SERVICIOS EN RED

C.F.G.M. SISTEMAS MICROINFORMÁTICOS Y REDES

UD 6. EL SERVICIO WEB

Contenido

1. Introducción a la World Wide Web 2
2. Direcciones URL (Uniform Resource Locator) 2
2.1. Formato general 2
2.2. Elementos de URL 3
3. EL Protocolo HTTP 4
3.1. Funcionamiento y Peticiones HTTP 4
3.2. Los tipos MIME 5
3.3. Sintaxis de mensajes mensajes HTTP. 6
3.4. Cookies 7
4. HTTPS . HIPERTEXT TRANSFER PROTOCOL SECURE 8
4.1. Diferencias con HTTP 8
4.2. SSL y TSL. 9
4.3. Autoridad Certificadora (AC) y Certificado Digital SSL/TLS 9
5. APLICACIONES WEB 11

1. Introducción a la World Wide Web

La World Wide Web (Red Global Mundial, WWW) es un servicio accesible en internet para el acceso a documentos enlazados entre sí. Estos documentos pueden ser solo de texto, llamados entonces hipertextos, o multimedia y/o interactivos, en este caso llamados hipermedios. Están enlazados entre ellos (de forma transparente para el usuario) con hiperenlaces, hipervínculos, links, vínculos o lo que simplemente llamamos enlaces.

Con un navegador web (cliente HTTP) el usuario visualiza los sitios web (site) compuestos por páginas web, en un mismo dominio o subdominio, con texto ASCII (de forma nativa) , imágenes en vídeos, sonido, contenido multimedia, etc.).

2. Direcciones URL (Uniform Resource Locator)

Un Localizador Uniforme de Recursos (URL) está formado por una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que designa recursos en una red, como Internet. Estos recursos pueden ser páginas web, documentos, imágenes, vídeos, sonidos, programas, etc.

El URL es una cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet. Existe un URL único para cada página de cada uno de los documentos de la WWW. Permite que el navegador web la encuentre y la muestre de forma adecuada. Por ello, el URL combina el nombre de la computadora que proporciona la información, el directorio donde se encuentra, el nombre del archivo, y el protocolo a usar para recuperar los datos para que no se pierda alguna información sobre dicho, factor que se emplea para el trabajo.

2.1. Formato general

El formato general de un URL es:

```
esquema://máquina.directorio.archivo
```

También pueden añadirse otros datos:

```
esquema://usuario:contraseña@host:puerto.directorio.archivo
```

Por ejemplo: **`http://www.wikipedia.org/`**

La especificación detallada se encuentra en la RFC 1738, titulada *Uniform Resource Locators*.

Un URL combina en una dirección simple los cinco elementos básicos de información necesarios para recuperar un recurso desde cualquier parte en Internet:

- El protocolo que se usa para comunicar, o enviar datos.
- El usuario de acceso al recurso.
- El anfitrión (servidor o *host*) con el que se comunica.
- El puerto de red en el servidor para conectarse.
- La ruta al recurso en el servidor (por ejemplo, su nombre de archivo).

Un URL típico puede ser del tipo:

`http://es.wikipedia.org:80/wiki/tren`

Donde:

- **`http`** es el protocolo.
- **`es.wikipedia.org`** es el anfitrión o host.
- **`80`** es el número de puerto de red en el servidor (siendo 80 el valor por omisión para el protocolo HTTP, esta porción puede ser omitida por completo).
- **`/wiki/tren`** es la ruta de recurso.

2.2. Elementos de URL

Esquema URL: Es el protocolo con el que se negocia la transmisión o comunicación.

Algunos ejemplos de esquemas URL:

- http - recursos Hypertext Transfer Protocol (HTTP).
- https - HTTP sobre *Secure Sockets Layer* (SSL).
- ftp - *File Transfer Protocol*.
- mailto - direcciones de correo electrónico (e-mail).
- file - recursos disponibles en el sistema local, o en una red local.
- news - grupos de noticias Usenet (newsgroup).
- telnet - el protocolo Telnet.
- h323 – para videoconferencia

Algunos de los esquemas URL, como los populares "mailto", "http", "ftp", y "file", junto a los de sintaxis general URL, se detallaron por primera vez en 1994, Todavía son válidos algunos de los esquemas definidos en el primer RFC, mientras que otros son debatidos o han sido refinados por estándares posteriores.

Userinfo: La URL puede llevar el nombre de usuario solo, o el nombre de usuarioy su contraseña. El userinfo precede al separador arroba (@) y utiliza dos puntos (:) entre usuario y contraseña.

