

March 13, 2024

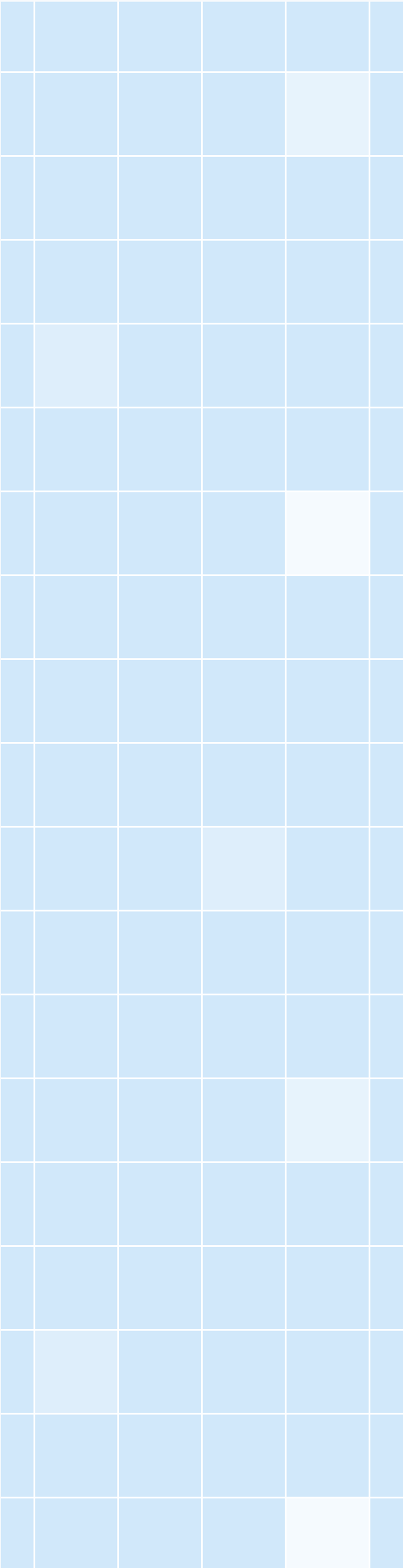
Vulnerability Scan Report

prepared by

HostedScan Security



hostedscan.com



Overview

1	Executive Summary	3
2	Risks By Target	4
3	Passive Web Application Vulnerabilities	6
4	Glossary	21

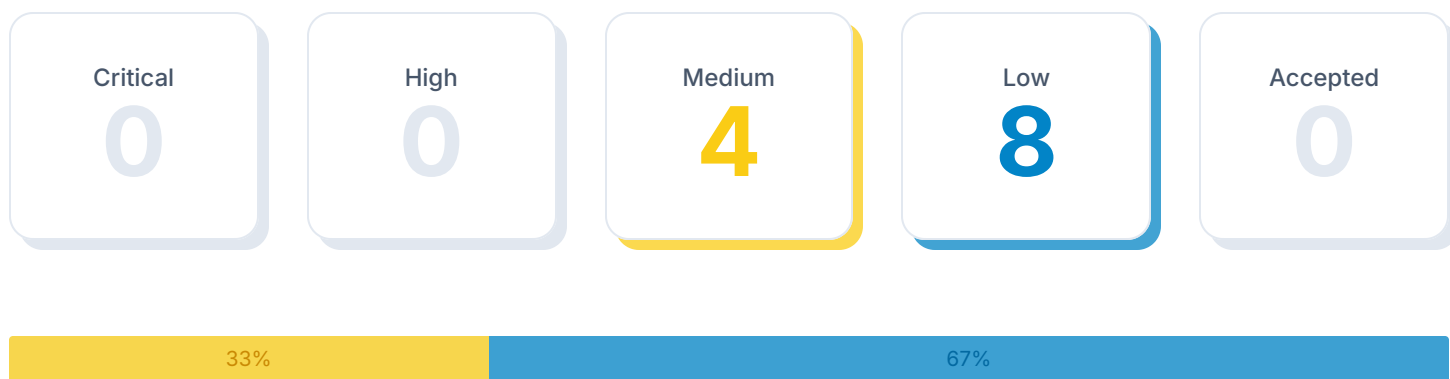


1 Executive Summary

Vulnerability scans were conducted on selected servers, networks, websites, and applications. This report contains the discovered potential risks from these scans. Risks have been classified into categories according to the level of threat and degree of potential harm they may pose.

1.1 Total Risks

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



1.2 Report Coverage

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

Vulnerability Categories

12






Passive Web Application Vulnerabilities

2 Risks By Target

This section contains the vulnerability findings for each target that was scanned. Prioritize the most vulnerable assets first.

2.1 Targets Summary

The total number of risks found for each target, by severity.

