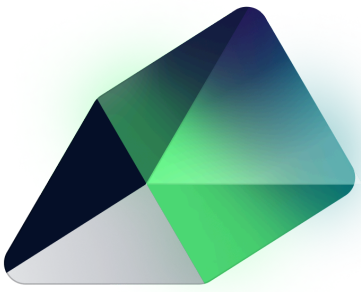
September 24, 2025

# Vulnerability Scan Report

Prepared By

**HostedScan Security**



hostedscan.com

# Overview

# 1 Executive Summary

Vulnerability scans were conducted on select servers, networks, websites, and applications. This report contains the discovered potential vulnerabilities from these scans. Vulnerabilities have been classified by severity. Higher severity indicates a greater risk of a data breach, loss of integrity, or availability of the targets.

## 1.1 Total Vulnerabilities

Below are the total number of vulnerabilities found by severity. Critical vulnerabilities are the most severe and should be evaluated first. An accepted vulnerability is one which has been manually reviewed and classified as acceptable to not fix at this time, such as a false positive detection or an intentional part of the system's architecture.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 3 | 3 | 0 |


## 1.2 Report Coverage

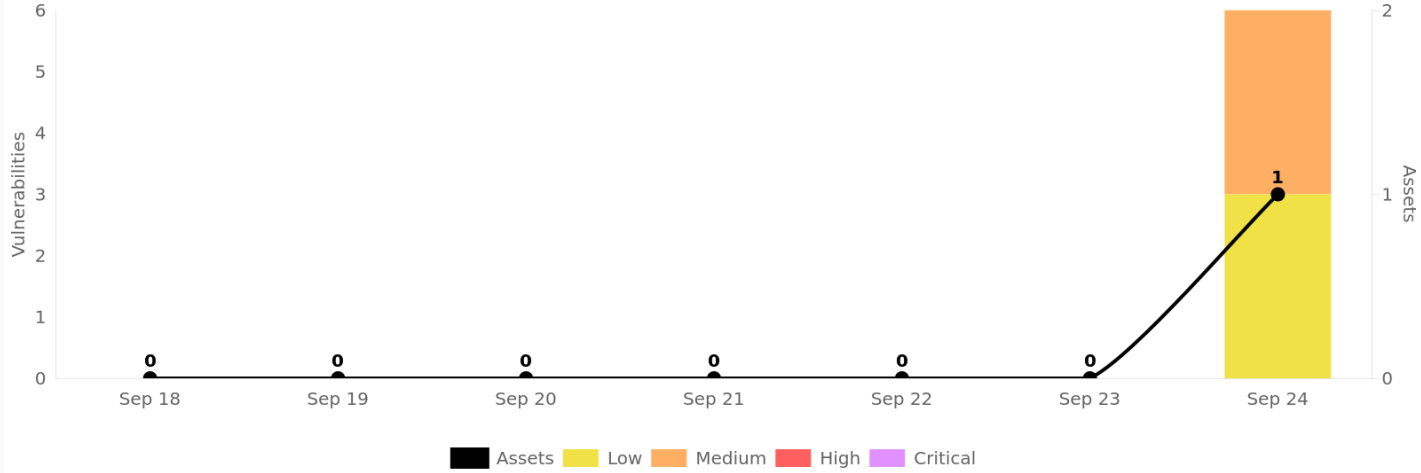
This report includes findings for **1 target** scanned. Each target is a single URL, IP address, or fully qualified domain name (FQDN).