Ejemplo: http://aduser:123456@info.cmex.es:80/aduser/index.html

Host:

Puede ser una FQDN , un nombre de dominio, una IPv4 o una IPv6 como por ejemplo [FEDC:BA98:7654:3210: FEDC:BA98:7654:3210], en este último caso va entre corchetes ([ ]).

Puerto (Port): Indica el puerto de comunicación del protocolo. Puede ser oficial, oficioso o reconfigurado por el administrador del servidor. Le precede el separador dos puntos (:).

Ruta (path): Dentro de la computadora, es la ruta donde se aloja el recurso o documento (permite direccionamiento relativo con directorio raíz y puede usarpunto-punto-barra para la carpeta padre [../]). El separador de directorios es labarra (/). Los archivos tienen un nombre y suelen llevar una extensión .También pueden llevar parámetros separados por punto y coma (;).

Query: Consultas. Son una o varias variables, precedidas de la interrogación (?) y con un valor precedido por un símbolo igual (=). Si son varias se separan con puntos y comas (;).

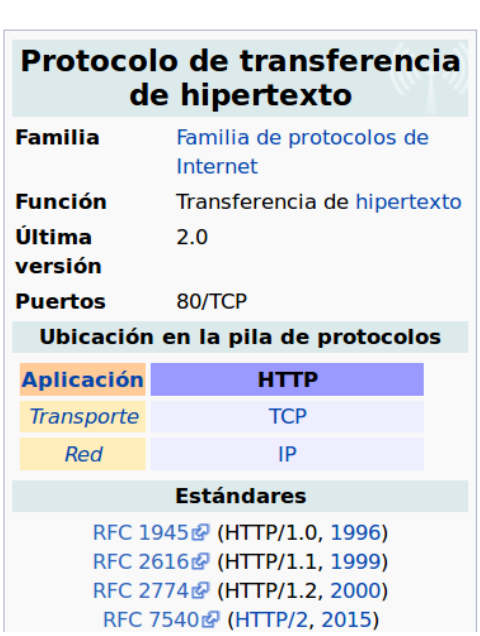
Fragment: El fragmento referencia una zona en el documento, un marcador, una posición. Va precedido de la almohadilla (o símbolo del número [#])

El único parámetro obligatorio en una dirección URL es el host, pues puede tener configurado por defecto el redireccionamiento a una URL completa pero transparente para el usuario.

Ejemplos:

```
ftp://ftp.rediris.es
mailto://user@hotmail.com
http://www.youtube.com/watch?v=2l4hGvSIZSA
```

3. EL Protocolo HTTP



El Protocolo de transferencia de hipertexto (Hypertext Transfer Protocol o HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. HTTP es un protocolo sin estado, es decir, cuando se hace una petición al servidor y se realiza la transacción se vuelve a cerrar el puerto 80 hasta una nueva conexión. No guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener información sobre conexiones amteriores y para ello se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de sesión, y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

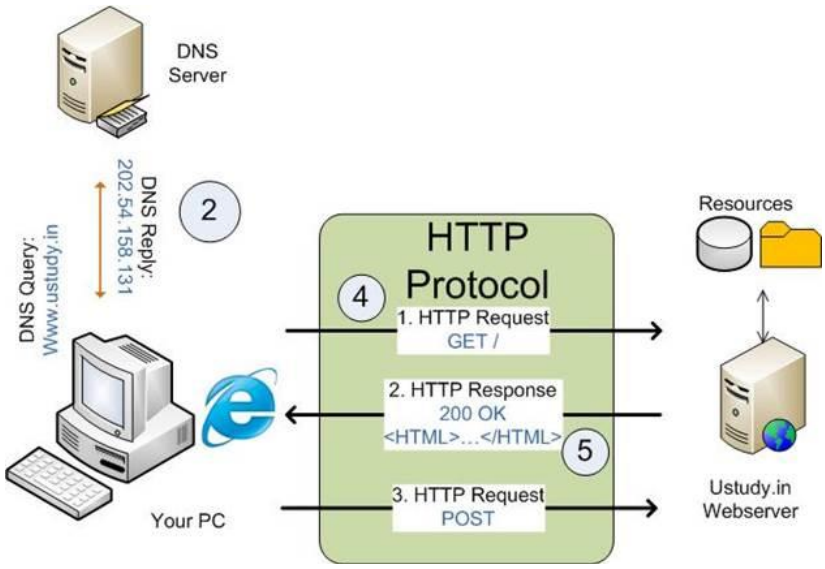
3.1. Funcionamiento y Peticiones HTTP

HTTP es un protocolo cliente servidor por lo tanto su funcionamiento está basado en el envío de mensajes entre ambas partes. Podemos resumir su funcionamiento como de la siguiente forma:

El cliente web es configurado con la dirección que se quiere consultar con una dirección URL.

