

Security Headers

- [Table of Contents: Security headers](#)

Table of Contents: Security headers

- [1. Access-Control-Allow-Credentials](#)
- [2. Access-Control-Allow-Headers](#)
- [3. Access-Control-Allow-Methods](#)
- [4. Access-Control-Allow-Origin](#)
- [5. Access-Control-Expose-Headers](#)
- [6. Access-Control-Max-Age](#)
- [7. Clear-Site-Data](#)
- [8. Content-Security-Policy](#)
- [9. Cross-Origin-Embedder-Policy](#)
- [10. Cross-Origin-Opener-Policy](#)
- [11. Cross-Origin-Resource-Policy](#)
- [12. Permissions-Policy](#)
- [13. Referrer-Policy](#)
- [14. Strict-Transport-Security \(HSTS\)](#)
- [15. X-Frame-Options](#)
- [16. X-Permitted-Cross-Domain-Policies](#)
- [17. X-Content-Type-Options](#)

1. Access-Control-Allow-Credentials

- **Impact:** Allows servers to include credentials (cookies, HTTP authentication) in cross-origin requests. Misconfiguring this header can expose sensitive data to unauthorized domains.
- **Remediation:** Set `Access-Control-Allow-Credentials` to `true` only when necessary and ensure `Access-Control-Allow-Origin` is set to specific trusted domains.
- **Installation:**
 - **Linux:**

- **Apache:** Add `Header set Access-Control-Allow-Credentials "true"` in the virtual host or `.htaccess`.
- **Nginx:** Use `add_header Access-Control-Allow-Credentials "true";` in the server block.
- **Windows:**
 - **IIS:** Go to HTTP Response Headers and add `Access-Control-Allow-Credentials` with the value `true`.

2. Access-Control-Allow-Headers

- **Impact:** Specifies which HTTP headers can be used in cross-origin requests. Allowing too many headers can increase the risk of sensitive data exposure.
- **Remediation:** Limit the headers to those that are necessary for your application, e.g., `Access-Control-Allow-Headers: Content-Type, Authorization`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Access-Control-Allow-Headers "Content-Type, Authorization"`.
 - **Nginx:** Use `add_header Access-Control-Allow-Headers "Content-Type, Authorization";`.
 - **Windows:**
 - **IIS:** Under HTTP Response Headers, add `Access-Control-Allow-Headers` with necessary headers.

3. Access-Control-Allow-Methods

- **Impact:** Specifies the HTTP methods allowed in cross-origin requests. Allowing unnecessary methods may expose the server to security risks.
- **Remediation:** Restrict this header to necessary methods, e.g., `Access-Control-Allow-Methods: GET, POST, OPTIONS`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"`.
 - **Nginx:** Use `add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";`.
 - **Windows:**
 - **IIS:** In the HTTP Response Headers, add `Access-Control-Allow-Methods`.

4. Access-Control-Allow-Origin

- **Impact:** Specifies which origins can access the resource. Using `*` can expose sensitive data to any origin.
- **Remediation:** Set this header to specific trusted origins, e.g., `Access-Control-Allow-Origin: https://trusted-site.com`.
- **Installation:**
 - **Linux:**

- **Apache:** Add `Header set Access-Control-Allow-Origin "https://trusted-site.com"`.
- **Nginx:** Use `add_header Access-Control-Allow-Origin "https://trusted-site.com";`.
- **Windows:**
 - **IIS:** Add `Access-Control-Allow-Origin` in HTTP Response Headers with the trusted origin.

5. Access-Control-Expose-Headers

- **Impact:** Indicates which headers can be exposed as part of the response, useful in CORS requests. Over-exposing headers can lead to information leakage.
- **Remediation:** Only expose necessary headers, e.g., `Access-Control-Expose-Headers: Content-Length, X-My-Custom-Header`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Access-Control-Expose-Headers "Content-Length, X-My-Custom-Header"`.
 - **Nginx:** Use `add_header Access-Control-Expose-Headers "Content-Length, X-My-Custom-Header";`.
 - **Windows:**
 - **IIS:** Add `Access-Control-Expose-Headers` under HTTP Response Headers.

6. Access-Control-Max-Age

- **Impact:** Specifies how long the results of a preflight request can be cached. If set too high, it may cache outdated policies.
- **Remediation:** Set to a reasonable value in seconds, e.g., `Access-Control-Max-Age: 3600` (1 hour).
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Access-Control-Max-Age "3600"`.
 - **Nginx:** Use `add_header Access-Control-Max-Age "3600";`.
 - **Windows:**
 - **IIS:** Add `Access-Control-Max-Age` in HTTP Response Headers with the value `3600`.

7. Clear-Site-Data

- **Impact:** Clears browsing data (cookies, storage, cache) associated with the website. Useful for logging out or on error pages.
- **Remediation:** Set `Clear-Site-Data` with appropriate directives, e.g., `Clear-Site-Data: "cache", "cookies", "storage"`.
- **Installation:**
 - **Linux:**

- **Apache:** Add `Header set Clear-Site-Data "\"cache\", \"cookies\", \"storage\""`.
- **Nginx:** Use `add_header Clear-Site-Data "\"cache\", \"cookies\", \"storage\"";`.
- **Windows:**
 - **IIS:** Add `Clear-Site-Data` under HTTP Response Headers.

8. Content-Security-Policy

- **Impact:** Prevents XSS, clickjacking, and other code injection attacks by specifying allowed content sources.
- **Remediation:** Define strict rules for content sources, e.g., `Content-Security-Policy: default-src 'self'; script-src 'self';`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Content-Security-Policy "default-src 'self'; script-src 'self';"`.
 - **Nginx:** Use `add_header Content-Security-Policy "default-src 'self'; script-src 'self';"`.
 - **Windows:**
 - **IIS:** Add `Content-Security-Policy` in HTTP Response Headers.

