**What is a Security Header?**

A security header is an HTTP header that helps enhance the security of web applications by controlling how browsers should behave when handling the website’s content.

These headers are exchanged between a client (like a web browser) and a server to define the security of HTTP communication.

**Why is it Used?**

Security headers are used to:

* **Prevent Attacks**: Protect against common web vulnerabilities like Cross-Site Scripting (XSS), Clickjacking, and other attacks.
* **Enforce Security Policies**: Ensure that browsers enforce certain security policies, such as only loading content over HTTPS.
* **Improve Security Posture**: Enhance the overall security of web applications by adding an extra layer of defense.

**Purpose of Security Headers**

The main purpose of security headers is to provide instructions to the browser on how to handle the website’s content securely.

This helps in mitigating various types of attacks and ensuring that the website operates in a secure manner.

**Why You Can’t Set HTTP Headers in HTML**

HTML is a markup language used to structure content on the web. It does not have the capability to set HTTP headers, which are part of the HTTP protocol used to transfer data between the server and the client.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Secure Web Application</title>

<script src="script.js" defer></script>

</head>

<body>

<h1>Hello, secure world!</h1>

</body>

</html>

**How to Set HTTP Headers**

HTTP headers must be set on the server side before the HTML content is sent to the client. Here are some examples of how to set HTTP headers in different server-side languages:

**PHP Example**

**PHP**

<?php

header("Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.cdn.com");

header("X-Frame-Options: DENY");

header("X-Content-Type-Options: nosniff");

header("Strict-Transport-Security: max-age=31536000; includeSubDomains");

header("Referrer-Policy: strict-origin-when-cross-origin");

header("X-XSS-Protection: 1; mode=block");

?>

1. **Content-Security-Policy (CSP)**

**PHP**

header("Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.cdn.com");

**Purpose**: Prevents Cross-Site Scripting (XSS) and other code injection attacks. **What it Prevents**: XSS attacks, data injection attacks. **Explanation**:

* + default-src 'self': Only allow content from the same origin.
  + script-src 'self' https://trusted.cdn.com: Only allow scripts from the same origin and a trusted CDN.

1. **X-Frame-Options**

**PHP**

header("X-Frame-Options: DENY");

**Purpose**: Prevents Clickjacking by controlling whether a browser should be allowed to render a page in a <frame>, <iframe>, <embed>, or <object>. **What it Prevents**: Clickjacking attacks. **Explanation**:

* + DENY: Prevents the page from being displayed in a frame, iframe, or object.

1. **X-Content-Type-Options**

**PHP**

header("X-Content-Type-Options: nosniff");

**Purpose**: Prevents MIME type sniffing, which can transform non-executable MIME types into executable MIME types. **What it Prevents**: MIME type confusion attacks. **Explanation**:

nosniff: Instructs the browser not to guess the MIME type and to follow the declared Content-Type.  
  
  
**MIME: Multipurpose Internet Mail Extensions**

**MIME** stands for **Multipurpose Internet Mail Extensions**. It is a standard that extends the format of email to support text in character sets other than ASCII, as well as attachments of audio, video, images, and application programs. MIME is widely used in web technologies to define the nature and format of a document or file.

**Purpose of MIME**

MIME was originally developed to allow the transmission of multimedia content via email.

However, it has since been adopted by web browsers and servers to handle various types of content on the internet.

**How MIME Works**

MIME types are used to specify the type of data being sent. Each MIME type consists of a type and a subtype, separated by a slash. For example, text/html indicates that the content is HTML text.

**Common MIME Types**

Here are some common MIME types and their uses:

**text/html**: HTML documents

**text/plain**: Plain text

**image/jpeg**: JPEG images

**image/png**: PNG images

**application/json**: JSON data

**application/pdf**: PDF documents

**audio/mpeg**: MP3 audio files

**video/mp4**: MP4 video files

**Example: Setting MIME Types in a Web Server**

**PHP Example**

In a PHP script, you can set the MIME type of the response using the header function:

**PHP**

<?php

// Set the MIME type to HTML

header("Content-Type: text/html");

// Output HTML content

echo "<!DOCTYPE html>

<html lang='en'>

<head>

<meta charset='UTF-8'>

<title>Example Page</title>

</head>

<body>

<h1>Hello, World!</h1>

</body>

</html>";