El cliente web analiza el protocolo y el host, hace la petición de resolución DNS si le es necesario y establece una conexión con el servidor solicitando la página (request).

El servidor envía la información de la página en código html (response).



Los métodos más usados en las peticiones son GET y POST:

- **GET**: pide la representación del recurso especificado.
- **POST**: presenta los datos que se procesarán en la petición del recurso especificado (por ejemplo un formulario, form, de HTML). Los datos se incluyen en el cuerpo de la petición. Este es el método idóneo para programación (PHP, ASP...).
- **HEAD:** Es similar a GET pero en este caso lo que se pide es sólo la cabecera (Metadatos) del recurso web.

3.2. Los tipos MIME

El tipo MIME (Multipurpose Internet Mail Extension) es un mecanismo para decir al cliente la variedad de documentos transmitidos. La extensión de un nombre de archivo no tiene significado en la web. Por lo tanto, es importante que el servidor esté configurado correctamente, de modo que se transmita el tipo MIME correcto con cada documento. Los navegadores a menudo usan el tipo MIME para determinar qué acción predeterminada realizar cuando se recupera un recurso.

Se establecieron inicialmente para mejorar las prestaciones de la mensajería de correo electrónico de forma que se pudieran adjuntar documentos de distintos tipos a los mensajes. Su uso se ha extendido a servicio web.

Se suelen usar con la cabecera Content –Type en mensajes http de la siguiente forma:

```
Content-type: tipo_mime_principal/subtipo_mime
```

Ejemplo: Content-type: image/gif

Se componen de tipo/subtipo. Algunos ejemplos de uso son :

Text/html: Define que los archivos de texto de la transacción son html.

Video/mpeg: Define que todos los archivos de video tranferidos son en formato mpeg.

Image/* :Define todos los archivos de imagen de cualquier formato jpg, gif, bmp,…

Application/pdf: Define que se soportan documentos pfd.

Puedes consultar una lista ampliada de tipos MIME en este enlace:

http://es.ccm.net/contents/649-formatos-y-extensiones-de-archivos

3.3. Sintaxis de mensajes mensajes HTTP.

En las transacciones HTTP, el cliente envía al servidor un mensaje de petición (GET o Head), que contiene varias cabeceras (headers) y datos de esas cabeceras.

```
GET /index.html HTTP/1.1
Host: www.google.com
Accept-Language: en
User-Agent: IE 7
[Línea en blanco]
```

La primera línea indica el tipo de petición GET, el archivo que se pide /index.html y el estándar usado HTTP/1.1 (Es obligatorio indicar el estándar a partir de la versión 1.1.).

Algunas de las cabeceras más usuales en las peticiones de los clientes son servidor son:

- Host: El nombre de dominio o dirección IP (puede incluir número de puerto). El uso de la cabecera es obligatorio a partir de HTTP 1.1.
- Accept-Language: Idiomas que se aceptan.
- User-Agent: Contiene la información del cliente de la petición, como el navegador, el sistema operativo, etc
- Content-type: *El Tipo Mime* o tipo de contenido. Esta cabecera aparece en peticiones que usen el método POST par indicar el tipo de contenido enviado.

Puedes consultar las siguiente tabla de cabeceras en las peticiones Http para más información:

https://es.wikipedia.org/wiki/Anexo:Cabeceras_HTTP

El servidor responderá con un mensaje dividido en un encabezado y el cuerpo de la página solicitada. En este encabezado también se incluyen cabeceras acerca de la información servida al cliente.

