V-Scanner

Web-based Assessment Tool For Web Applications

Scan Report

August 02, 2022

Summary

This document reports on the results of an automatic security scan. The scan started at 2022:08:02 11:29:16 and ended at 2022:08:02 11:29:29. The report first summarises the results found. Then, for target host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Table Of Content

Results Overview

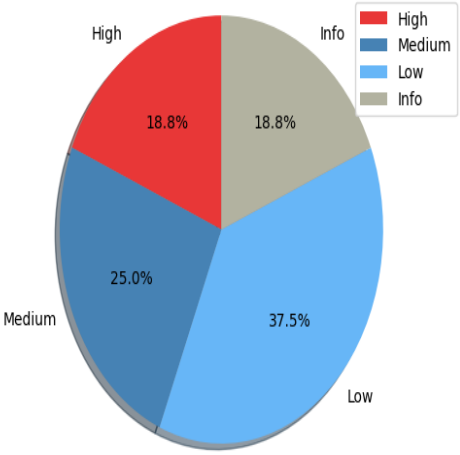
High	Medium	Low
This level indicates that an attacker can fully compromise the confidentiality, integrity, or availability of a system without specialized access, user interaction, or circumstances that are beyond the attacker’s control. It is very likely that the attacker may be able to escalate the attack to the operating system and other systems.	This level indicates that an attacker can partially compromise the confidentiality, integrity, or availability of a target system. They may need specialized access, user interaction, or circumstances that are beyond the attacker’s control. Such vulnerabilities may be used together with other vulnerabilities to escalate an attack.	This level indicates that an attacker can compromise the confidentiality, integrity, or availability of a target system in a limited way. They need specialized access, user interaction, or circumstances that are beyond the attacker’s control. To escalate an attack, such vulnerabilities must be used together with other vulnerabilities.

http://127.0.0.1/sqli_1.php

Overall risk level:

High

Issues Ratio:



Scan Information:

Start Time:	2022:08:02 11:29:16
Finish Time:	2022:08:02 11:29:29
Scan Duration:	0 min, 12 sec
Crawled Pages:	7
Scan Status:	Finished

Results For Target

➔ 127.0.0.1 ➔ Headers

Content-Security-Policy

Description:

Control the loading of resources in a website and block any request that is intended for malicious or unknown resource.

Result: ❗

It was detected that target website is missing Content-Security-Policy header.

Impact:

Absense of 'Content-Security-Policy' increases the chances of Cross-site Scripting (XSS) and other data injection attacks which lead to data theft, site defacement and malware distribution.

Solution:

There can be various possible directives for Content-Security-Policy header and these vary according to the nature of the website. Here is a guide to find suitable directives for your website Content-Security-Policy

Reference:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Strict-Transport-Security (HSTS)

Description:

HSTS ensures that website will only be accessible over HTTPS, It protects the website from ssl-striping attacks.

Result: ❗

It was detected that target website is missing Strict-Transport-Security header.

Impact:

Absense of Strict-Transport-Security caused ssl-striping which in turn leads to Man-in-the-middle (MITM) and other eavesdropping attacks that can result in data theft, credential stealing and payload delivery.

Solution:

Recommended directive for HSTS is 'Strict-Transport-Security: max-age=expire-time'. There are also some other optional directives can be used with recommended directive. For detail guide check it out Strict-Transport-Security

Reference:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

X-Content-Type-Options

Description:

X-Content-Type-Options header protect against MIME type sniffing attacks.

Result: ❗

It was detected that target website is missing X-Content-Type-Options header.

Impact:

Absense of X-Content-Type-Options caused MIME sniffing which is a technique used by browsers to determine the type of resource that received in response. This technique can be used by attackers to perform XSS.

Solution:

Recommended directive for X-Content-Type-Options header is 'X-Content-Type-Options: nosniff'.

Reference:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

X-Frame-Options

Description:

X-Frame-Options header protect against attacks like ClickJacking or Frame Injection. It tells browser that whether it allowed to load the website in an iframe or not.

Result: ❗

It was detected that target website is missing X-Frame-Options header.

Impact:

Absense of X-Frame-Options caused website to load in an iframe and this can result in ClickJacking which in turn used for credential stealing & phishing.

Solution:

Recommended directive for X-Frame-Options header is 'X-Frame-Options: DENY'. For allowing iframe for same-domain the directive is 'X-Frame-Options: SAME-ORIGIN'.

Reference:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

X-XSS-Protection

Description:

X-XSS-Protection header protect against Cross-site Scripting (XSS) attacks. It enable XSS-filtering in the browser and block page from rendering if it contain any malicious javascript.

Result: ❗

It was detected that target website is missing X-XSS-Protection header.

Impact:

Absense of X-XSS-Protection header increases the chances of Cross-site Scripting (XSS) vulnerabilities.

Solution:

Recommended directive for X-XSS-Protection header is 'X-XSS-Protection: 1; mode=block'.

