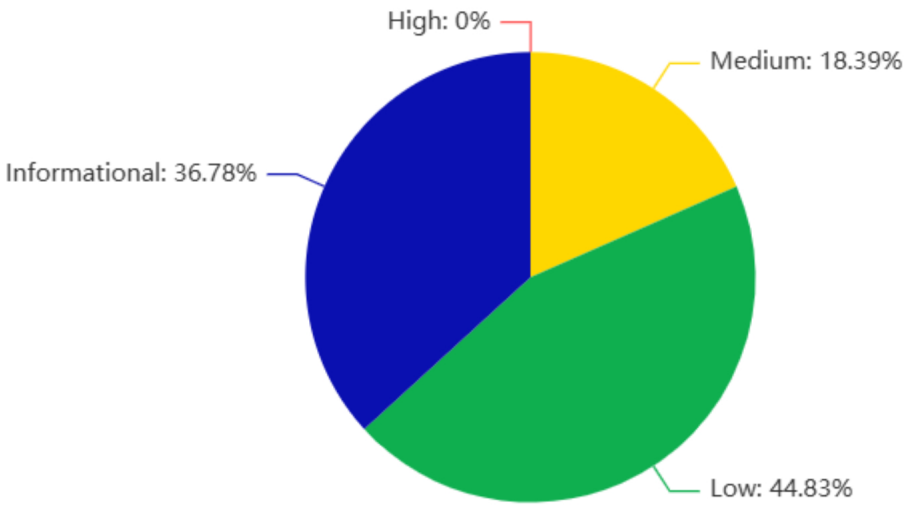
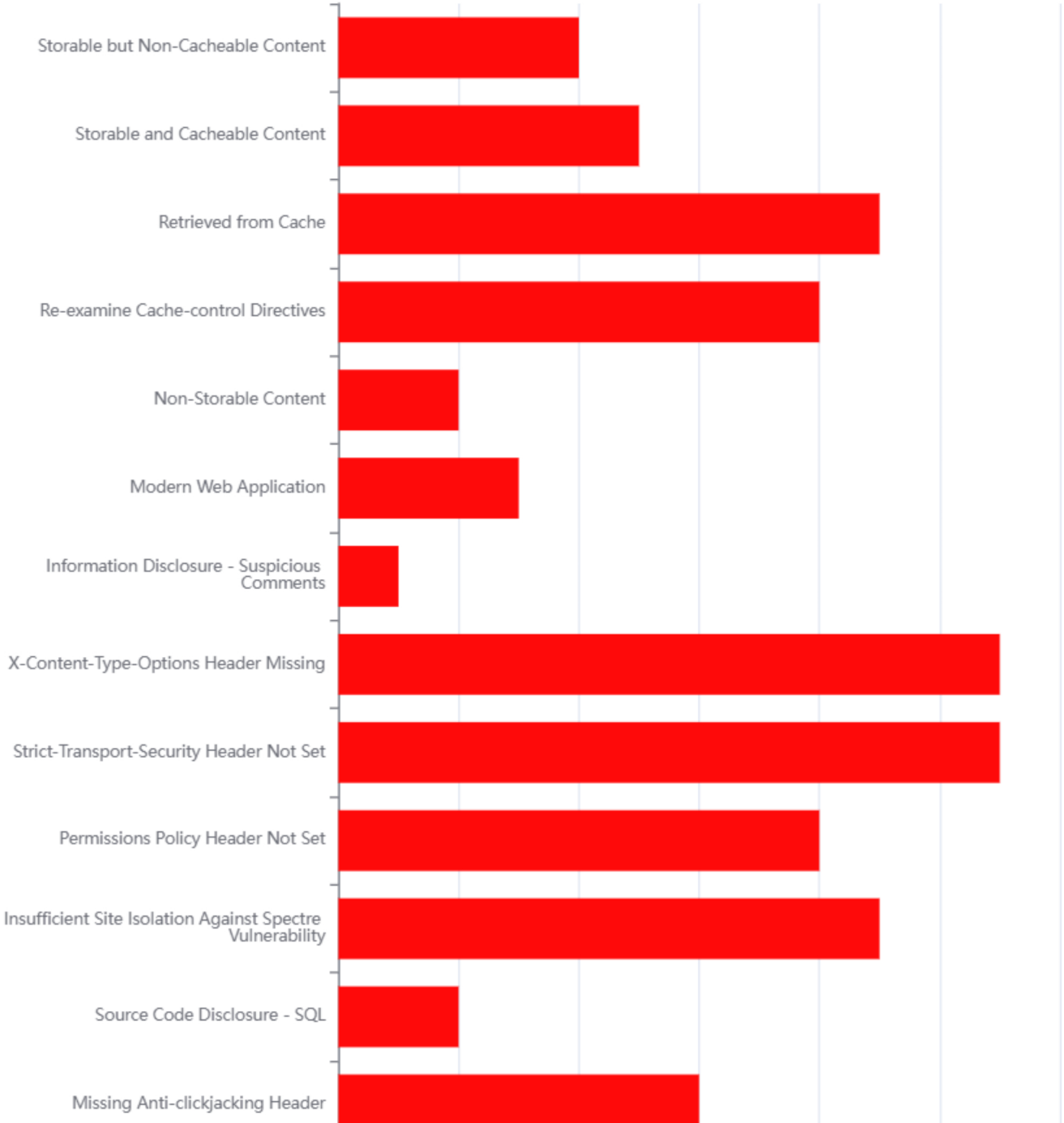
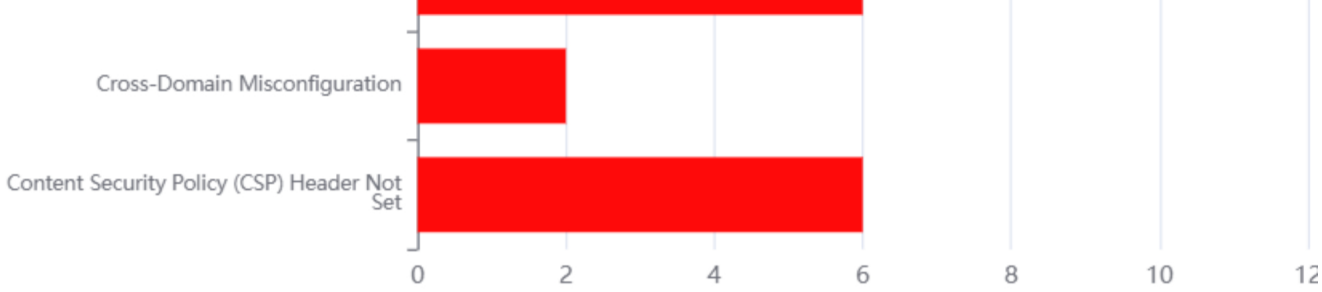
Security Test Report