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2009 23:59:59 GMT
Content-Length: 1221
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Content-Type: text/html; charset=UTF-8
<html>
<body> (Contenido de la página web)... </body>
</html>
```

La primera línea contiene el código de estado de la petición.

HTTP/1.1. 200 OK es la respuesta estándar para peticiones correctas. Por ejemplo la respuesta HTTP/1.1. 204 No Content indicaría que la petición se ha completado pero la respuesta no tiene ningún contenido o la respuesta o HTTP/1.1. 400 Bad Request cuando la petición contiene sintaxis errónea.

Los principales códigos de estado son:

- `1XX` Mensajes informativos:
  - `100` Continuar (continuar con la petición).
- `2XX` Operaciones exitosas:
  - `200` OK (todo correcto).
  - `206` Contenido Parcial (Partial Content).
- `3XX` Redirección a otra URL:
  - `301` Mudado Permanentemente (Moved Permanently).
  - `307` Redirección Temporal (Temporary Redirect).
- `4XX` Error por parte del cliente:
  - `401` No autorizado (Unauthorized).
  - `403` Prohibido (Forbidden).
  - `404` No Encontrado (Not Found o File Not Found, puede ser que no se haya escrito bien la URL, o que se haya puesto un espacio en blanco o tildes. Esteerror es el más común).
  - `408` Tiempo de Espera Agotado (Request Timeout).
- `5XX` Error por parte del servidor:
  - `500` Error Interno (Internal Server Error).
  - `503` Servicio No Disponible (Service Unavailable).

Puedes consultar la mas información sobre códigos de respuestas HTTP en la siguiente dirección:

https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP

Algunas de las cabeceras más usuales en las respuestas de servidor son:

- Date: Indica la fecha y hora exacta a la que el servidor ha realizado la respuesta para que conste (la hora del servidor).
- Content-Length: El tamaño de la respuesta en octetos (8 bits)
- Content-type: *El Tipo Mime* o tipo de contenido que se envía como respuesta. Esta cabecera puede apareceré en peticiones que usen el método POST par indicar el tipo de contenido enviado.
- Server: indica el tipo de servidor HTTP empleado.

3.4. Cookies

Las cookies o «galletas» son archivos que el navegador del cliente graba en el disco duro a petición del servidor. Estos archivos almacenan datos que normalmente utiliza el servidor en otras conexiones.

Se suelen utilizar para:

• Guardar los nombres de usuario y contraseñas (son útiles para «cestas de la compra», blogs y otras páginas que necesitan mantener datos entre sesiones).

• Recopilar información de virus, hábitos de navegación de los usuarios, spyware... y usos publicitarios.

Las cookies se pueden generar con distintos lenguajes como JavaScript, etc.

Cuando el servidor envía una respuesta incluye el parámetro **Set-Cookie: name=value** y, a partir de ese momento, el navegador del cliente añade el parámetro **Cookie: name=value** a todas las peticiones de ese servidor (y de los servidores de elementos externos que contenga ese recurso). En los navegadores las cookies se pueden activar, desactivar o preguntar cada vez que se vaya a enviar alguna.

Existen varios tipos de *cookies*, pero a las más comunes se les llama ***session cookies***, que tienen un corto tiempo de vida ya que **son borradas cuando cierras el navegador**. También tenemos ***persistent cookies*** o *cookies* persistentes, que se usan para rastrear al usuario guardando información sobre su comportamiento en un sitio web durante un período de tiempo determinado; las *cookies* persistentes **pueden ser borradas limpiando los datos del navegador pero algunas tienen una fecha de expiración**.

Las ***secure cookies*** o *cookies* seguras almacenan información cifrada para evitar que los datos almacenados en ellas sean vulnerables a ataques maliciosos de terceros. **Se usan sólo en conexiones HTTPS.**

4. HTTPS . HIPERTEXT TRANSFER PROTOCOL SECURE

Hypertext Transfer Protocol Secure (en español: *Protocolo seguro de transferencia de hipertexto*), más conocido por sus siglas **HTTPS**, es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

