CIENCIA DE LA COMPUTACIÓN

REDES Y COMUNICACIÓN

Guía para principiantes: HTTP, HTTPS y SSL/TLS

Alumno: José David Mamani Vilca

Semestre: VIII

Mayo del 2019

"Los alumnos declaran haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

Guía para principiantes: HTTP, HTTPS y SSL/TLS

HTTP:

HTTP (Protocolo de Transferencia de Hipertexto) es un protocolo para la transmisión de información dentro de la World Wide Web. HTTP define la sintaxis y semántica que se utilizan dentro de la arquitectura para que tanto clientes, servidores, proxis, etc. puedan comunicarse. Es un protocolo sin estado, lo que implica que no guarda información sobre conexiones previas por lo que el desarrollo de aplicaciones bajo esta temática necesita de características que les permitan acceder al estado previo.

HTTP es un protocolo fuertemente enfocado al esquema "*petición-respuesta*" entre cliente servidor. El cliente usuario realiza una petición al enviar un mensaje, bajo un formato determinado, al servidor. El servidor por su lado, envía un mensaje de respuesta describiendo si la operación solicitada es posible o en caso sea necesario retornar errores ya predefinidos. HTTP función además bajo el puerto predefinido 80, siendo necesario una especificación en la URL si se desea utilizar algún otro puerto

Mensajes HTTP:

Los mensajes enviados por HTTP son fácilmente legibles y sencillos de depurar. Poseen una estructura en donde se indican todas las propiedades que el mensaje posee además de la información a transmitir. Sin embargo, esta misma ventaja resulta ser muy inconveniente pues información muy bien descrita suelen pagar dicha claridad con contenido innecesariamente largo y pesado de enviar.

Otro punto a destacar es que en HTTP, se usan dos "*tipos*" de mensajes que en palabras mejor empleadas pueden agruparse en torno a Peticiones (Datos solicitados al destino) y Respuestas (Mensajes con los contenidos deseados.)

Los mensajes HTTTP poseen la siguiente estructura:

- Línea inicial: Contenido dispuesto al inicio del mensaje tanto para Peticiones como para Respuestas.
    - En los mensajes de Petición esta línea contiene la acción solicitada al servidor (métodos de petición que analizaremos más adelante), la URL del recurso solicitado (la ubicación del recurso que queremos dentro del dominio externo) y la versión HTTP que el cliente soporta.
    - En los mensajes de respuesta esta línea solo incluye la versión HTTP y un código de respuesta que indica los resultados de la información solicitada. Existen diversos códigos de respuesta pero los más extendidos son los siguientes:
        - 200 Ok: La solicitud hecha es correcta. Esta es la respuesta estándar para respuestas correctas.
        - 403 Forbidden. La solicitud fue válida pero el servidor se niega a responder.
        - 404 Not Found. El recurso de la solicitud no se ha podido encontrar pero podría estar disponible en el futuro. Se permiten solicitudes subsecuentes por parte del cliente.

        - 503 Service Unavailable. El servidor está actualmente no disponible, ya sea por mantenimiento o por sobrecarga.
        - 505 HTTP Version Not Supported. El servidor no soporta la versión del protocolo HTTP usada en el request.

    Cabe destacar que existen mucho más códigos sin embargo estos no están tan diversificados por el poco uso que se les dan en la práctica.

- Cabeceras del Mensaje: Conformado por una serie de metadatos que otorgan mayor flexibilidad al protocolo. Esta información concluye en la siguiente línea en blanco.
- Cuerpo del Mensaje: Posee una naturaleza opcional por lo que no es obligatorio definirlo. Típicamente contiene los datos que se intercambian entre cliente y servidor.
    - Dependiendo del mensaje podrían: Contener ciertos datos que se necesitan enviar al servidor para que este los procese o contener los datos que el servidor le ha enviado al cliente a modo de respuesta.

Métodos de petición:

Como se mencionó previamente, existen diversos métodos de describen los comportamientos que debe adquirir el objetivo en el destino de nuestro mensaje. Conforme evolucionaba la versión del HTTP (posteriormente analizaremos las diversas versiones HTTP), estos métodos han ido aumentando conforme se deseaba incrementar y añadir mayores funcionalidades al protocolo. Según sea el método que se utilice y los recursos que este involucre, se realizará una tarea con los recursos propios del servidor.

