



PROTOSCOLOS: HTTP HTTPS

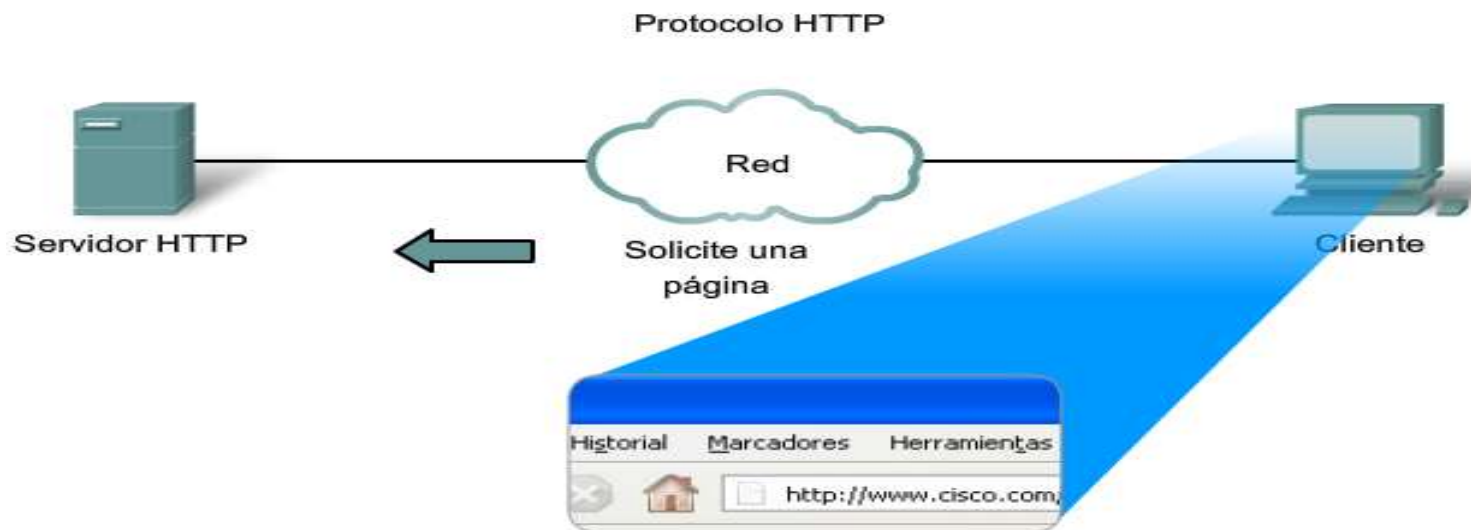


https://




Servicio www y HTTP

- Cuando se escribe una dirección Web (o **URL**) en un explorador de Internet, el explorador establece una conexión con el servicio Web del servidor que utiliza el protocolo HTTP. Los navegadores Web son las aplicaciones cliente que utilizamos para conectarnos a la World Wide Web y acceder a recursos almacenados en un servidor Web.
- Para acceder al contenido, los clientes Web realizan conexiones al servidor y solicitan los recursos deseados. El servidor responde con el recurso y, al recibirlo, el explorador interpreta los datos y los presenta al usuario.



Pasos utilizados por http

Primero, el explorador interpreta las tres partes de la URL: http:// www.palomatica.info

- http (el protocolo o esquema)
- www.palomatica.info (el nombre del servidor)
- index.html (el nombre de archivo específico solicitado)

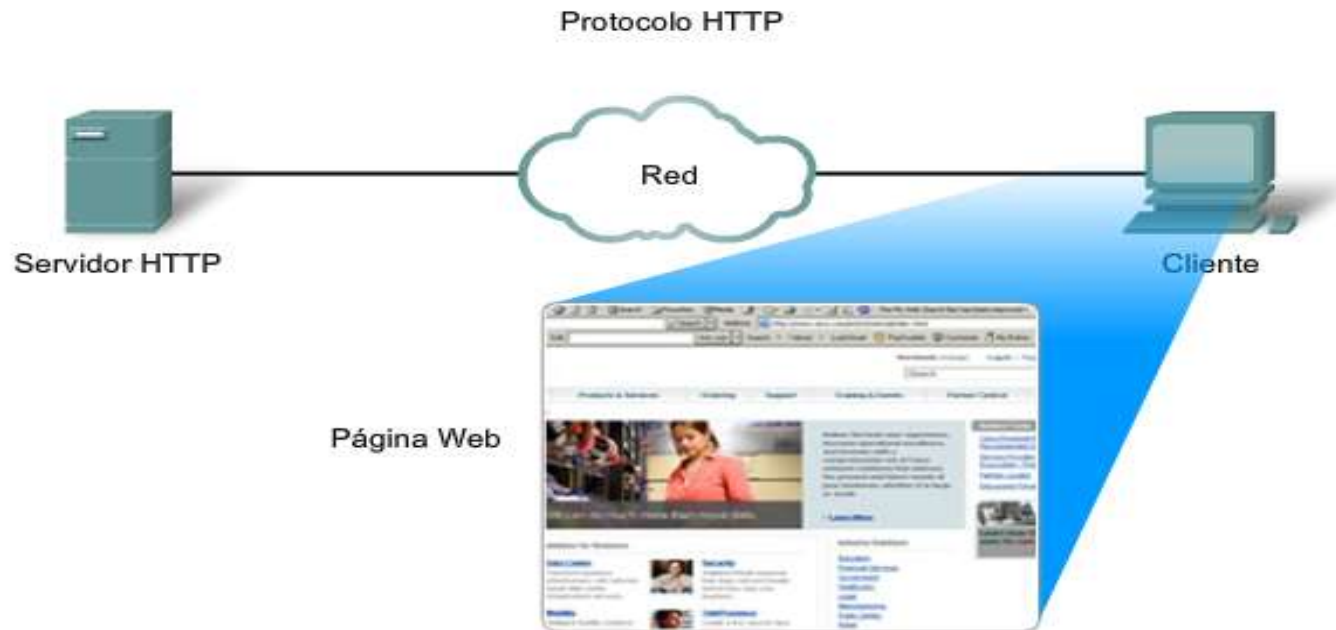
Después, el DNS convierte www.palomatica.info en una dirección numérica, que utilizará para conectarse con el servidor. Al utilizar los requerimientos del protocolo HTTP, el navegador (user agent) envía una solicitud GET al servidor y pide el archivo index.html. El servidor, a su vez, envía al explorador el código HTML de esta página Web. Finalmente, el explorador descifra el código HTML y da formato a la página para la ventana del explorador.