El sistema HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible (usuario y claves de paso normalmente) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.

El puerto estándar para este protocolo es el 443.

Hypertext Transfer Protocol Secure (HTTPS)	
Familia	Familia de protocolos de Internet
Función	Transferencia segura de hipertexto
Puertos	443/TCP
Ubicación en la pila de protocolos	
Aplicación	**HTTPS**
Transporte	SSL/TLS
	TCP
Red	IP
Estándares	
RFC 2818 – HTTP sobre TLS	

Este servicio requiere la confianza de la Autoridad de Certificación que necesita instalar plugins en el navegador. Una vez instalado, el navegador nos confirma que estamos en zona segura o insegura (Internet Explorer, Firefox y Google Chrome lo hacen con un candado abierto en la parte derecha de la dirección o en la parte inferior derecha de la barra de estado.

4.1. Diferencias con HTTP

En el protocolo HTTP las URLs comienzan con "http://" y utilizan por omisión el puerto 80, las URLs de HTTPS comienzan con "https://" y utilizan el puerto 443 por omisión.

HTTP es inseguro y está sujeto a ataques que pueden permitir al atacante obtener acceso a cuentas de un

sitio web e información confidencial. HTTPS está diseñado para resistir esos ataques y ser más seguro.

4.2. SSL y TSL.

HTTP opera en la capa más alta del modelo OSI, la capa de aplicación; pero el protocolo de seguridad opera en una subcapa más baja, cifrando un mensaje HTTP previo a la transmisión y descifrando un mensaje una vez recibido. Estrictamente hablando, HTTPS no es un protocolo separado, pero refiere el uso del HTTP ordinario sobre una Capa de Conexión Segura Cifrada.

SSL (Secure Sockets Layer) y TLS (Transport Layer Security) son protocolos que hacen uso de certificados digitales para establecer comunicaciones seguras a través de Internet.

Te permite confiar información personal a sitios web, ya que tus datos se ocultan a través de métodos criptográficos mientras navegas en sitios seguros.Es utilizado ampliamente en bancos, tiendas en línea y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas. No todos los sitios web usan SSL, por eso debes ser cuidadoso.

Se usan certificados X.509 y por lo tanto criptografía asimétrica para autentificar a la contraparte con quien se están comunicando y para intercambiar una llave simétrica. Esta sesión es luego usada para cifrar el flujo de datos entre las partes.

El protocolo SSL fue desarrollado originalmente por Netscape. Las priemras versiones 1.0 y 2.0 contenía una cantidad de falloss de seguridad que al final llevaron al diseño de la versión SSL 3.0. Dicha versión, presentada en 1996. SSL 3.0 fue publicado por la Internet Engineering Task Force (IETF) en el RFC 6101.

TLS (*Transport Layer Security*) es un protocolo Internet Engineering Task Force (IETF), definido por primera vez en 1999 y actualizado en el RFC 5246 (agosto de 2008) y en RFC 6176 (marzo de 2011). Se basa en las especificaciones previas de SSL 3.0. Actualmente se recomienda el uso de TSL debido a los fallos de seguridad encontrados en SSL.

Versiones de estos protocolos están ampliamente utilizadas además de en navegación web en otras aplicaciones como correo electrónico, fax por Internet, mensajería instantánea, y voz-sobre-IP (VoIP).

SSL/TSL utilizan un método denominado “handshake” (apretón de manos) por el que negocian el envío de la clave pública del servidor al cliente por un método de encriptación. Fue una medida de mejora de la seguridad que evita interceptar la clave pública y suplantación de identidad del cliente.

4.3. Autoridad Certificadora (AC) y Certificado Digital SSL/TLS

Una Autoridad Certificadora (AC, en inglés CA) es una entidad confiable que se encarga de garantizar que el poseedor de un certificado digital sea quien dice ser, bridando confianza a ambas partes de una comunicación segura SSL/TLS.

Un certificado digital para su uso con SSL/TLS es un documento digital único que garantiza la vinculación entre una persona o entidad con su llave pública.