| 1 | 6 |
|---|---|
| Total Targets | Passive Web Application Vulnerabilities |

# 2 Trends

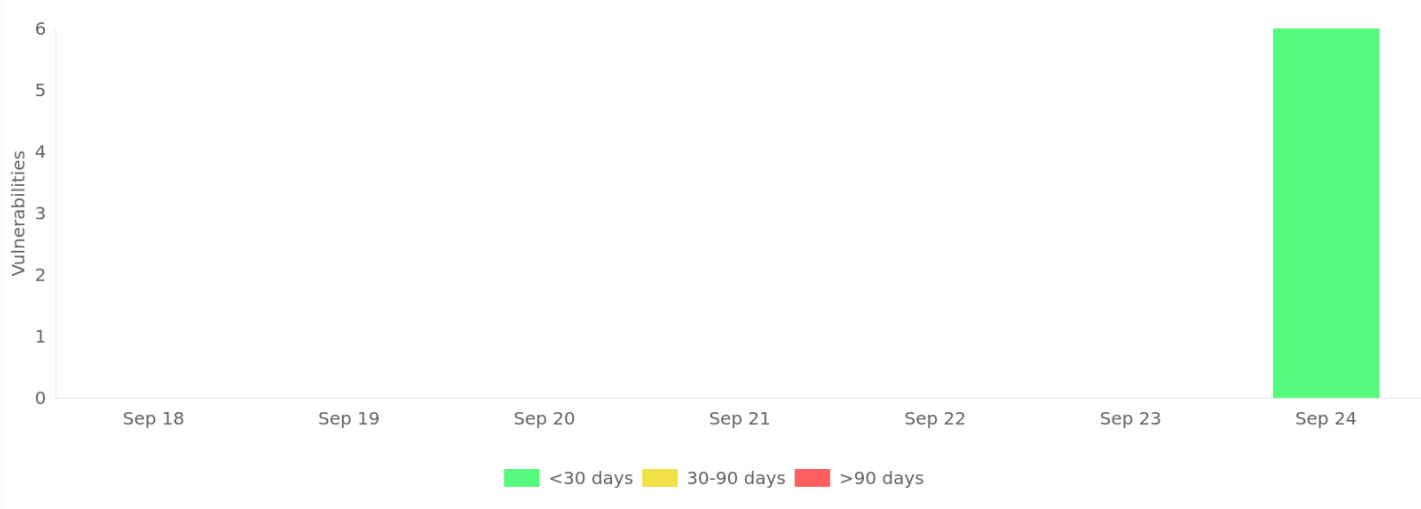
## 2.1 Open Risks

Total number of vulnerabilities grouped by severity level.




## 2.2 Exposure Window

Total number of unresolved vulnerabilities grouped by age (time since first detection).



# 3 Vulnerabilities By Target

This section contains the vulnerability findings for each scanned target. Prioritization should be given to the targets with the highest severity vulnerabilities. However, it is important to take into account the purpose of each system and consider the potential impact a breach or an outage would have for the particular target.

## 3.1 Targets Summary (1)

The number of potential vulnerabilities found for each target by severity.

| Target | Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|---|
| http://testphp.vulnweb.com | 0 | 0 | 3 | 3 | 0 |

## 3.2 Target Breakdowns

Details for the potential vulnerabilities found for each target by scan type.

# http://testphp.vulnweb.com

### Total Risks

| 0 | 0 | 3 | 3 | 0 |
|---|---|---|---|---|


| Passive Web Application Vulnerabilities | Severity | First Detected | Last Detected |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | Medium | 0 days ago | 0 days ago |
| Content Security Policy (CSP) Header Not Set | Medium | 0 days ago | 0 days ago |
| Missing Anti-clickjacking Header | Medium | 0 days ago | 0 days ago |
| X-Content-Type-Options Header Missing | Low | 0 days ago | 0 days ago |
| Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) | Low | 0 days ago | 0 days ago |
| Server Leaks Version Information via "Server" HTTP Response Header Field | Low | 0 days ago | 0 days ago |

# 4 Passive Web Application Vulnerabilities

The OWASP ZAP Passive Web Application scan crawls the pages of a website or web application. The passive scan inspects each page as well as the requests and responses sent between the server. The passive scan checks for vulnerabilities such as cross-domain misconfigurations, insecure cookies, vulnerable Javascript dependencies, and more.

## 4.1 Total Vulnerabilities

Total number of vulnerabilities found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 3 | 3 | 0 |


## 4.2 Vulnerabilities Breakdown

Summary list of all detected vulnerabilities.

| Title | Severity | Open | Accepted |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | Medium | 1 | 0 |
| Content Security Policy (CSP) Header Not Set | Medium | 1 | 0 |
| Missing Anti-clickjacking Header | Medium | 1 | 0 |
| X-Content-Type-Options Header Missing | Low | 1 | 0 |
| Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) | Low | 1 | 0 |
| Server Leaks Version Information via "Server" HTTP Response Header Field | Low | 1 | 0 |

## 4.3 Vulnerability Details

Detailed information about each potential vulnerability found by the scan.

# Absence of Anti-CSRF Tokens

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
| --- | --- | --- |
| Medium | 1 target | 0 days ago |

## Description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

## Solution

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Phase: Implementation

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Note that this can be bypassed using XSS.

Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note that this can be bypassed using XSS.

Use the ESAPI Session Management control.

This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Phase: Implementation

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

### Instances (1 of 40)

uri: http://testphp.vulnweb.com/
method: GET
evidence: <form action="search.php?test=query" method="post">
otherinfo: No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, _csrfToken, data[_Token][key], _wpnonce] was found in the following HTML form: [Form 1: "goButton" "searchFor" ].

### References

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/352.html

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# Content Security Policy (CSP) Header Not Set

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
| --- | --- | --- |
| Medium | 1 target | 0 days ago |

## Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

## Solution

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

### Instances (1 of 48)

uri: http://testphp.vulnweb.com/
method: GET

### References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://www.w3.org/TR/CSP/
https://w3c.github.io/webappsec-csp/
https://web.dev/articles/csp
https://caniuse.com/#feat=contentsecuritypolicy
https://content-security-policy.com/

| Vulnerable Target | First Detected | Last Detected |
| --- | --- | --- |
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# Missing Anti-clickjacking Header

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
| --- | --- | --- |
| Medium | 1 target | 0 days ago |

## Description

The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

## Solution

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.