Reference:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

Results For Target

→ 127.0.0.1 → Vulnerabilities

Cross-site Scripting (XSS)

Description:

Cross-site scripting is a vulnerability that comes under the category of injections, it allows an attacker to compromise the interactions of the user with the vulnerable application. XSS ranked among one of the top 10 security risks by Owasp. OWASP Top 10:2021

Result: ✓

It was detected that target website is not vulnerable to XSS.

Reference:

https://www.acunetix.com/websitesecurity/cross-site-scripting/

SQL Injection (SQLi)

Description:

SQL injection is one of the most notorious vulnerabilities still found in web applications, although it’s a very old vulnerability still it is included in the top 10 most common web application vulnerabilities. SQLi ranked among one of the top 10 security risks by Owasp. OWASP Top 10:2021

Result: ❗

It was detected that target website is vulnerable to SQL Injection (Inference).

Impact:

This vulnerability has severe effects on the privacy of users and the data of the application, it allows an attacker to interfere with a database that the application is used for retrieving data. A successful SQL injection can expose the data of all the application users including their personal information submitted to the vulnerable application.

Solution:

Perform validation and sanitization on all inputs of the application. Escape special characters from user input & dont construct dynamic SQL queries from direct user-supplied input instead use parameterized queries. Turn off SQL errors and use whitelist for allowed characters in the input.

Reference:

https://www.acunetix.com/websitesecurity/sql-injection/

High	Easy	100%
Severity	Exploitability	Confidence

Cross-site Request Forgery (CSRF)

Description:

This vulnerability forces users of the vulnerable application to perform unintended actions without their consent, it bypasses the same-origin policy which avoid different websites to interfere with each other. CSRF ranked among one of the top 10 security risks by Owasp. OWASP Top 10:2021

Result: ❗

It was detected that target website is vulnerable to Cross-site Request Forgery (CSRF).

Impact:

By exploiting CSRF an attacker can perform actions like changing email, and password or making funds transfer by sending malicious requests to the application by the authenticated user without knowing about it.

Solution:

Use anti-csrf tokens to mitigate the CSRF attacks. For securing cookies from CSRF, use Samesite directive in the Set-Cookie response header. For further detail check it out CSRF Prevention.

Reference:

https://owasp.org/www-community/attacks/csrf

Medium	Hard	95%
Severity	Exploitability	Confidence

ClickJacking (Frame Injection)

Description:

Clickjacking is an attack that fools users into thinking they are clicking on one thing when they are actually clicking on another. Users think they are using a web page's normal UI, but in fact there is a hidden UI in control; in other words, the UI has been redressed. When users click something they think is safe, the hidden UI performs a different action.

Result: ❗

It was detected that target website is vulnerable to ClickJacking.

Impact:

By exploiting CSRF an attacker can perform actions like changing email, and password or making funds transfer by sending malicious requests to the application by the authenticated user without knowing about it.

Solution:

Use X-Frame-Options header in the response headers of the application. Recommended directive for X-Frame-Options is 'DENY'. Content-Security-Policy header can also be used to avoid ClickJacking.

Reference:

https://owasp.org/www-community/attacks/Clickjacking

Medium

Severity

Easy

Exploitability

95%

Confidence

Results For Target

➔ 127.0.0.1 ➔ Footprinting

Port	Status	Service	Banner
80	Open	HTTP	None
139	Open	NETBIOS-SSN	None
445	Open	MICROSOFT-DS	None
902	Open	NONE	220 VMware Authentication Daemon Version 1.10: SSL Required, ServerDaemonProtocol:SOAP, MKSDisplayProtocol:VNC , VMXARGS supported, NFCSSL supported/t, SHA256 supported
631	Open	IPP	None
5989	Open	NONE	None
8000	Open	NONE	None
8081	Open	TPROXY	None

Recommendation:

Keep unnecessary ports close because open ports are like open doors for the attacker. An attacker can launch various attacks against services running on these open ports. For best practices set custom banner to the service running on each port so that attacker cant get an idea about the running service and its version. More

Results For Target

➔ 127.0.0.1 ➔ Warnings

Name	Value	Warning
Protocol	❗ HTTP	Target is running over HTTP which is vulnerable to Man-in-the-middle attacks (MITM).
Server	❗ Apache/2.4.7 (Ubuntu)	Senstive information is being leaked
Technology	❗ PHP/5.5.9-1ubuntu4.14	Senstive information is being leaked
OS	❗ Unknown	Senstive information is being leaked
Cookies	❗ {'cookies': True, 'Secure': False, 'HttpOnly': False, 'SameSite': False}	Incorrect directives! Secure flag must be set in cookies. HttpOnly flag must be set in cookies. SameSite flag must be set in cookies.

Recommendation:

Above information help an attacker to build an effective attack against the target.

- Use HTTPS or HSTS to avoid MITM attacks and it will also improve target SEO as well. More
- Server field in the headers must not leak any useful information. Set custom value to Server field. More
- Any information related to underlying technology should be hidden. Set custom value in the headers (X-Powered-By) to hide the leakage. More
- Cookies must have safe flags in order to avoid cookie stealing attacks. Some recommended flags are Secure, HttpOnly and SameSite. More