Algunos de los métodos de petición más extendidos son:

- GET: El método GET solicita una representación del recurso especificado. Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto
- HEAD: Pide una respuesta idéntica a la que correspondería a una petición GET, pero en la respuesta no se devuelve el cuerpo.
- POST: Envía los datos para que sean procesados por el recurso identificado. Los datos se incluirán en el cuerpo de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes o ambas cosas.
- PUT: El método PUT permite escribir un archivo en una conexión socket establecida con el servidor. La desventaja del método PUT es que los servidores de alojamiento compartido no lo tienen habilitado.
- DELETE: Borra el recurso especificado.

Versiones HTTP:

HTTP ha evolucionado a lo largo de diversas versiones en donde destaca a primera vista la implementación de nuevos métodos de petición. Estas evoluciones también incluyen cambios en la forma en la que se realizan las conexiones entre servidores y usuarios. Las versiones de HTTP son las siguientes:

- HTTP/1.0 Primera versión HTTP con sólo los métodos GET, HEAD y POST. Muy extendida en servidores proxy.
- HTTP/1.1 Versión más extendida actualmente. Destaca mejoras en la conexión entre cliente y servidor al crear múltiples peticiones para una sola conexión.

- HTTP/2.0 Aparece a inicios del año 2012 como borradores primitivos. Actualmente no posee mucha difusión si bien los planes para este protocolo es sustituir al viejo HTTP/1.1.

Cabeceras:

Dentro de la estructura de los mensajes HTTP existe el concepto de cabeceras. Estos metadatos (datos que describen otros datos) se envían tanto en las peticiones como en las respuestas y su único fin es brindar información esencial sobre los datos que se están intercambiando.

Las cabeceras como aplicación le dan bastante flexibilidad al protocolo pues permiten describir nuevas funcionalidades sin tener que cambiar la raíz del protocolo. Son relativamente sencillas de establecer por lo que conforme evolucionaban las versiones del HTTP se vio factible la creación de nuevas cabeceras.

Entre las principales cabeceras tenemos:

- Cabeceras con referencias a URLS:
  - **Location:** Indica el lugar del contenido.
  - **Referer:** Indica el origen de la petición.
- Cabeceras para el control de cookies:
  - **Set-Cookie, Cookie.**
- Cabeceras que describen el contenido:
  - **Content-type:** MIME del contenido.
  - **Content-Length**: (longitud del mensaje)
  - **Content-Range, Content-Encoding, Content-Language, Content-Location**
- Cabeceras que indicant las capacidades que acepta el emisor del mensaje,.
  - **Accept** (indica el MIME aceptado)
  - **Accept-Charset** (indica el código de caracteres aceptado).
  - **Accept-Encoding** (indica el método de compresión aceptado).
  - **Accept-Language** (indica el idioma aceptado).
  - **Allow** (métodos permitidos para el recurso y que estén en relación con la versión HTTP utilizada.)

Ejemplo de Mensaje HTTP:

Mensaje de Respuesta:

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Título del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
  .
  .
  .
</body>
</html>
```

Mensaje de Solicitud:

