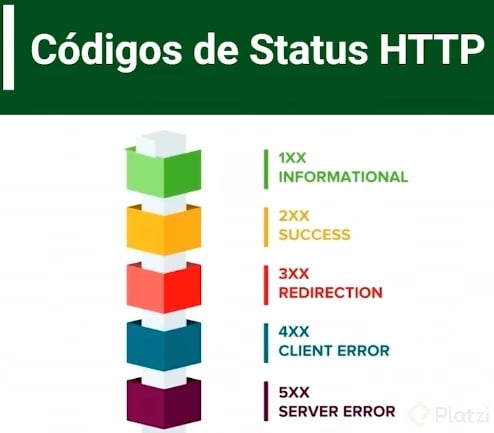
- El método de solicitud HTTP GET se utiliza para solicitar un recurso del servidor. La solicitud GET solo debe recibir datos (el servidor no debe cambiar su estado). - El método de solicitud POST solicita que un servidor web acepte los datos incluidos en el cuerpo del mensaje de solicitud, muy probablemente para almacenarlos. A menudo se usa al cargar un archivo o al enviar un formulario web completo. - HEAD es un método de solicitud que solicita una respuesta idéntica a la de una solicitud GET, pero sin el cuerpo de la respuesta. Esto es útil para recuperar metainformación escrita en encabezados de respuesta, sin tener que transportar todo el contenido. - La operación HTTP PUT se usa para un reemplazo completo de un recurso OSLC. El método PUT actualiza las propiedades literales y las propiedades de los recursos locales y elimina las propiedades de los recursos locales que no están incluidas en la solicitud. - El método HTTP DELETE se utiliza para eliminar un recurso del servidor. A diferencia de las solicitudes GET y HEAD, las solicitudes DELETE pueden cambiar el estado del servidor. Enviar un cuerpo de mensaje en una solicitud DELETE puede hacer que algunos servidores rechacen la solicitud. Pero aún puede enviar datos al servidor utilizando parámetros de URL.

Métodos HTTP

* GET : Permite hacer solicitud al servidor
* HEAD : Permite retornar las cabeceras del servidor
* POST : Permite enviar datos
* PUT : Permite crear archivos en el servidor
* DELETE : Permite Borrar en el servidor

Cada tipo de solicitud tiene una respuesta



HTTP (HyperText Transfer Protocol) define una serie de métodos y códigos de estado que se utilizan en las comunicaciones entre clientes (generalmente navegadores web) y servidores. A continuación, se describen los métodos más comunes y los códigos de estado.

**Métodos HTTP Comunes**

1. **GET**:
   * Solicita una representación de un recurso específico. Las peticiones GET deben solo recuperar datos.
   * Ejemplo: GET /index.html HTTP/1.1
2. **POST**:
   * Se utiliza para enviar datos al servidor para crear o actualizar un recurso.
   * Ejemplo: POST /submit-form HTTP/1.1
3. **PUT**:
   * Reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la solicitud.
   * Ejemplo: PUT /user/123 HTTP/1.1
4. **DELETE**:
   * Elimina el recurso especificado.
   * Ejemplo: DELETE /user/123 HTTP/1.1
5. **HEAD**:
   * Solicita una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
   * Ejemplo: HEAD /index.html HTTP/1.1
6. **OPTIONS**:
   * Describe las opciones de comunicación para el recurso de destino.
   * Ejemplo: OPTIONS /index.html HTTP/1.1
7. **PATCH**:
   * Se utiliza para aplicar modificaciones parciales a un recurso.
   * Ejemplo: PATCH /user/123 HTTP/1.1
8. **CONNECT**:
   * Establece un túnel hacia el servidor identificado por el recurso de destino.
   * Ejemplo: CONNECT www.example.com:443 HTTP/1.1
9. **TRACE**:
   * Realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.
   * Ejemplo: TRACE /index.html HTTP/1.1

**Códigos de Estado HTTP**

Los códigos de estado HTTP se agrupan en cinco clases:

1. **1xx (Informativos)**: Indican que la solicitud fue recibida y el proceso continúa.
   * **100 Continue**: El cliente debe continuar con su solicitud.
   * **101 Switching Protocols**: El servidor acepta cambiar a otro protocolo.
2. **2xx (Éxito)**: Indican que la solicitud fue recibida, entendida y procesada correctamente.
   * **200 OK**: La solicitud ha tenido éxito.
   * **201 Created**: La solicitud ha sido cumplida y ha resultado en la creación de un nuevo recurso.
   * **202 Accepted**: La solicitud ha sido aceptada para procesamiento, pero el procesamiento no se ha completado.
   * **204 No Content**: La solicitud ha sido procesada con éxito, pero no hay contenido que devolver.
3. **3xx (Redirección)**: Indican que el cliente debe tomar acciones adicionales para completar la solicitud.
   * **301 Moved Permanently**: El recurso solicitado ha sido movido permanentemente a una nueva URI.
   * **302 Found**: El recurso solicitado reside temporalmente en una URI diferente.
   * **304 Not Modified**: Indica que el recurso no ha sido modificado desde la última solicitud.
4. **4xx (Errores del Cliente)**: Indican que la solicitud contiene sintaxis incorrecta o no puede ser cumplida.
   * **400 Bad Request**: La solicitud no puede ser procesada debido a una sintaxis incorrecta.
   * **401 Unauthorized**: La solicitud requiere autenticación del usuario.
   * **403 Forbidden**: El servidor entiende la solicitud, pero se niega a autorizarla.
   * **404 Not Found**: El servidor no puede encontrar el recurso solicitado.
   * **405 Method Not Allowed**: El método especificado en la solicitud no está permitido para el recurso.
5. **5xx (Errores del Servidor)**: Indican que el servidor falló en cumplir una solicitud aparentemente válida.
   * **500 Internal Server Error**: El servidor encontró una condición inesperada que le impide cumplir con la solicitud.
   * **501 Not Implemented**: El servidor no reconoce el método de solicitud o no tiene capacidad para cumplirla.
   * **502 Bad Gateway**: El servidor, actuando como una puerta de enlace o proxy, recibió una respuesta no válida del servidor upstream.
   * **503 Service Unavailable**: El servidor no está disponible temporalmente, generalmente debido a mantenimiento o sobrecarga.
   * **504 Gateway Timeout**: El servidor, actuando como una puerta de enlace o proxy, no recibió una respuesta a tiempo del servidor upstream.

**Ejemplo de Solicitud y Respuesta**

**Solicitud GET:**

bash

Copy code

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

**Respuesta 200 OK:**

php

Copy code

HTTP/1.1 200 OK

Date: Mon, 13 Jul 2023 10:00:00 GMT

Server: Apache/2.4.41 (Ubuntu)

Last-Modified: Mon, 13 Jul 2023 09:00:00 GMT

Content-Type: text/html

Content-Length: 137

Connection: keep-alive

<html>

<head><title>Example</title></head>

<body><h1>Hello, World!</h1></body>

</html>

**Conclusión**

Comprender los métodos y códigos de estado HTTP es fundamental para realizar pruebas de seguridad de aplicaciones web, así como para el desarrollo y depuración de aplicaciones web. Si tienes alguna pregunta específica o necesitas más detalles sobre algún aspecto, no dudes en preguntar.