Target	 Critical	 High	 Medium	 Low	 Accepted
 MUPAPAT	0	0	4	8	0

2.2 Target Breakdowns

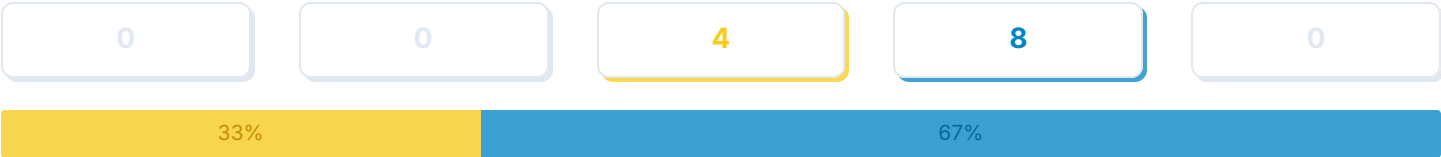
The risks discovered for each target.



Target - <https://arz-apat.ru/>

MUPAPAT

Total Risks



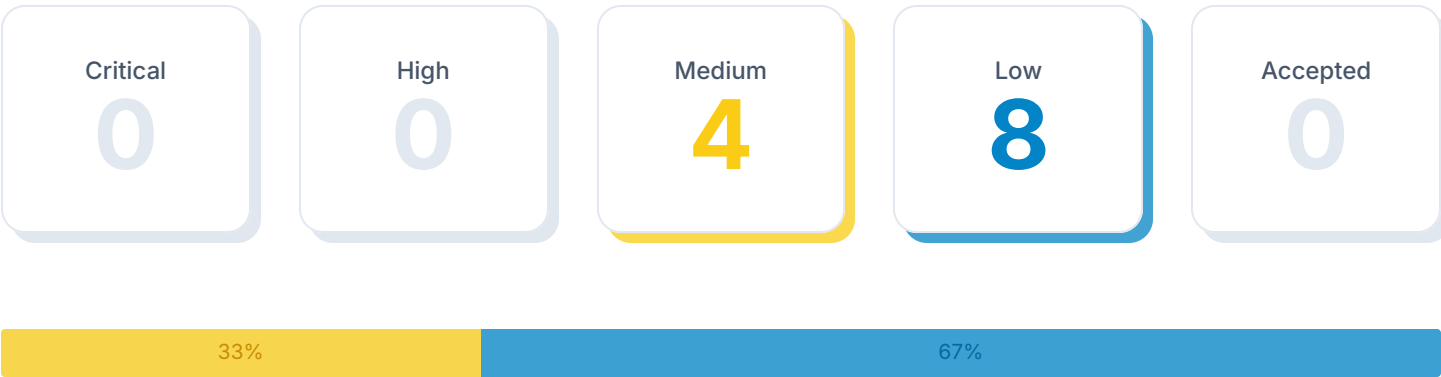
Passive Web Application Vulnerabilities	Threat Level	First Detected	Last Detected
Absence of Anti-CSRF Tokens	● Medium	0 days ago	0 days ago
Vulnerable JS Library	● Medium	0 days ago	0 days ago
Missing Anti-clickjacking Header	● Medium	0 days ago	0 days ago
Content Security Policy (CSP) Header Not Set	● Medium	0 days ago	0 days ago
X-Content-Type-Options Header Missing	● Low	0 days ago	0 days ago
Cross-Domain JavaScript Source File Inclusion	● Low	0 days ago	0 days ago
Cookie Without Secure Flag	● Low	0 days ago	0 days ago
Cookie No HttpOnly Flag	● Low	0 days ago	0 days ago
Server Leaks Version Information via "Server" HTTP Response Header Field	● Low	0 days ago	0 days ago
Strict-Transport-Security Header Not Set	● Low	0 days ago	0 days ago
Cookie without SameSite Attribute	● Low	0 days ago	0 days ago
Cookie with SameSite Attribute None	● Low	0 days ago	0 days ago

3 Passive Web Application Vulnerabilities

The OWASP ZAP passive web application scan crawls the pages of a web application. It inspects the web pages 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 js dependencies, and more.

3.1 Total Risks

Total number of risks found by severity.



3.2 Risks Breakdown

Summary list of all detected risks.

Title	Threat Level	Open	Accepted
Absence of Anti-CSRF Tokens	Medium	1	0
Vulnerable JS Library	Medium	1	0
Missing Anti-clickjacking Header	Medium	1	0
Content Security Policy (CSP) Header Not Set	Medium	1	0
X-Content-Type-Options Header Missing	Low	1	0
Cross-Domain JavaScript Source File Inclusion	Low	1	0
Cookie Without Secure Flag	Low	1	0
Cookie No HttpOnly Flag	Low	1	0
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	1	0

Cookie without SameSite Attribute	<div><div></div>Low</div>	1	0
Cookie with SameSite Attribute None	<div><div></div>Low</div>	1	0

3.3 Full Risk Details

Detailed information about each risk found by the scan.