If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

### Instances (1 of 44)

uri: http://testphp.vulnweb.com/
method: GET
param: x-frame-options

### References

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

| Vulnerable Target | First Detected | Last Detected |
| --- | --- | --- |
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# X-Content-Type-Options Header Missing

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
| --- | --- | --- |
| Low | 1 target | 0 days ago |

## Description

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

## Solution

Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.

If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

### Instances (1 of 68)

uri: http://testphp.vulnweb.com/
method: GET
param: x-content-type-options
otherinfo: This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.

### References

https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
https://owasp.org/www-community/Security_Headers

| Vulnerable Target | First Detected | Last Detected |
| --- | --- | --- |
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
|---|---|---|
| Low | 1 target | 0 days ago |

## Description

The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

## Solution

Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

### Instances (1 of 62)

uri: http://testphp.vulnweb.com/
method: GET
evidence: X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1

### References

https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework
https://www.troyhunt.com/shhh-dont-let-your-response-headers/

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# Server Leaks Version Information via "Server" HTTP Response Header Field

| SEVERITY | AFFECTED TARGETS | LAST DETECTED |
| --- | --- | --- |
| Low | 1 target | 0 days ago |

## Description

The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

## Solution

Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

### Instances (1 of 74)

uri: http://testphp.vulnweb.com/
method: GET
evidence: nginx/1.19.0

### References

https://httpd.apache.org/docs/current/mod/core.html#servertokens
https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648552(v=pandp.10)
https://www.troyhunt.com/shhh-dont-let-your-response-headers/

| Vulnerable Target | First Detected | Last Detected |
| --- | --- | --- |
| http://testphp.vulnweb.com | 0 days ago | 0 days ago |

# 5 Glossary

**Accepted Vulnerability**

An accepted vulnerability is one which has been manually reviewed and classified as acceptable to not fix at this time, such as a false positive scan result or an intentional part of the system's architecture.

**Fully Qualified Domain Name (FQDN)**

A fully qualified domain name is a complete domain name for a specific website or service on the internet. This includes not only the website or service name, but also the top-level domain name, such as .com, .org, .net, etc. For example, 'www.example.com' is an FQDN.

**Passive Web Application Vulnerabilities**

The OWASP ZAP Passive Web Application scan crawls the pages of a website or web application. The passive scan inspects each page as well as the requests and responses sent between the server. The passive scan checks for vulnerabilities such as cross-domain misconfigurations, insecure cookies, vulnerable Javascript dependencies, and more.

**Vulnerability**

A weakness in the computational logic (e.g., code) found in software and hardware components that, when exploited, results in a negative impact to confidentiality, integrity, or availability. Mitigation of the vulnerabilities in this context typically involves coding changes, but could also include specification changes or even specification deprecations (e.g., removal of affected protocols or functionality in their entirety).

**Target**

A target represents target is a single URL, IP address, or fully qualified domain name (FQDN) that was scanned.

**Severity**

Severity represents the estimated impact potential of a particular vulnerability. Severity is divided into 5 categories: Critical, High, Medium, Low and Accepted.

**CVSS Score**

The CVSS 3.0 score is a global standard for evaluating vulnerabilities with a 0 to 10 scale. CVSS maps to threat levels:
0.1 - 3.9 = Low
4.0 - 6.9 = Medium
7.0 - 8.9 = High
9.0 - 10.0 = Critical

**EPSS Score**

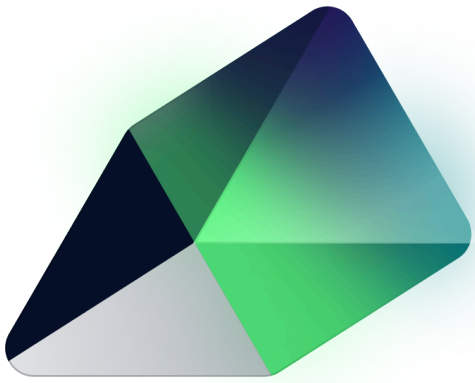
The EPSS score is the estimated probability that a given vulnerability will be exploited in the wild within the next 30 days, on a 0% to 100% scale.

This report was prepared using

## HostedScan Security ®

For more information, visit **hostedscan.com**

Founded in Seattle, Washington in 2019, HostedScan, LLC. is dedicated to making continuous vulnerability scanning and risk management much more easily accessible to more businesses.



HostedScan, LLC.

2212 Queen Anne Ave N
Suite #521
Seattle, WA 98109

Terms & Policies
hello@hostedscan.com