**Hypertext Transfer Protocol (HTTP)**

Hypertext Transfer Protocol (HTTP) is a protocol for transmitting hypermedia documents such as HTML over Internet. Development of HTTP was initiated by Tim Berners-Lee at CERN in 1989. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access. HTTP is an extensible protocol which has evolved over time, now it is used not only fetch hypertext documents, but also images and videos or to post content to servers, like with HTML form results. HTTP can also be used to fetch parts of documents to update Web pages on demand.



HTTP functions as a request–response protocol in the client–server computing model i.e., when a client submits an HTTP *request* message to the server it waiting until a *response* message is returned to the client. Clients and servers communicate by exchanging individual. The messages sent by the client, usually a Web browser, are called requests and the messages sent by the server as an answer are called responses. Between the client and the server there are numerous entities, collectively called proxies, which perform different operations and act as gateways or caches. HTTP is also a stateless protocol, meaning that the server does not keep any data (state) between two requests.

This extensible nature of HTTP has, over time, allowed for more control and functionality of the Web.

* Caching: Caching is a technique that stores a copy of a given resource and serves it back when requested instead of re-downloading from the originating server which reduce loading speed
* **Relaxing the origin constraint**: To prevent snooping and other privacy invasions, Web browsers enforce strict separation between Web sites. Only pages from the **same origin** can access all the information of a Web page.
* **Authentication**: Some pages may be protected so that only specific users can access them.
* **Sessions**: Using HTTP cookies allows you to link requests with the state of the server. This creates sessions, despite basic HTTP being a state-less protocol.
* **Proxy and tunneling** Servers or clients are often located on intranets and hide their true IP address from other computers. HTTP requests then go through proxies to cross this network barrier.

**Components of HTTP-based systems**

**User-agent**: The user-agent is any tool that acts on behalf of the user. This role is primarily performed by the Web browser and also used by developers to debug their applications. To display a Web page, the browser sends an original request to fetch the HTML document that represents the page. It then parses this file, making additional requests corresponding to execution scripts, layout information to display.

**Web Server:** On the opposite side of the communication channel is the server, which serves the document as requested by the client. A server appears as only a single machine virtually; but it may actually be a collection of servers sharing the load or a complex piece of software on demand

**Proxies:** Between the Web browser and the server, numerous computers and machines relay the HTTP messages. Due to the layered structure of the Web stack, most of these operate at the transport, network or physical levels, becoming transparent at the HTTP layer. Those operating at the application layers are generally called **proxies.**

Proxies may perform numerous functions like caching, filtering, load balancing, authentication and logging.

**REQUEST – RESPONSE**

**REQUEST MESSAGES**

A client sends *request messages* to the server, which consist of:

* **a request line, consisting of the case-sensitive request method, a space, the request target, another space, the protocol version, a carriage return, and a line feed.**
* **zero or more request header fields, each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, and optional trailing whitespace and ending with a carriage return and a line feed; an empty line, consisting of a carriage return and a line feed**
* **an optional message body.**

**HTTP defines methods to indicate the desired action to be performed on the identified resource. This resource represents, whether pre-existing data or data that is generated dynamically depends on the implementation of the server.**

* **GET The GET method requests that the target resource transfers a representation of its state. GET requests should only retrieve data and should have no other effect**
* **HEAD The HEAD method requests that the target resource transfers a representation of its state, like for a GET request, but without the representation data enclosed in the response body.**
* **POST The POST method requests that the target resource processes the representation enclosed in the request according to the semantics of the target resource.**
* **PUT The PUT method requests that the target resource creates or updates its state with the state defined by the representation enclosed in the request.**
* **DELETE The DELETE method requests that the target resource deletes its state.**
* **CONNECT The CONNECT method request that the intermediary establishes a TCP/IP tunnel to the origin server identified by the request target. It is often used to secure.**
* **OPTIONS The OPTIONS method requests that the target resource transfers the HTTP methods that it supports.**
* **TRACE The TRACE method requests that the target resource transfers the received request in the response body.**
* **PATCH The PATCH method requests that the target resource modifies its state according to the partial update defined in the representation enclosed in the request.**

