

HTTP Headers

We have seen examples of HTTP requests and response headers in the previous section. Such HTTP headers pass information between the client and the server. Some headers are only used with either requests or responses, while some other general headers are common to both.

Headers can have one or multiple values, appended after the header name and separated by a colon. We can divide headers into the following categories:

1. General Headers
2. Entity Headers
3. Request Headers
4. Response Headers
5. Security Headers

Let's discuss each of these categories.

General Headers

General headers are used in both HTTP requests and responses. They are contextual and are used to **describe the message rather than its contents**.

Header	Example	Description
Date	Date: Wed, 10 Feb 2022 10:38:44 GMT	Holds the date and time at which the message originated. It's preferred to convert the time to the standard UTC time zone.
Connection	Connection: close	Dictates if the current network connection should stay alive after the request finishes. Two commonly used values for this header are close and keep-alive . The close value from either the client or server means that they would like to terminate the connection, while the keep-alive header indicates that the connection should remain open to receive more data and input.

Entity Headers

Similar to general headers, **Entity Headers** can be **common to both the request and response**. These headers are used to **describe the content** (entity) transferred by a message. They are usually found in responses and POST or PUT requests.

Header	Example	Description
Content-Type	Content-Type: text/html	Used to describe the type of resource being transferred. The value is automatically added by the browsers on the client-side and returned in the server response. The charset field denotes the encoding standard, such as UTF-8 .
Media-Type	Media-Type: application/pdf	The media-type is similar to Content-Type , and describes the data being transferred. This header can play a crucial role in making the server interpret our input. The charset field may also be used with this header.
Boundary	boundary="b4e4fbd93540"	Acts as a marker to separate content when there is more than one in the same message. For example, within a form data, this boundary gets used as --b4e4fbd93540 to separate different parts of the form.
Content-Length	Content-Length: 385	Holds the size of the entity being passed. This header is necessary as the server uses it to read data from the message body, and is automatically generated by the browser and tools like cURL.
Content-Encoding	Content-Encoding: gzip	Data can undergo multiple transformations before being passed. For example, large amounts of data can be compressed to reduce the message size. The type of encoding being used should be specified using the Content-Encoding header.

Request Headers

The client sends **Request Headers** in an HTTP transaction. These headers are **used in an HTTP request and do not relate to the content** of the message. The following headers are commonly seen in HTTP requests.

Header	Example	Description
Host	Host: www.inlanefreight.com	Used to specify the host being queried for the resource. This can be a domain name or an IP address. HTTP servers can be configured to host different websites, which are revealed based on the hostname. This makes the host header an important enumeration target, as it can indicate the existence of other hosts on the target server.
User-Agent	User-Agent: curl/7.77.0	The User-Agent header is used to describe the client requesting resources. This header can reveal a lot about the client, such as the browser, its version, and the operating system.
Referer	Referer: http://www.inlanefreight.com/	Denotes where the current request is coming from. For example, clicking a link from Google search results would make https://google.com the referer. Trusting this header can be dangerous as it can be easily manipulated, leading to unintended consequences.
Accept	Accept: */*	The Accept header describes which media types the client can understand. It can contain multiple media types separated by commas. The */* value signifies that all media types are accepted.
Cookie	Cookie: PHPSESSID=b4e4fbd93540	Contains cookie-value pairs in the format name=value . A cookie is a piece of data stored on the client-side and on the server, which acts as an identifier. These are passed to the server per request, thus maintaining the client's access. Cookies can also serve other purposes, such as saving user preferences or session tracking. There can be multiple cookies in a single header separated by a semi-colon.
Authorization	Authorization: BASIC cGZz3dvcnQK	Another method for the server to identify clients. After successful authentication, the server returns a token unique to the client. Unlike cookies, tokens are stored only on the client-side and retrieved by the server per request. There are multiple types of authentication types based on the webserver and application type used.

A complete list of request headers and their usage can be found [here](#).

Response Headers

[Cheat Sheet](#)[Go to Questions](#)

Table of Contents

HTTP Fundamentals

- [HyperText Transfer Protocol \(HTTP\)](#)
- [Hypertext Transfer Protocol Secure \(HTTPS\)](#)
- [HTTP Requests and Responses](#)
- [HTTP Headers](#)

HTTP Methods

- [HTTP Methods and Codes](#)
- [GET](#)
- [POST](#)
- [CRUD API](#)

My Workstation

OFFLINE

[Start Instance](#)

∞ / 1 spawns left

Response Headers can be used in an HTTP response and do not relate to the content. Certain response headers such as **Age**, **Location**, and **Server** are used to provide more context about the response. The following headers are commonly seen in HTTP responses.

Header	Example	Description
Server	Server: Apache/2.2.14 (Win32)	Contains information about the HTTP server, which processed the request. It can be used to gain information about the server, such as its version, and enumerate it further.
Set-Cookie	Set-Cookie: PHPSESSID=b4e6fbd95540	Contains the cookies needed for client identification. Browsers parse the cookies and store them for future requests. This header follows the same format as the Cookie request header.
WWW-Authenticate	WWW-Authenticate: BASIC realm="localhost"	Notifies the client about the type of authentication required to access the requested resource.