?>

**Node.js (Express) Example**

In a Node.js application using Express, you can set the MIME type of the response using the res.set method:

**JavaScript**

const express = require('express');

const app = express();

app.get('/', (req, res) => {

// Set the MIME type to HTML

res.set('Content-Type', 'text/html');

// Send HTML content

res.send(`

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example Page</title>

</head>

<body>

<h1>Hello, World!</h1>

</body>

</html>

`);

});

// Start the server

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

1. **Strict-Transport-Security (HSTS)**

**PHP**

header("Strict-Transport-Security: max-age=31536000; includeSubDomains");

**Purpose**: Ensures that the browser only communicates with the server over HTTPS, preventing man-in-the-middle attacks. **What it Prevents**: Man-in-the-Middle (MitM) attacks, protocol downgrade attacks. **Explanation**:

* + max-age=31536000: Enforces HTTPS for one year (in seconds).
  + includeSubDomains: Applies the policy to all subdomains.

1. **Referrer-Policy**

**PHP**

header("Referrer-Policy: strict-origin-when-cross-origin");

**Purpose**: Controls how much referrer information should be included with requests. **What it Prevents**: Information leakage. **Explanation**:

* + strict-origin-when-cross-origin: Sends the origin only when the protocol security level stays the same (HTTPS to HTTPS), but not when it downgrades (HTTPS to HTTP).

1. **X-XSS-Protection**

**PHP**

header("X-XSS-Protection: 1; mode=block");

**Purpose**: Provides a basic protection against reflected XSS attacks. **What it Prevents**: Reflected XSS attacks. **Explanation**:

* + 1; mode=block: Enables XSS filtering and blocks the page if an attack is detected.

**Most frequently used Security headers**

**<IfModule mod\_headers.c>**

Header set X-XSS-Protection "1; mode=block"

Header set X-Frame-Options "SAMEORIGIN"

Header set X-Content-Type-Options "nosniff"

Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" env=HTTPS

Header set Referrer-Policy "same-origin"

Header always set X-Permitted-Cross-Domain-Policies "none"

Header always set Permissions-Policy "camera=(), microphone=(), geolocation=()"

Header add Content-Security-Policy "default-src 'self'; script-src 'self' https://trusted.cdn.com"

Header set Access-Control-Allow-Origin "https://staging.boldsign.com"

Header set Cache-Control "no-store, no-cache, must-revalidate"

Header always edit Set-Cookie ^(.\*)$ $1;HttpOnly;Secure;SameSite=Lax

Header unset X-Powered-By

**</IfModule>**

**Purpose of <IfModule> Tag**

The primary purpose of the <IfModule> tag is to ensure that certain **configuration directives** are only executed if the required module is present.

This helps in creating flexible and portable configuration files that can work across different server environments where certain modules might or might not be

**1. X-XSS-Protection: This header is used to enable the cross-site scripting (XSS) filter built into most browsers.**

1. Header set X-XSS-Protection "1; mode=block"

**Example: If an XSS attack is detected, the browser will prevent rendering of the page.**

1. **X-Frame-Options: This header is used to control whether a browser should be allowed to render a page in a <frame>, <iframe>, <embed>, or <object>.**

Header set X-Frame-Options "SAMEORIGIN"

* **Page A**: https://example.com/pageA
  + **Iframe (Page B)**: https://example.com/pageB
    - **Iframe (Page C)**: https://example.com/pageC

[**Example: The page can only be displayed in a frame on the same origin as the page itself, preventing clickjacking attacks**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options)**.  
  
<IfModule mod\_headers.c>**

**Header set Content-Security-Policy "frame-ancestors 'self' https://sub1.example.com https://sub2.example.com"**

**</IfModule>**

1. **X-Content-Type-Options: This header is used to prevent browsers from MIME-sniffing a response away from the declared content-type.**

Header set X-Content-Type-Options "nosniff"

[**Example: Ensures that browsers follow the MIME types specified in the Content-Type headers2**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options)**.**

1. **Strict-Transport-Security (HSTS): This header tells browsers to only communicate with the site using HTTPS.**

Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" env=HTTPS

**What is HSTS?**

HSTS stands for **HTTP Strict Transport Security**. It’s a security feature that tells web browsers to always use HTTPS (the secure version of HTTP) when communicating with your website.