Vulnerability Severity



Vulnerability Count





AI Summary 1/1:

Report Metadata

- **Generated On:** Wed, 9 Apr 2025 03:08:03
- **Scanned Site:** https://splix.io
- **Total Vulnerabilities Identified:** 15

Top 3 Critical Vulnerabilities

Content Security Policy (CSP) Header Not Set

- **Risk Level:** Medium (High)
- **Count:** 6
- **Exploitability:** Medium
- **Impact:** 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.
- **Why is it critical?** It is critical because it could allow malicious attacks such as Cross Site Scripting (XSS) and data injection attacks.
- **Mitigation Strategy:** Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
- **References:** https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy

Cross-Domain Misconfiguration

- **Risk Level:** Medium (Medium)
- **Count:** 2
- **Exploitability:** Medium
- **Impact:** Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.
- **Why is it critical?** It is critical because it may allow unauthorized access to sensitive data.
- **Mitigation Strategy:** Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
- **References:** https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

Missing Anti-clickjacking Header

- **Risk Level:** Medium (Medium)
- **Count:** 6

- **Exploitability:** Medium
- **Impact:** The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.
- **Why is it critical?** It is critical because it may allow malicious attacks such as ClickJacking.
- **Mitigation Strategy:** 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.
- **References:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Remaining Vulnerabilities

Source Code Disclosure - SQL

Insufficient Site Isolation Against Spectre Vulnerability

Permissions Policy Header Not Set

Strict-Transport-Security Header Not Set

X-Content-Type-Options Header Missing

Information Disclosure - Suspicious Comments

Modern Web Application

Non-Storable Content

Re-examine Cache-control Directives

Retrieved from Cache

Storable and Cacheable Content

Storable but Non-Cacheable Content

Additional Security Recommendations

- Identify recurring patterns in vulnerabilities and suggest general security improvements.
- Highlight missing security controls (e.g., weak authentication, lack of encryption, missing headers).
- Suggest proactive defense strategies.

Final Risk Assessment

- **Overall Security Rating:** Medium
- **Key Areas of Concern:** Content Security Policy (CSP) Header Not Set, Cross-Domain Misconfiguration, Missing Anti-clickjacking Header.
- **Critical Fixes Required:** Content Security Policy (CSP) Header Not Set, Cross-Domain Misconfiguration, Missing Anti-clickjacking Header.
- **Suggested Next Steps:** Implement Content Security Policy (CSP) Header, Configure Cross-Origin Resource Sharing (CORS) properly, Set Anti-clickjacking Header.

Report

DAST Test Configurations
program name: ZAP

version: 2.16.0

generated: Wed, 9 Apr 2025 03:08:03

Site General Information
name: https://splix.io
host: splix.io
port: 443
ssl: true

Name	Impact
Content Security Policy (CSP) Header Not Set Risk: Medium (High)	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. **Possible Solution:** Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header. **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/

Cross-Domain Misconfiguration Risk: Medium (Medium)	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server. **Possible Solution:** Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner. **References:** https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy
Missing Anti-clickjacking Header Risk: Medium (Medium)	The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options. **Possible 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. **References:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
Source Code Disclosure - SQL Risk: Medium (Medium)	Application Source Code was disclosed by the web server. - SQL **Possible Solution:** Ensure that application Source Code is not available with alternative extensions, and ensure that source code is not present within other files or data deployed to the web server, or served by the web server. **Ref**

