates and encryption settings, the use of default certificates, and improper authentication implementation with external systems all have the potential to compromise the confidentiality of information
URL	1. http://172.17.0.2/phpinfo.php 2. http://172.17.0.2/robots.txt
Prevention Methods	• Keep the software up to date • Disable all the default accounts and change passwords regularly • Develop strong app architecture and encrypt data which has sensitive information • Make sure that the security settings in the framework and libraries are set to secured values • Perform regular audits and run tools to identify the holes in the system
References	1. https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration 2. https://www.whitehatsec.com/glossary/content/server-misconfiguration 3. https://outpost24.com/blog/What-are-security-misconfigurations-and-how-to-prevent-them

Quick Scanner

Scanning report

Target Url's

URL Web pages Inside target

1. http://172.17.0.2/phpinfo.php
2. http://172.17.0.2/vulnerabilities/javascript/
3. http://172.17.0.2/vulnerabilities/csp
4. http://172.17.0.2/vulnerabilities/csrf
5. http://172.17.0.2/?phpids=on
6. http://172.17.0.2/vulnerabilities/csp/
7. http://172.17.0.2/vulnerabilities/xss_s/
8. http://172.17.0.2/?test=%22><script>eval(window.name)</script>
9. http://172.17.0.2/vulnerabilities/fi/?page=include.php
10. http://172.17.0.2/?doc=readme
11. http://172.17.0.2/vulnerabilities/fi/.?page=include.php
12. http://172.17.0.2/vulnerabilities/sqli_blind
13. http://172.17.0.2/
14. http://172.17.0.2/about.php
15. http://172.17.0.2/vulnerabilities/weak_id
16. http://172.17.0.2/vulnerabilities/fi/?page=file3.php
17. http://172.17.0.2/?doc=PDF
18. http://172.17.0.2/vulnerabilities/xss_d/
19. http://172.17.0.2/setup.php
20. http://172.17.0.2/.
21. http://172.17.0.2/vulnerabilities/upload
22. http://172.17.0.2/vulnerabilities/upload/
23. http://172.17.0.2/vulnerabilities/javascript
24. http://172.17.0.2/instructions.php?doc=PHPIDS-license
25. http://172.17.0.2/vulnerabilities/captcha/
26. http://172.17.0.2/vulnerabilities/exec
27. http://172.17.0.2/vulnerabilities/weak_id/
28. http://172.17.0.2/index.php
29. http://172.17.0.2/vulnerabilities/sqli/
30. http://172.17.0.2/security.php
31. http://172.17.0.2/vulnerabilities/xss_r
32. http://172.17.0.2/vulnerabilities/sqli_blind/
33. http://172.17.0.2/vulnerabilities/xss_r/
34. http://172.17.0.2/ids_log.php

35. http://172.17.0.2/vulnerabilities/fi/?page=file2.php
36. http://172.17.0.2/vulnerabilities/brute/
37. http://172.17.0.2/vulnerabilities/brute
38. http://172.17.0.2/vulnerabilities/captcha
39. http://172.17.0.2/vulnerabilities/fi/?page=file1.php
40. http://172.17.0.2/vulnerabilities/exec/
41. http://172.17.0.2/?doc=PHPIDS-license
42. http://172.17.0.2/instructions.php
43. http://172.17.0.2/vulnerabilities/csrf/
44. http://172.17.0.2/vulnerabilities/sqli
45. http://172.17.0.2/vulnerabilities/xss_d
46. http://172.17.0.2/?doc=changelog
47. http://172.17.0.2/?doc=copying
48. http://172.17.0.2/vulnerabilities/xss_s

Go Back

Quick Scanner

Scanning report

Local Images

Local images Inside target

1. dvwa/images/login_logo.png
2. ../../dvwa/images/logo.png
3. dvwa/images/lock.png
4. dvwa/images/spanner.png

Go Back

Quick Scanner

Scanning report

Internet Images

Internet Images Inside target Inside target

Go Back