Absence of Anti-CSRF Tokens

● Medium

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 2)

uri: https://arz-apat.ru/company/feedback/
method: GET
evidence: <form action="/company/feedback/" 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, data[_Token][key]] was found in the following HTML form: [Form 1: "PARAMS_HASH" "sessid" "submit" "user_email" "user_name"].

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
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Vulnerable JS Library

●

Medium

Description

The identified library jquery, version 1.9.0 is vulnerable.

Solution

Please upgrade to the latest version of jquery.

Instances (1 of 1)

uri: <https://ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js>
method: GET
evidence: /1.9.0/jquery.min.js
otherinfo: CVE-2020-11023 CVE-2020-11022 CVE-2015-9251 CVE-2019-11358

References

<https://github.com/jquery/jquery/issues/2432>
<http://blog.jquery.com/2016/01/08/jquery-2-2-and-1-12-released/>
<http://research.insecurelabs.org/jquery/test/>
<https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/>
<https://nvd.nist.gov/vuln/detail/CVE-2019-11358>
<https://github.com/advisories/GHSA-rmxg-73gg-4p98>
<https://nvd.nist.gov/vuln/detail/CVE-2015-9251>
<https://github.com/jquery/jquery/commit/753d591aea698e57d6db58c9f722cd0808619b1b>
<https://bugs.jquery.com/ticket/11974>
<https://github.com/jquery/jquery.com/issues/162>
<https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Missing Anti-clickjacking Header

● Medium

Description

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

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 100)

uri: https://arz-apat.ru/
method: GET
param: x-frame-options

References

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

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Content Security Policy (CSP) Header Not Set

● Medium

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 100)

uri: <https://arz-apat.ru/>
method: GET

References

https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
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
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

X-Content-Type-Options Header Missing

● Low

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 100)

uri: https://arz-apat.ru/
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
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Cross-Domain JavaScript Source File Inclusion

● Low

Description

The page includes one or more script files from a third-party domain.

Solution

Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

Instances (1 of 100)

uri: https://arz-apat.ru/
method: GET
param: https://ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js
evidence: <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js"></script>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Cookie Without Secure Flag

● Low

Description

A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.

Solution

Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.

Instances (1 of 2)

uri: https://arz-apat.ru/
method: GET
param: PHPSESSID
evidence: Set-Cookie: PHPSESSID

References

https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Cookie No HttpOnly Flag

● Low

Description

A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Solution

Ensure that the HttpOnly flag is set for all cookies.

Instances (1 of 4)

uri: https://mc.yandex.ru/metrika/tag.js
method: GET
param: _yasc
evidence: Set-Cookie: _yasc

References

<https://owasp.org/www-community/HttpOnly>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

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

● Low

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 100)

uri: http://arz-apat.ru/sitemap.xml
method: GET
evidence: nginx/1.24.0

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://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
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Strict-Transport-Security Header Not Set

● Low

Description

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

Solution

Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

Instances (1 of 100)

uri: <https://arz-apat.ru/>
method: GET

References

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
<https://owasp.org/www-community/Security-Headers>
https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
<https://caniuse.com/stricttransportsecurity>
<https://datatracker.ietf.org/doc/html/rfc6797>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Cookie without SameSite Attribute

● Low

Description

A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Solution

Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Instances (1 of 4)

uri: <https://mc.yandex.ru/metrika/tag.js>
method: GET
param: _yasc
evidence: Set-Cookie: _yasc

References

<https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

Cookie with SameSite Attribute None

● Low

Description

A cookie has been set with its SameSite attribute set to "none", which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Solution

Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Instances (1 of 6)

uri: <https://mc.yandex.ru/metrika/tag.js>
method: GET
param: i
evidence: Set-Cooki

References

<https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site>

Vulnerable Target	First Detected	Last Detected
https://arz-apat.ru/ MUPAPAT	0 days ago	0 days ago

4 Glossary

Accepted Risk

An accepted risk is one which has been manually reviewed and classified as acceptable to not fix at this time, such as a false positive 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 web application. It inspects the web pages 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 js dependencies, and more.

Risk

A risk is a finding from a vulnerability scan. Each risk is a potential security issue that needs review. Risks are assigned a threat level which represents the potential severity.

Target

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

Threat Level

The threat level represents the estimated potential severity of a particular risk. Threat level is divided into 4 categories: High, Medium, Low and Accepted.

Threat Level

The threat level represents the estimated potential severity of a particular risk. Threat level 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

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