**Why Use HSTS?**

Using HTTPS ensures that the data sent between the user’s browser and your website is encrypted and secure**. HSTS makes sure that browsers always use this secure connection.**

[**Example: Instructs browsers to remember to only use HTTPS for the next 2 years (63072000 seconds)**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security)**.**

1. **Referrer-Policy: This header controls how much referrer information should be included with requests.**

Header set Referrer-Policy "same-origin"

[**Example: Only sends the referrer information for requests to the same origin**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy)**.  
Example Scenario**

**Let’s say you have a website https://example.com with a** page https://example.com/page1  that contains a link to https://example.com/page2 and another link to https://another-site.com.

1. **Same-Origin Request:**
   * When a user clicks the link to https://example.com/page2, the request is made to the same origin (https://example.com).
   * The Referer header will be sent with the request, containing the URL of the referring page (https://example.com/page1).
2. **Cross-Origin Request:**
   * When a user clicks the link to https://another-site.com, the request is made to a different origin (https://another-site.com).
   * The Referer header will not be sent with the request because the origin is different.
3. **X-Permitted-Cross-Domain-Policies: This header specifies if cross-domain policies are allowed.**

Header always set X-Permitted-Cross-Domain-Policies "none"

[**Example: Disallows all cross-domain policy files**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options)**.**

1. **Permissions-Policy: This header allows or denies the use of browser features in a document or within any <iframe> elements.**

Header always set Permissions-Policy "camera=(), microphone=(), geolocation=()"

[**Example: Disables the use of the camera, microphone, and geolocation features**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Permissions-Policy)**.**

1. **Content-Security-Policy (CSP): This header helps prevent a variety of attacks such as XSS and data injection attacks.**

Header add Content-Security-Policy "default-src 'self'; img-src 'self' https://example.com; script-src 'self' 'unsafe-inline';"

[**Example: Allows resources to be loaded only from the same origin and https://example.com for images, and inline scripts are allowed**](https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP)**.**

1. **Access-Control-Allow-Origin: This header specifies which origins can access the resource.**

Header set Access-Control-Allow-Origin "https://staging.boldsign.com"

[**Example: Only allows requests from https://staging.boldsign.com**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options)**.**

1. **Cache-Control: This header specifies caching policies.**

Header set Cache-Control "no-store, no-cache, must-revalidate"

[**Example: Prevents browsers from caching the response**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control)**.**

1. **Set-Cookie: This header is used to edit cookies to include security attributes.**

Header always edit Set-Cookie ^(.\*)$ $1;HttpOnly;Secure;SameSite=Lax

**Example: Ensures cookies are only sent over HTTPS, are not accessible via JavaScript, and are restricted to the same site.**

1. **X-Powered-By: This header is often used to indicate the technology used by the web server. Unsetting it can help obscure the server’s details.**

Header unset X-Powered-By

**Example: Removes the X-Powered-By header to prevent attackers from gaining information about the server.**

**Basic of HTTP Working:-**

* **Internet:**

**The global network connecting millions of devices.**

* **Web:**

**A service on the internet for accessing linked web pages.**

**Proxy:**

**Good for hiding your IP address and accessing geo-restricted content. Not very secure as it doesn’t encrypt your traffic.**

**VPN:**

**Excellent for securing your internet connection, encrypting your data, and hiding your IP address. Ideal for privacy and security.**

**1. Physical Address (MAC Address)**

* **Description: A unique identifier assigned to a network interface card (NIC) by the manufacturer. It is used for communication within a local network.**
* **Example: 00:1A:2B:3C:4D:5E**
* **Purpose: Used to identify devices on the same local network (e.g., Ethernet or Wi-Fi).**

**2. IP Address**

* **Description: An address assigned to each device connected to a network that uses the Internet Protocol for communication. There are two versions: IPv4 and IPv6.**
  + **IPv4: A 32-bit address written in dotted-decimal format.**
    - **Example: 192.168.1.1**
  + **IPv6: A 128-bit address written in hexadecimal format.**
    - **Example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334**
* **Purpose: Used to identify devices on a network and route traffic between them.**

**3. Port Address**

* **Description: A 16-bit number used to identify specific processes or services on a device.**
* **Example: 80 (HTTP), 443 (HTTPS)**
* **Purpose: Used to direct traffic to the correct application or service on a device.**