9. Cross-Origin-Embedder-Policy

- **Impact:** Prevents a document from loading cross-origin resources unless explicitly permitted. Protects against resource loading attacks.
- **Remediation:** Set `Cross-Origin-Embedder-Policy: require-corp` to require all resources to have CORS enabled or be same-origin.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Cross-Origin-Embedder-Policy "require-corp"`.
 - **Nginx:** Use `add_header Cross-Origin-Embedder-Policy "require-corp";`.
 - **Windows:**
 - **IIS:** Add `Cross-Origin-Embedder-Policy` in HTTP Response Headers.

10. Cross-Origin-Opener-Policy

- **Impact:** Controls how a document interacts with its opener across origins, reducing risks like cross-origin attacks.
- **Remediation:** Set `Cross-Origin-Opener-Policy: same-origin` to isolate the window from documents from other origins.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Cross-Origin-Opener-Policy "same-origin"`.
 - **Nginx:** Use `add_header Cross-Origin-Opener-Policy "same-origin";`.
 - **Windows:**

- **IIS:** Add `Cross-Origin-Opener-Policy` in HTTP Response Headers.

11. Cross-Origin-Resource-Policy

- **Impact:** Prevents other domains from reading the response of the resources it is set on, protecting against unauthorized data access.
- **Remediation:** Set `Cross-Origin-Resource-Policy: same-origin` for sensitive resources.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Cross-Origin-Resource-Policy "same-origin"`.
 - **Nginx:** Use `add_header Cross-Origin-Resource-Policy "same-origin";`.
 - **Windows:**
 - **IIS:** Add `Cross-Origin-Resource-Policy` in HTTP Response Headers.

12. Permissions-Policy

- **Impact:** Controls which browser features and APIs can be used in the document or embedded iframes, reducing the risk of feature abuse: `contentReference[oaicite:0]{index=0}`.
- **Remediation:** Set `Permissions-Policy` with appropriate directives, e.g., `Permissions-Policy: geolocation=(), microphone=(), camera=():contentReference[oaicite:1]{index=1}`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Permissions-Policy "geolocation=(), microphone=(), camera=()"`.
 - **Nginx:** Use `add_header Permissions-Policy "geolocation=(), microphone=(), camera=()";`.
 - **Windows:**
 - **IIS:** Add `Permissions-Policy` in HTTP Response Headers with the required settings.

13. Referrer-Policy

- **Impact:** Controls the amount of referrer information included in requests, enhancing user privacy and reducing potential information leakage.
- **Remediation:** Use a strict policy, e.g., `Referrer-Policy: strict-origin-when-cross-origin`, to limit the information sent in the `Referer` header.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Referrer-Policy "strict-origin-when-cross-origin"`.
 - **Nginx:** Use `add_header Referrer-Policy "strict-origin-when-cross-origin";`.
 - **Windows:**
 - **IIS:** Set `Referrer-Policy` in HTTP Response Headers.

14. Strict-Transport-Security (HSTS)

- **Impact:** Enforces the use of HTTPS for all future connections to the domain, preventing downgrade attacks and cookie hijacking.
- **Remediation:** Set `Strict-Transport-Security` to enforce HTTPS, e.g., `Strict-Transport-Security: max-age=31536000; includeSubDomains`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set Strict-Transport-Security "max-age=31536000; includeSubDomains"`.
 - **Nginx:** Use `add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";`.
 - **Windows:**
 - **IIS:** Add `Strict-Transport-Security` in HTTP Response Headers.

15. X-Frame-Options

- **Impact:** Prevents clickjacking attacks by controlling whether a browser should be allowed to render a page in a frame or iframe.
- **Remediation:** Set `X-Frame-Options` to `DENY` or `SAMEORIGIN` to control framing, e.g., `X-Frame-Options: DENY`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set X-Frame-Options "DENY"`.
 - **Nginx:** Use `add_header X-Frame-Options "DENY";`.
 - **Windows:**
 - **IIS:** Add `X-Frame-Options` in HTTP Response Headers.

16. X-Permitted-Cross-Domain-Policies

- **Impact:** Controls which cross-domain policies the browser should allow, reducing potential attack surfaces.
- **Remediation:** Set `X-Permitted-Cross-Domain-Policies` to a strict policy, e.g., `X-Permitted-Cross-Domain-Policies: none`.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set X-Permitted-Cross-Domain-Policies "none"`.
 - **Nginx:** Use `add_header X-Permitted-Cross-Domain-Policies "none";`.
 - **Windows:**
 - **IIS:** Add `X-Permitted-Cross-Domain-Policies` in HTTP Response Headers.

17. X-Content-Type-Options

- **Impact:** Prevents the browser from interpreting files as a different MIME type than what is declared by the server. This is essential to prevent MIME type sniffing, which can lead to security vulnerabilities such as Cross-Site Scripting (XSS) attacks:contentReference[oaicite:0]{index=0}.
- **Remediation:** Set `X-Content-Type-Options` to `nosniff` to instruct browsers not to perform MIME type sniffing. This ensures that the browser strictly follows the Content-Type header provided by the server:contentReference[oaicite:1]{index=1}.
- **Installation:**
 - **Linux:**
 - **Apache:** Add `Header set X-Content-Type-Options "nosniff"` in your server configuration or `.htaccess` file.
 - **Nginx:** Use `add_header X-Content-Type-Options "nosniff";` in the server block or location block of your configuration file.
 - **Windows:**
 - **IIS:** In the IIS Manager, go to HTTP Response Headers for the specific site and add `X-Content-Type-Options` with the value `nosniff`.