# Walkthrough Tryhackme Free Roadmap

## HTTP in Detail

**Escrito por:**
Ian E. Acosta Sian
Pentester Jr.

# Indice

# 1. ¿Que es HTTP(S)?

## 1.1. ¿Que es HTTP?

HTTP (Hyper Text Transfer Protocol) es un protocolo desarrollado entre 1989 y 1991 por Tim Berners-Lee para la comunicación entre navegadores y servidores web. Permite la transmisión de datos como HTML, imagenes y videos

## 1.2. ¿Que es HTTPS?

HTTPS (Hyper Text Transfer Protocol Secure) es la versión segura de HTTP. Utiliza cifrado para proteger la transmisión de datos y garantizar que la comunicación sea con el servidor legitimo, evitando suplantaciones.

## 1.3. Respuestas

Para resolver el desafio hay que ingresar a la pagina que esta en el mismo desafío y solo dar click en el candado de la URL del dominio de la maquina que se desplegó.

1. What does HTTP stand for?
   **Respuesta:** HyperText Transfer Protocol
2. What does the S in HTTPS stand for?
   **Respuesta:** secure
3. On the mock webpage on the right there is an issue, once you've found it, click on it. What is the challenge flag?
   **Respuesta:** THM{INVALID_HTTP_CERT}

# 2. Requests And Responses

## 2.1. ¿Que es una URL?

Una URL (Uniform Resourse Locator) es una dirección utilizada para acceder a recursos en internet. Sus partes incluyen:

- Esquema: Protocolo a usar (HTTP, HTTPS, FTP).
- Usuario: Credenciales opcionales para autenticación.
- Host: Dominio o IP del servidor.
- Puerto: Generalmente 80 para HTTP y 443 para HTTPS.
- Query String: Parametros adicionales en la solicitud.
- Fragmento: Referencia a una parte especifica de la pagina.

## 2.2. Solicitud HTTP

Es le mensaje enviado por le navegador al servidor. Por ejemplo:

Get /HTTP /1.1
Host: tryhackme.com
User-agent: Mozilla/5.0 Firefox/87.0
Referer: https://tryhackme.com/

- Metodo GET: Solicita la pagina principal.
- Host: Especifica el sitio web deseado.
- User-agent: Indica el navegador usado.
- Referer: Muestra la página de origen.
- Linea en blanco: Finaliza la solicitud.

### 2.3. Respuesta HTTP

Es la respuesta del servidor con la información solicitada. Por ejemplo:

HTTP /1.1 200 ok
Server: nginx/1.15.8
Date: Fri, 09 APR 2021 13:34:03 GMT
Content-type: text/html
Content-Length:98

<html >
<head ><tittle >tryhackme </tittle ></head >
<body>Welcome to tryhackme.com </body>
<html >

- Codigo 200 ok: La solicitud fue exitosa.
- Servidor: Indica el software del servidor.
- Fecha: Hora y fecha de la respuesta.
- Content-type: Tipo de contenido devuelto.
- Content-length: Longitud del contenido.
- HTML: Datos enviados en la respuesta.

### 2.4. Respuestas

1. What HTTP protocol is being used in the above example?
   **Respuesta:** HTTP/1.1
2. What response header tells the browser how much data to expect?
   **Respuesta:** Content-Length

## 3. HTTP Methods

Los métodos HTTP indican la acción que el cliente desea realizar en una solicitud. Los mas comunes son:

- GET: Recupera información de un servidor web.
- POST: Envía datos al servidor para crear nuevos registros.
- PUT: Envía datos al servidor para actualizar información existente.
- DELETE: Elimina información o registros en el servidor.

Los metodos mas utilizados son GET y POST, especialmente en la navegación web y formularios.

### 3.1. Respuestas

1. What method would be used to create a new user account?
   **Respuesta:** POST
2. What method would be used to update your email address?
   **Respuesta:** PUT
3. What method would be used to remove a picture you've uploaded to your account?
   **Respuesta:** DELETE
4. What method would be used to view a news article?
   **Respuesta:** GET

## 4. Códigos de estado HTTP

Los códigos de estado HTTP indican el resultado de una solicitud y se dividen en 5 categorías:

- 100-199 (Informativos) → La solicitud ha sido aceptada y debe continuar.
- 200-299 (Éxito) → La solicitud ha sido aceptada y debe continuar.
- 300-399 (Redirección) → Se requiere otra acción para completar la solicitud.
- 400-499 (Errores del cliente) → Problemas con la solicitud.
- 500-599 (Errores del servidor) → Problemas en el servidor.

### 4.1. Códigos mas comunes

- 200 - ok: La solicitud se completo con exito.
- 201 - Created: Se creó un recurso nuevo (usuario, publicación, ect).
- 301 - Moved Permanently: Redirección permanente a otra URL.
- 302 - Found: Redirección temporal.
- 400 - Bad Request: Error en la solicitud del cliente.
- 401 - Not Authorized: Requiere autenticación para acceder.
- 403 - Forbidden: Accesos denegado, incluso con autenticación.
- 404 - Page Not Found: El recurso solicitado no existe.
- 405 - Method Not Allowed: Metodo HTTP no permitido.
- 500 - Internal Server Error: Error inesperado en el servidor.
- 503 - Service Unavailable: El servidor esta sobrecargado o en mantenimiento.