**4. Logical Address**

* **Description: An address used by networking software to identify a device on a network, independent of the physical connection.**
* **Example: An IP address like 192.168.1.1**
* **Purpose: Used to route packets across different networks.**

**5. Application-Specific Address**

* **Description: User-friendly addresses designed for specific applications.**
  + **Email Address: Used to identify email recipients.**
    - **Example: example@example.com**
  + **URL (Uniform Resource Locator): Used to locate resources on the web.**
    - **Example: https://www.example.com**
* **Purpose: Used for specific applications like email and web browsing.**

**Summary**

* **Physical Address (MAC): Identifies devices on a local network.**
* **IP Address: Identifies devices on a network and routes traffic.**
* **Port Address: Directs traffic to specific applications or services.**
* **Logical Address: Identifies devices on a network, independent of physical connections.**
* **Application-Specific Address: Used for specific applications like email and web browsing.**

**Difference between the logical and the ip address**

**Logical Address: Can refer to various types of addresses, such as IP addresses, MAC addresses, or even application-specific addresses.**

**IP Address: Specifically refers to addresses used in the Internet Protocol suite.**

**Difference between the logical and the ip address**

* **IP Address: Identifies devices globally and routes data between networks.**
* **MAC Address: Identifies devices within a local network and ensures data delivery within that network.**

**HTTP**

* **HTTP is a**[**protocol**](https://developer.mozilla.org/en-US/docs/Glossary/Protocol)**for fetching resources such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is typically constructed from resources such as text content, layout instructions, images, videos, scripts, and more.**

**Cookies:**

**A cookie is a small text file that a website stores on your computer or mobile device when you visit it. This file contains data that helps the website remember information about your visit, such as your preferences or login status.**

**Uses of Cookies**

1. **Session Management:**
   * **Description: Cookies help manage your session on a website, such as keeping you logged in as you navigate different pages.**
   * **Example: When you log into an online banking site, a cookie keeps you logged in as you move from your account summary to your transaction history.**
2. **Personalization:**
   * **Description: Cookies store your preferences to personalize your experience on a website.**
   * **Example: An e-commerce site might remember your language preference or items in your shopping cart using cookies.**
3. **Tracking and Analytics:**
   * **Description: Cookies help websites track user behavior and gather analytics data.**
   * **Example: A news website might use cookies to track which articles you read, helping them recommend similar content.**
4. **Advertising:**
   * **Description: Cookies are used to deliver targeted ads based on your browsing history.**
   * **Example: If you visit a travel website and search for flights to Paris, you might see ads for hotels in Paris on other websites.**

* **Server-Side: The server sets the cookie and its attributes in the HTTP response headers.**
* **Client-Side: The client (browser) stores and manages the cookie based on the server’s instructions.**

**✅ What is Content Security Policy (CSP)?**

**CSP is a security feature that helps control which resources (scripts, images, styles, etc.) a webpage is allowed to load. It prevents malicious content like cross-site scripting (XSS), and helps reduce data theft and code injection.**

**🔸 What does default-src mean in CSP?**

**default-src is the fallback rule. If you don’t specify a rule for something (like scripts, images, etc.), then default-src applies.**

**🧠 Think of it like this:**

**"If I forget to say what kind of resources a specific type (like img-src, script-src) can load, use whatever rule I put in default-src."**

**💡 Example:**

**http**

**CopyEdit**

**Content-Security-Policy: default-src 'self';**

* **'self' means: only load resources from your own domain.**
* **So this policy says:**
  + **✅ Load scripts, images, styles, etc., only from your site.**
  + **❌ Block anything from external sites like Google Fonts, YouTube, or CDNs.**

**✨ Detailed Example – With Breakdown:**

**http**

**CopyEdit**

**Content-Security-Policy: default-src 'self'; img-src 'self' https://images.example.com; script-src 'self' https://cdn.example.com;**

**What does this do?**

| **Resource Type** | **Allowed From** |
| --- | --- |
| **Scripts** | **Your domain + cdn.example.com** |
| **Images** | **Your domain + images.example.com** |
| **Everything Else (e.g., CSS, if not specified)** | **Only your domain ('self')** |