```
GET /index.html HTTP/1.1
Host: www.example.com
Referer: www.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Connection: keep-alive
[Línea en blanco]
```

HTTPS:

Es un protocolo de aplicación basado en HTTP con el objetivo de realizar una transferencia segura de datos.

El Protocolo Seguro de Transferencia de Hipertexto utiliza un cifrado SSL/TLS para establecer un canal cifrado para el tráfico de la información sensible dentro del protocolo HTTP. De esta manera la información más sensible del usuario (claves, perfiles, etc) no puede ser utilizada por los atacantes. A diferencia de HTTP, el puerto estándar para este protocolo es el 443 o el 4433.

El protocolo HTTPS puede utilizarse siempre que el servidor destino acepte este tipo de conexiones. Para lograrlo, el administrador debe crear un certificado de clave pública que debe estar firmado por una autoridad de certificación para que el navegador web lo acepte. La adquisición de cualquiera de estos certificados puede ser gratuita o costar una licencia que debe ser pagada periódicamente. Se debe resaltar además, que puede darse el caso que el certificado haya sido vulnerado por lo que su renovación ha de ser inmediata.

Estrictamente hablando HTTPS no es un protocolo independiente, sino que utiliza a HTTP como medio tradicional para la transmisión de la información. De hecho HTTPS opera sobre la capa más alta del modelo TCP/IP (Capa de apliación) por lo que sus mensajes no son más que HTTP encriptados bajo un cifrado SSL/TSL. En su encriptación se incluyen tanto a los headers, como a los datos de peticiones y respuestas. Dado este caso, un atacante solo podría obtener información referente al nombre del dominio y la dirección IP.

SSL/TLS

SSL(Secure Socket Layer) y su sucesor TLS(Transport Layer Security) son protocolos criptográficos que proporcionan comunicaciones seguras en una red.

Para ambos casos se utilizan métodos de criptografía asimétrica con el objetivo de identificar la contraparte con la que se ha establecido la comunicación. Una vez compartidas las llaves se procede a establecer un canal de comunicación con un flujo de datos cifrado entre ambas partes. De esta manera se garantiza la confidencialidad de los mensajes enviados.

Durante su aplicación es común que solo el servidor necesite autentificarse (de ahí el uso de certificados). Para el establecimiento del canal de comunicación han de seguirse algunos pasos previos:

- Negociar el algoritmo que ha de utilizarse durante la conexión (RSA, Diffie Hellman, DSA, etc…)
- Intercambiar las claves públicas y realizar la autentificación apoyándose en certificados.
- Cifrar el tráfico con cifrados simétricos.

Al iniciar una comunicación se empieza con un protocolo de handshaking. El cliente recibe varias estructuras handshake (o de saludo):

- Envía un mensaje *clienthello* que incluye una lista de conjuntos cifrados, métodos de compresión y la versión más alta que el SSL le permita.
- Después se recibe un *serverhello* como respuesta del servidor en donde se elige los parámetros de conexión que el cliente le ha ofrecido en la conexión previa.
- Cuando los parámetros iniciales son establecidos, cliente y servidor intercambian certificados.
- Posteriormente se negocia una clave secrete simétrica denominada como *master secret* posiblemente tras haber efectuado un Diffie Hellman. Todos los datos posteriores a esto serán cifrados con esa clave maestra.

Autenticación e Intercambio/acuerdo de claves

Algoritmo	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Estatus
RSA	Sí	Sí	Sí	Sí	Sí		Definido para TLS 1.2 en RFCs
DH-RSA	No	Sí	Sí	Sí	Sí		
DHE-RSA (forward secrecy)							
ECDH-RSA	No	No	Sí	Sí	Sí		
ECDHE-RSA (forward secrecy)							
DH-DSS	No	Sí	Sí	Sí	Sí		
DHE-DSS (forward secrecy)							
ECDH-ECDSA	No	No	Sí	Sí	Sí		
ECDHE-ECDSA (forward secrecy)							
DH-ANON (inseguro)	No	No	Sí	Sí	Sí		
ECDH-ANON (inseguro)	No	No	Sí	Sí	Sí		
GOST R 34.10-94 / 34.10-2001[13]	No	No	Sí	Sí	Sí		Propuesto en borradores RFC

Referencias:

- Dierks, T. y E. Rescorla (agosto de 2008). «The Transport Layer Security (TLS) Protocol, Version 1.2.
- «THE SSL PROTOCOL». Netscape Corporation. 2007.
- «SSL Pulse: Survey of the SSL Implementation of the Most Popular Web Sites».
- https://es.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=116554854
- https://es.wikipedia.org/w/index.php?title=Protocolo_seguro_de_transferencia_de_hipertexto&oldid=116440804
- «RFC 7301 - Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension».
- https://es.wikipedia.org/w/index.php?title=Protocolo_de_transferencia_de_hipertexto&oldid=11599776