**HTTP HEADER FIELDS**

**HTTP header fields are a list of linefeed-separated HTTP data being sent and received by both the client program and server on every HTTP request. These headers are usually invisible to the end-user and are only visible to the backend programs and people maintaining the internet system. They define how information sent/received through the connection are encoded, the session verification and identification of the client or their anonymity thereof, how the server should handle data, the age of the document being downloaded, amongst others.**

**RESPONSE MESSAGES**

**The server sends response messages to the client, which consist of:**

* **a status line, consisting of the protocol version, a space, the response status code, another space, a possibly empty reason phrase, a carriage return, and a line feed.**
* **zero or more response header fields, each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, and optional trailing whitespace and ending with a carriage return and a line feed;**
* **an empty line, consisting of a carriage return and a line feed.**
* **an optional message body.**

**The first line of the HTTP response is called the status line and includes a numeric status code and a textual reason phrase. The response status code is a three-digit integer code representing the result of the server's attempt to understand and satisfy the client's corresponding request.**

**The first digit of the status code defines its class:**

**1XX (informational) : The request was received, continuing process.**

**2XX (successful) : The request was successfully received, understood, and accepted.**

**3XX (redirection) : Further action needs to be taken in order to complete the request.**

**4XX (client error) : The request contains bad syntax or cannot be fulfilled.**

**5XX (server error) : The server failed to fulfill an apparently valid request.**

**Security vulnerabilities in HTTP**

**HTTP header injection**

HTTP header injection is a general class of web application security vulnerability which occurs when Hypertext Transfer Protocol (HTTP) headers are dynamically generated based on user input. Header injection in HTTP responses can allow for HTTP response splitting, Session fixation via the Set-Cookie header, cross-site scripting, and malicious redirect attacks via the location header.

**HTTP request smuggling**

HTTP request smuggling is a security exploit on the HTTP protocol that uses inconsistency between the interpretations of Content-length and/or Transfer-encoding headers between HTTP server implementations in an HTTP proxy server chain and is a relatively new area for web-based attacks

**HTTP response splitting**

HTTP response splitting is a form of web application vulnerability, resulting from the failure of the application or its environment to properly sanitize input values. It can be used to perform cross-site scripting attacks, cross-user defacement, web cache poisoning, and similar exploits, which allows the attacker to set arbitrary headers, take control of the body, or break the response into two or more separate responses

**HTTP Parameter Pollution**

HTTP Parameter Pollution is a web application vulnerability exploited by injecting encoded query string delimiters in already existing parameters. The vulnerability occurs if user input is not sanitized correctly by a web application, which can cause simple annoyance to complete disruption of the intended behavior of a web application.

**HTTP response status codes**

100 Continue

101 Switching Protocols

103 Early Hints

200 OK

201 Created

202 Accepted

203 Non-Authoritative Information

204 No Content

205 Reset Content

206 Partial Content

300 Multiple Choices

301 Moved Permanently

302 Found

303 See Other

304 Not Modified

307 Temporary Redirect

308 Permanent Redirect

400 Bad Request

401 Unauthorized

402 Payment Required

403 Forbidden

404 Not Found

405 Method Not Allowed

406 Not Acceptable

407 Proxy Authentication Required

408 Request Timeout

409 Conflict

410 Gone

411 Length Required

412 Precondition Failed

413 Payload Too Large

414 URI Too Long

415 Unsupported Media Type

416 Range Not Satisfiable

417 Expectation Failed

418 I'm a teapot

422 Unprocessable Entity

425 Too Early

426 Upgrade Required

428 Precondition Required

429 Too Many Requests

431 Request Header Fields Too Large

451 Unavailable For Legal Reasons

500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

505 HTTP Version Not Supported

506 Variant Also Negotiates

507 Insufficient Storage

508 Loop Detected

510 Not Extended

511 Network Authentication Required