**❓ Will CSP stop other websites from using my content?**

**➡️ No. CSP only controls what your site loads in, not what other sites load from your site.**

**✅ To block others from embedding or loading your content, you should use:**

* **X-Frame-Options or frame-ancestors in CSP (for preventing embedding).**
* **Access-Control-Allow-Origin (CORS) headers (for blocking cross-site AJAX/image requests).**
* **robots.txt or meta tags (to block scraping/indexing).**

**🔒 Example of preventing content theft:**

**To stop other websites from embedding your site in an <iframe>:**

**http**

**CopyEdit**

**Content-Security-Policy: frame-ancestors 'self';**

**Or:**

**http**

**CopyEdit**

**X-Frame-Options: SAMEORIGIN**

**This ensures only your site can embed your own pages, blocking others from using your content in an iframe.**

**✅ Summary**

| **Header** | **What it Does** |
| --- | --- |
| **default-src** | **Fallback rule for all resource types** |
| **script-src, img-src, etc.** | **Specific rules for each resource** |
| **frame-ancestors or X-Frame-Options** | **Controls who can embed your page** |
| **Access-Control-Allow-Origin** | **Controls which origins can fetch your content (CORS)** |

**If you'd like, I can help you write a proper CSP policy for your WordPress site. Just let me know what types of external resources you're using (e.g., fonts, images, YouTube, etc.).**

**CORS  
Great! Let's break this down into simple and clear terms. You're asking about:**

* **Access-Control-Allow-Origin**
* **What it does**
* **What types of requests it affects (like API vs. media)**
* **Other CORS headers (and how they work)**

**✅ What is Access-Control-Allow-Origin?**

**It's a security header that tells the browser:**

**"Which websites are allowed to access resources (like APIs or data) from this server?"**

**This is part of a system called CORS (Cross-Origin Resource Sharing).**

**📌 Why is it needed?**

**By default, browsers block cross-origin requests (e.g. from staging.boldsign.com to boldsign.com) for security.**

**If a site wants to allow other origins (domains) to access its data or APIs, it uses the Access-Control-Allow-Origin header.**

**🧪 Example: Simple API Request**

**Without CORS:**

**js**

**CopyEdit**

**// From https://myfrontend.com**

**fetch("https://boldsign.com/api/data")**

**// ❌ Will fail in the browser due to CORS**

**With CORS:**

**If boldsign.com adds this response header:**

**arduino**

**CopyEdit**

**Access-Control-Allow-Origin: https://myfrontend.com**

**Then:**

**js**

**CopyEdit**

**fetch("https://boldsign.com/api/data")**

**// ✅ Works because the origin is allowed**

**🔍 Example You Mentioned:**

**http**

**CopyEdit**

**Access-Control-Allow-Origin: https://boldsign.com, https://y.clarity.ms/**

**This looks like it's trying to allow two domains, but technically:**

* **The Access-Control-Allow-Origin header only allows ONE origin at a time (not a comma-separated list).**
* **So this won't work unless the server sets it dynamically depending on the request.**

**What it *should* do:**

**If https://boldsign.com receives a request from https://y.clarity.ms, it checks the origin, and responds with:**

**arduino**

**CopyEdit**

**Access-Control-Allow-Origin: https://y.clarity.ms**

**🧠 What Does It Affect?**

**It applies to:**

* **✅ API requests (e.g., fetch(), XMLHttpRequest)**
* **✅ Web fonts**
* **✅ Media (images, videos) — *only if you try to read them from JS***
* **✅ AJAX requests across domains**
* **❌ It does not block simple linking (like <img src="..."> or <a href="...">)**