	References: https://www.wsj.com/articles/BL-CIOB-2999
Insufficient Site Isolation Against Spectre Vulnerability Risk: Low (Medium)	Cross-Origin-Resource-Policy header is an opt-in header designed to counter side-channels attacks like Spectre. Resource should be specifically set as shareable amongst different origins. **Possible Solution:** Ensure that the application/web server sets the Cross-Origin-Resource-Policy header appropriately, and that it sets the Cross-Origin-Resource-Policy header to 'same-origin' for all web pages. 'same-site' is considered as less secured and should be avoided. If resources must be shared, set the header to 'cross-origin'. If possible, ensure that the end user uses a standards-compliant and modern web browser that supports the Cross-Origin-Resource-Policy header (https://caniuse.com/mdn-http_headers_cross-origin-resource-policy). **References:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Cross-Origin_Resource_Policy
Permissions Policy Header Not Set Risk: Low (Medium)	Permissions Policy Header is an added layer of security that helps to restrict from unauthorized access or usage of browser/client features by web resources. This policy ensures the user privacy by limiting or specifying the features of the browsers can be used by the web resources. Permissions Policy provides a set of standard HTTP headers that allow website owners to limit which features of browsers can be used by the page such as camera, microphone, location, full screen etc. **Possible Solution:** Ensure that your web server, application server, load balancer, etc. is configured to set the Permissions-Policy header. **References:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Permissions-Policy https://developer.chrome.com/blog/feature-policy/

	https://scotthelme.co.uk/a-new-security-header-feature-policy/ https://w3c.github.io/webappsec-feature-policy/ https://www.smashingmagazine.com/2018/12/feature-policy/
Strict-Transport-Security Header Not Set Risk: Low (High)	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. **Possible Solution:** Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security. **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
X-Content-Type-Options Header Missing Risk: Low (Medium)	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. **Possible 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.

	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
Information Disclosure - Suspicious Comments	The response appears to contain suspicious comments which may help an attacker.
Modern Web Application	The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.
Non-Storable Content Risk: Informational (Medium)	The response contents are not storable by caching components such as proxy servers. If the response does not contain sensitive, personal or user-specific information, it may benefit from being stored and cached, to improve performance. **Possible Solution:** The content may be marked as storable by ensuring that the following conditions are satisfied: The request method must be understood by the cache and defined as being cacheable ("GET", "HEAD", and "POST" are currently defined as cacheable) The response status code must be understood by the cache (one of the 1XX, 2XX, 3XX, 4XX, or 5XX response classes are generally understood) The "no-store" cache directive must not appear in the request or response header fields For caching by "shared" caches such as "proxy" caches, the "private" response directive must not appear in the response For caching by "shared" caches such as "proxy" caches, the "Authorization" header field must not appear in the request, unless the response explicitly allows it (using one of the "must-revalidate", "public", or "s-maxage" Cache-Control response directives) In addition to the conditions above, at least one of the following conditions must also be satisfied by the response: It must contain an "Expires" header field

	It must contain a "max-age" response directive For "shared" caches such as "proxy" caches, it must contain a "s-maxage" response directive It must contain a "Cache Control Extension" that allows it to be cached It must have a status code that is defined as cacheable by default (200, 203, 204, 206, 300, 301, 404, 405, 410, 414, 501). **References:** https://datatracker.ietf.org/doc/html/rfc7234 https://datatracker.ietf.org/doc/html/rfc7231 https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html
Re-examine Cache-control Directives Risk: Informational (Low)	The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached. **Possible Solution:** For secure content, ensure the cache-control HTTP header is set with "no-cache, no-store, must-revalidate". If an asset should be cached consider setting the directives "public, max-age, immutable". **References:** https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control https://grayduck.mn/2021/09/13/cache-control-recommendations/
	The content was retrieved from a shared cache. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment. This is primarily an issue where caching servers such as "proxy" caches are configured on the local network. This configuration is

Retrieved from Cache Risk: Informational (Medium)	configured on the local network. This configuration is typically found in corporate or educational environments, for instance. **Possible Solution:** Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user: Cache-Control: no-cache, no-store, must-revalidate, private Pragma: no-cache Expires: 0 This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request. **References:** https://tools.ietf.org/html/rfc7234 https://tools.ietf.org/html/rfc7231 https://www.rfc-editor.org/rfc/rfc9110.html
Storable and Cacheable Content Risk: Informational (Medium)	The response contents are storable by caching components such as proxy servers, and may be retrieved directly from the cache, rather than from the origin server by the caching servers, in response to similar requests from other users. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment. This is primarily an issue where "shared" caching servers such as "proxy" caches are configured on the local network. This configuration is typically found in corporate or educational environments, for instance. **Possible Solution:** Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user:

(Medium)	Cache-Control: no-cache, no-store, must-revalidate, private Pragma: no-cache Expires: 0 This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request. **References:** https://datatracker.ietf.org/doc/html/rfc7234 https://datatracker.ietf.org/doc/html/rfc7231 https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html
Storable but Non-Cacheable Content	The response contents are storable by caching components such as proxy servers, but will not be retrieved directly from the cache, without validating the request upstream, in response to similar requests from other users.