### 4.2. Respuestas

1. What response code might you receive if you've created a new user or blog post article?
   **Respuesta:** 201
2. What response code might you receive if you've tried to access a page that doesn't exist?
   **Respuesta:** 404
3. What response code might you receive if the web server cannot access its database and the application crashes?
   **Respuesta:** 503
4. What response code might you receive if you try to edit your profile without logging in first?
   **Respuesta:** 401

# 5. Headers

Encabezados de una solicitud (Request Headers)

- Host: Indica el dominio solicitado (útil en servidores con múltiples sitios).
- User-agent: Informa al servidor sobre el navegador y su versión.
- Content-Length: Especifica la cantidad de datos enviados en una solicitud.
- Acept-Enconding: Indica los métodos de compresión soportados por el navegador.
- Cookie: Envía información almacenada en el navegador al servidor.

Encabezados de una respuesta (Response Headers)

- Set-Cookie: Establece cookies en el navegador.
- Cache-Control: Define cuanto tiempo almacenar los datos en caché antes de solicitar una nueva versión.
- Content-Type: Especifica el tipo de contenido enviado (HTML, JSON, imagenes, etc).
- Content-Encoding: Indica el método de compresión usado para transmitir los datos.

## 5.1. Respuestas

- What header tells the web server what browser is being used?
  **Respuesta:** User-Agent
- What header tells the browser what type of data is being returned?
  **Respuesta:** Content-Type
- What header tells the web server which website is being requested?
  **Respuesta:** Host

# 6. Cookies

## 6.1. ¿Que son las cookies?

Son pequeños fragmentos de datos almacenados en nuestras computadoras cuando visitamos un sitio web. Se guardan mediante el encabezado 'Set-Cookie' en la respuesta del servidor y se envía de vuelta en futuras solicitudes.

## 6.2. ¿Para que sirven?

- Autenticación: Mantiene la sesión iniciada sin necesidad de reingresar credenciales.
- Personalización: Guardan preferencias del usuario (tema, idioma, etc).
- Seguimiento: Permiten a los sitios recordar si has estado antes en ellos.

Ejemplo de Solicitudes y Respuestas:

| El cliente solicita la pagina web |
|---|
| GET / HTTP/ 1.1<br>Host: cookies.thm<br>User-agent: xxxx |

| Respuesta del servidor |
|---|
| HTTP / 1.1 200 ok<br>Server: nginx/ 1.15.8<br>Date: xxxx<br>Content-type: Text/html; charser= UTF-8 |

| El cliente envía de nuevo el formulario con el nombre Adam |
|---|
| POST / HTTP 1.1<br>Host: cookies.thm<br>User-agent: xxxx<br>Content-type: application /x-wwwform-urlencode<br>Content-length: 9<br><br>name = Adam |

| El servidor responde con una cabecera Set-cookie guardando name=Adam |
|---|
| HTTP /1.1 200 ok<br>Server: nginx/1.15.8<br>Date: xxxx<br>Set-Cookie: name = Adam<br>Content-type: text/html; charset=UTF-8<br><br>HTML DATA... |

| En las próximas solicitudes el cliente envía los datos cookies al servidor |
|---|
| GET / HTTP / 1.1<br>Host: cookies.thm<br>User-agent: xxxx<br>Cookie: name = Adam |

| El servidor ve las cookies y muestra un mensaje de bienvenida |
|---|
| HTTP /1.1 200 ok<br>Server: nginx/1.15.8<br>Date: xxxx<br>Content-type: text/html; charset=UTF-8<br><br><html><body>Welcome back Adam </body></html> |

### 6.3. Respuestas

1. Which header is used to save cookies to your computer?
   **Respuesta:** Set-cookie

## 7. Making Requests

Primero abrimos la maquina de simulación haciendo click en **View Site**, esto desplegara la maquina para resolver el desafío.
Seleccionamos la opción **GET** y ponemos en la URL de simulación **http://tryhackme.com/room** esto dará el Response de la solicitud y dentro esta el primer flag.
Para el segundo seleccionamos **GET**, la URL **http://tryhackme.com/blog** y en el engranaje agregamos un key llamada **id** y en value **1**, guardamos y damos a Go.
Para el tercero seleccionamos el metodo **DELETE** y ponemos la URL **http://tryhackme.com/user/1**, damos a Go y tenemos la flag en el response.
En el cuarto seleccionamos el metodo **PUT**, URL **http://tryhackme.com/user/2** y en los parametros key **username** y value **admin**.
En el último utilizamos el metodo **POST**, URL **http://tryhackme.com/login** y en los parámetros configuramos 2:

1. key **username** y value **thm**
2. key **password** y value **letmein**

### 7.1. Respuestas

1. THM{YOU'RE_IN_THE_ROOM}

2. THM{YOU_FOUND_THE_BLOG}
3. THM{USER_IS_DELETED}
4. THM{USER_HAS_UPDATED}
5. THM{HTTP_REQUEST_MASTER}