Security Scan Report

Target: https://example.com
Scan Type: FULL
Generated: Wed Dec 17 17:50:54 IST 2025

Executive Summary

High: 0
Medium: 2
Low: 3
Informational: 2

Detailed Findings

Content Security Policy (CSP) Header Not Set

Affected URLs:

https://example.com/robots.txt
https://example.com/sitemap.xml
https://example.com

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.

Recommended Fix:

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

Affected URLs:

https://example.com/robots.txt
https://example.com/sitemap.xml
https://example.com

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.

Recommended Fix:

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

HTTPS Content Available via HTTP

Affected URLs:

https://example.com

Description:

Content which was initially accessed via HTTPS (i.e.: using SSL/TLS encryption) is also accessible via HTTP (without encryption).

Recommended Fix:

Ensure that your web server, application server, load balancer, etc. is configured to only serve such content via HTTPS. Consider implementing HTTP Strict Transport Security.

Affected URLs:

https://example.com

Description:

Content which was initially accessed via HTTPS (i.e.: using SSL/TLS encryption) is also accessible via HTTP (without encryption).

Recommended Fix:

Ensure that your web server, application server, load balancer, etc. is configured to only serve such content via HTTPS. Consider implementing HTTP Strict Transport Security.

Missing Anti-clickjacking Header

Affected URLs:

https://example.com

Description:

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

Recommended Fix:

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.

Affected URLs:

https://example.com

Description:

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

Recommended Fix:

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.

Strict-Transport-Security Header Not Set

Affected URLs:

https://example.com/sitemap.xml
https://example.com
https://example.com/robots.txt

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.

Recommended Fix:

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

Affected URLs:

https://example.com/sitemap.xml
https://example.com
https://example.com/robots.txt

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.

Recommended Fix:

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

Re-examine Cache-control Directives

Affected URLs:

https://example.com

Description:

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.

Recommended Fix:

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".

Affected URLs:

https://example.com

Description:

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.

Recommended Fix:

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".

Retrieved from Cache

Affected URLs:

https://example.com/sitemap.xml
https://example.com
https://example.com/robots.txt

Description:

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 typically found in corporate or educational environments, for instance.

Recommended Fix:

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.

Affected URLs:

https://example.com/sitemap.xml
https://example.com
https://example.com/robots.txt

Description:

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 typically found in corporate or educational environments, for instance.

Recommended Fix:

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.

X-Content-Type-Options Header Missing

Affected URLs:

https://example.com

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.

Recommended Fix:

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.

Affected URLs:

https://example.com

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.

Recommended Fix:

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.