Contiene información de su propietario como nombre, dirección, correo electrónico, organización a la que pertenece y su llave pública, así como información propia del certificado por mencionar: periodo de validez, número de serie único, nombre de la AC que emitió, firma digital de la AC cifrada con su llave privada y otros datos más que indican cómo puede usarse ese certificado.

Funcionamiento de HTTPS

Vamos a interpretar el funcionamiento en una conexión https://facebook.com

Cuando el navegador hace una petición al sitio seguro de Facebook, éste envía un mensaje donde indica que quiere establecer una conexión segura y envía datos sobre la versión del protocolo SSL/TLS que soporta y otros parámetros necesarios para la conexión.

En base a esta información enviada por el navegador, el servidor web de Facebook responde con un mensaje informando que está de acuerdo en establecer la conexión segura con los datos de SSL/TLS proporcionados.

Una vez que ambos conocen los parámetros de conexión, el sitio de Facebook presenta su certificado digital al navegador web para identificarse como un sitio confiable.

Verificación de validez del certificado

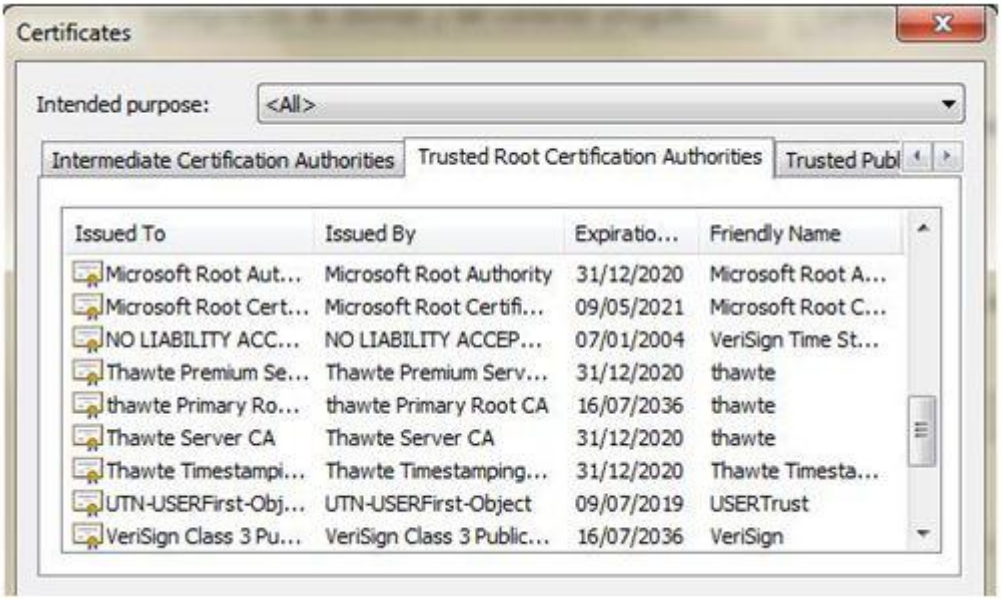
Una vez que el navegador tiene el certificado del sitio web de Facebook, realiza algunas verificaciones antes de confiar en el sitio:

Integridad del certificado: Verifica que el certificado se encuentre íntegro, esto lo hace descifrando la firma digital incluida en él mediante la llave pública de la AC y comparándola con una firma del certificado generada en ese momento, si ambas son iguales entonces el certificado es válido.

Vigencia del certificado: Revisa el periodo de validez del certificado, es decir, la fecha de emisión y la fecha de expiración incluidos en él.

Verifica emisor del certificado: Hace uso de una lista de Certificados Raíz almacenados en tu computadora y que contienen las llaves públicas de las ACs conocidas y de confianza (Imagen 2). Puedes acceder a esta lista desde las opciones avanzadas de tu navegador web (en este caso usamos Google Chrome).

Con base a esta lista, el navegador revisa que la AC del certificado sea de confianza, de no serlo, el navegador mostrará una advertencia indicando que el certificado fue emitido por una entidad en la cual no confía.



Estableciendo la conexión segura

Una vez que el certificado cumplió con todas las pruebas del navegador, se establece la conexión segura al sitio de Facebook, lo cual se traduce en seguridad para tus valiosos datos personales.

5. APLICACIONES WEB

Las aplicaciones web son aquellas que los usuarios pueden utilizar accediendo a un servidor web. Suele tratarse de aplicaciones que se codifican en un lenguaje soportado por los navegadores (HTML, JavaScript, Java, PHP, ASP.NET [C#, VB.NET o VBScript], Perl, Ruby, Python, XML, ActionScript [Flash]).

Las más populares son:

•Webmail: correo electrónico web, como Hotmail o Gmail.

•Wiki: web cuyo contenido es colaborativo, cualquier usuario puede añadir o editar su contenido. Usa una tecnología donde el título de la página es la URL relativa de la página web. Los wikis más conocidos son Wikipedia, Mediawiki, etc.

• Weblogs: bitácoras o diarios, como Blogger, WordPress, Fotolog, Videolog, etc.

•Tiendas en línea: montadas por el usuario o, por ejemplo, por su caja o banco.

•Juegos: como Counter Strike, WOW, Aion, Conan, Warcraft, Lineage, Minijuegos, etc.

•Aplicaciones ofimáticas: Google Wave o Google Apps (suites), Google Spreadsheets (hoja de cálculo), Google Page Creator (editor de páginas web), GoogleCalc (calculadora), etc.

•SO: sistemas operativos, como EyeOS.

•Videochat: existen muchos de código abierto en Flash.

•Webmessenger: como el Windows Live Messenger Online.

•Otros: de diseño (Adobe Photoshop Express), de antivirus (como QuickScan de Bitdefender, Panda ActiveScan, Kaspersky Online, McAfee FreeScan, EsetNOD32), de compresores (Shrinkfile.net, Zip online, Wobzip, Krunch), de vídeo (Youtube), de utilidades (File Destructor), etc.