Security Headers

Finally, we have **Security Headers**. With the increase in the variety of browsers and web-based attacks, defining certain headers that enhanced security was necessary. HTTP Security headers are **a class of response headers used to specify certain rules and policies** to be followed by the browser while accessing the website.

Header	Example	Description
Content-Security-Policy	Content-Security-Policy: script-src 'self'	Dictates the website's policy towards externally injected resources. This could be JavaScript code as well as script resources. This header instructs the browser to accept resources only from certain trusted domains, hence preventing attacks such as Cross-site scripting (XSS) .
Strict-Transport-Security	Strict-Transport-Security: max-age=31536000	Prevents the browser from accessing the website over the plaintext HTTP protocol, and forces all communication to be carried over the secure HTTPS protocol. This prevents attackers from sniffing web traffic and accessing protected information such as passwords or other sensitive data.
Referrer-Policy	Referrer-Policy: origin	Dictates whether the browser should include the value specified via the Referer header or not. It can help in avoiding disclosing sensitive URLs and information while browsing the website.

Note: This section only mentions a small subset of commonly seen HTTP headers. There are many other contextual headers that can be used in HTTP communications. It's also possible for applications to define custom headers based on their requirements. A complete list of standard HTTP headers can be found [here](#).

cURL

In the previous section, we saw how using the **-v** flag with cURL shows us the full details of the HTTP request and response. If we were only interested in seeing the response headers, then we can use the **-I** flag to send a **HEAD** request and only display the response headers. Furthermore, we can use the **-i** flag to display both the headers and the response body (e.g. HTML code). The difference between the two is that **-I** sends a **HEAD** request (as will see in the next section), while **-i** sends any request we specify and prints the headers as well.

The following command shows an example output of using the **-I** flag:

```
HTTP Headers

MisaelMacias@htb[/htb]$ curl -I https://www.inlanefreight.com

Host: www.inlanefreight.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/605.1.15 (KHTML, like Gecko)
Cookie: cookie1=2982f09hf012fh2; cookie2=u32t4o3tb3gg4
Accept: text/plain
Referer: https://www.inlanefreight.com/
Authorization: BASIC c6Fzc3dvcnQK

Date: Sun, 06 Aug 2020 08:49:37 GMT
Connection: keep-alive
Content-Length: 26012
Content-Type: text/html; charset=ISO-8859-4
Content-Encoding: gzip
Server: Apache/2.2.14 (Win32)
Set-Cookie: name1=value1,name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
WWW-Authenticate: BASIC realm="localhost"
Content-Security-Policy: script-src 'self'
Strict-Transport-Security: max-age=31536000
Referrer-Policy: origin
```

Exercise: Try to go through all of the above headers, and see whether you can recall the usage for each of them.

In addition to viewing headers, cURL also allows us to set request headers with the **-H** flag, as we will see in a later section. Some headers, like the **User-Agent** or **Cookie** headers, have their own flags. For example, we can use the **-A** to set our **User-Agent**, as follows:

```
HTTP Headers

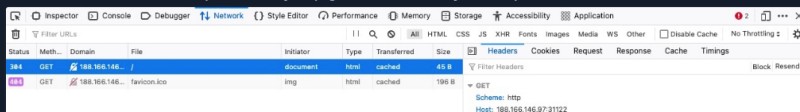
MisaelMacias@htb[/htb]$ curl https://www.inlanefreight.com -A 'Mozilla/5.0'

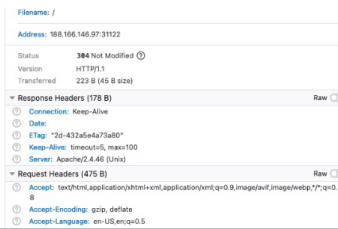
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
...SNIP...
```

Exercise: Try to use the **-I** or the **-v** flags with the above example, to ensure that we did change our User-Agent with the **-A** flag.


Browser DevTools

Finally, let's see how we can preview the HTTP headers using the browser devtools. Just as we did in the previous section, we can go to the **Network** tab to view the different requests made by the page. We can click on any of the requests to view its details:






In the first **Headers** tab, we see both the HTTP request and HTTP response headers. The devtools automatically arrange the headers into sections, but we can click on the **Raw** button to view their details in their raw format. Furthermore, we can check the **Cookies** tab to see any cookies used by the request, as discussed in an upcoming section.

**Connect to Pwnbox**
Your own web-based Parrot Linux Instance to play our labs.

Pwnbox Location

UK 162ms

 Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left


Waiting to start...

☐ Enable step-by-step solutions for all questions

Questions Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): [Click here to spawn the target system!](#)

 The server above loads the flag after the page is loaded. Use the Network tab in the browser devtools to see what requests are made by the page, and find the request to the flag.

HTB{p493_r3qu3\$!\$_m0n1t0r}

Submit Hint

Previous Next Mark Complete & Next