**🔐 Example with Image:**

**html**

**CopyEdit**

**<img src="https://boldsign.com/images/logo.png">**

**This works even without CORS because you are just displaying it, not reading it with JS.**

**But this:**

**js**

**CopyEdit**

**const img = new Image();**

**img.src = "https://boldsign.com/images/logo.png";**

**img.onload = function() {**

**const canvas = document.createElement("canvas");**

**const ctx = canvas.getContext("2d");**

**ctx.drawImage(img, 0, 0); // ❌ BLOCKED unless CORS allowed**

**}**

**This needs CORS, because you are reading/manipulating the image.**

**🛡 Other Important CORS Headers**

**1. Access-Control-Allow-Methods**

**Specifies what HTTP methods are allowed:**

**pgsql**

**CopyEdit**

**Access-Control-Allow-Methods: GET, POST, PUT, DELETE**

**2. Access-Control-Allow-Headers**

**Specifies what custom headers the request can include:**

**pgsql**

**CopyEdit**

**Access-Control-Allow-Headers: Content-Type, Authorization**

**3. Access-Control-Allow-Credentials**

**Allows sending cookies with cross-origin requests:**

**yaml**

**CopyEdit**

**Access-Control-Allow-Credentials: true**

**Your fetch must also set:**

**js**

**CopyEdit**

**fetch(url, { credentials: "include" });**

**4. Access-Control-Expose-Headers**

**Lets the browser see certain response headers:**

**mathematica**

**CopyEdit**

**Access-Control-Expose-Headers: X-Custom-Header**

**5. Access-Control-Max-Age**

**How long the browser should cache the preflight (OPTIONS) request:**

**mathematica**

**CopyEdit**

**Access-Control-Max-Age: 600**

**🚫 CORS Errors in Dev**

**If you're testing on staging.boldsign.com and calling app.boldsign.com, and CORS is not properly set, you'll get:**

**pgsql**

**CopyEdit**

**Access to XMLHttpRequest at '...' from origin 'staging.boldsign.com' has been blocked by CORS policy**

**To fix it:**

* **The server at app.boldsign.com must respond with:**

**arduino**

**CopyEdit**

**Access-Control-Allow-Origin: https://staging.boldsign.com**

**✅ Summary**

| **Header** | **Purpose** |
| --- | --- |
| **Access-Control-Allow-Origin** | **Allows specific domain(s) to access resources** |
| **Access-Control-Allow-Methods** | **Allowed HTTP methods** |
| **Access-Control-Allow-Headers** | **Allowed request headers** |
| **Access-Control-Allow-Credentials** | **Allows cookies/auth in cross-origin** |
| **Access-Control-Expose-Headers** | **Exposes custom headers in response** |
| **Access-Control-Max-Age** | **Caches the CORS preflight result** |

**Let me know if you want a live example or how to apply this in your WordPress/Elementor project!**

**Reffer Policy  
✅ What is "Referrer"?**

**When you click a link from one website to another, your browser sends the URL of the current page (called the referrer) to the destination website.**

**This referrer info is sent in the HTTP headers under the name Referer (misspelled in the spec).**

**🧾 What is a "Full URL"?**

**Let’s say you are on this page:**

**arduino**

**CopyEdit**

**https://yoursite.com/page1?user=123**

**This is the full URL:**

* **Protocol: https**
* **Domain: yoursite.com**
* **Path: /page1**
* **Query string: ?user=123**

**So the full referrer would be:**

**arduino**

**CopyEdit**

**https://yoursite.com/page1?user=123**

**📤 Where is this URL sent?**

**It is sent in the HTTP request headers when your browser makes a request to another page after you click a link.**

**Example:**

**You click a link on your site that takes you to:**

**arduino**

**CopyEdit**

**https://othersite.com/blog**

**The browser sends this request:**

**bash**

**CopyEdit**

**GET /blog HTTP/1.1**

**Host: othersite.com**

**Referer: https://yoursite.com/page1?user=123**

**➡️ That last line is the referrer header – it's how the new site (othersite.com) knows where the visitor came from.**

**❓Why does this happen?**

**This helps website owners know:**

* **Who is referring traffic to them**
* **What pages users came from**

**📊 Example: Google Analytics or ad systems use this to track source traffic.**

**🔐 But This Can Be a Problem**

* **Your URL might contain private or sensitive data, like:**

**arduino**

**CopyEdit**

**https://yourbank.com/statement?account=98765**

**If you don’t control how referrer data is shared, another site might receive that entire URL, including the sensitive data. This is why the Referrer-Policy header matters.**

**🛡 So in simple words:**

**“Full URL sent” means:**

**Your entire current page address is passed to the next site when you click a link.**

**“URL is sent in headers” means:**

**It's hidden from users, but visible in the technical request that the browser sends behind the scenes.**

**🧠 Example in Real Time (in DevTools):**

**You can see this yourself:**

1. **Open DevTools (F12) in Chrome**
2. **Go to the "Network" tab**
3. **Click a link to another website**
4. **Click the request, and look at Headers → Request Headers**
5. **You’ll see a line like:**

**arduino**

**CopyEdit**

**Referer: https://yoursite.com/page1**