En respuesta a la solicitud, el servidor HTTP envía el código para una página Web.

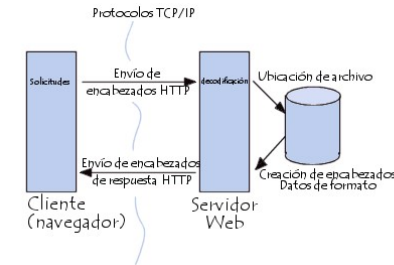
HTTP

- Se basa en el envío de mensajes entre un cliente (navegador web) y un servidor. Dichos mensajes son la unidad básica de la comunicación HTTP.
- El cliente es siempre quien inicia la comunicación, y para ello envía un mensaje que contiene una petición, a lo que el servidor contesta con otro que contiene la respuesta.



El navegador interpreta el código HTML y muestra una página Web.

HTTP/HTTPS



- HTTP es un **protocolo sin estado**, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto 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.
- Para una comunicación segura a través de Internet, se utiliza el protocolo HTTP seguro (**HTTPS**) para acceder o subir información al servidor Web. HTTPS puede utilizar autenticación y cifrado para asegurar los datos cuando viajan entre el cliente y el servidor. HTTPS especifica reglas adicionales para pasar los datos entre la capa de aplicación y la capa de transporte. El sistema HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado. El puerto estándar para este protocolo es el 443

Transacciones HTTP

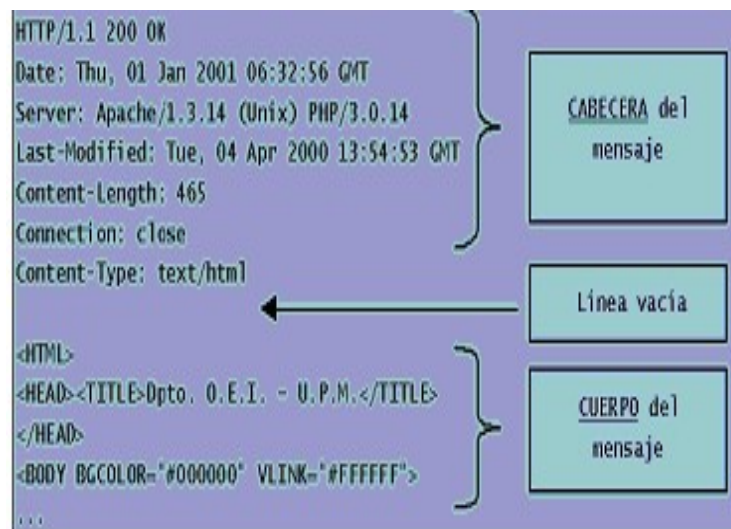
- Una **transacción** HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado.
- El uso de campos de encabezados enviados en las transacciones HTTP le dan gran flexibilidad al protocolo. Estos campos permiten que se envíe información descriptiva en la transacción, permitiendo así la autenticación, cifrado e identificación de usuario.
- Un encabezado es un bloque de datos que precede a la información propiamente dicha, por lo que muchas veces se hace referencia a él como **metadato** —porque tiene datos sobre los datos—.

Partes de un mensaje

- Cada mensaje, está formado por dos partes: la **cabecera** y el **cuerpo del mensaje**. La cabecera se separa del cuerpo del mensaje por una línea vacía. Por su parte, la primera línea de la cabecera es una línea de solicitud (en las peticiones) o una línea de estado (para las respuestas).
- Formato general de la cabecera:
variable_de_cabecera : valor
- En la cabecera de un mensaje se define información acerca de su contenido. Así, por ejemplo, se define el tipo de mensaje, el tipo MIME del contenido, el tamaño, etc.
- La estructura de cada línea consiste en un nombre de campo seguido del carácter dos puntos (':') y el valor asociado a dicho campo. Estos campos permiten añadir una importante característica al protocolo HTTP: la negociación de los tipos y la representación de contenidos entre cliente y servidor.

Composición de los Mensajes

- En el cuerpo del mensaje (si es que existe, ya que un mensaje puede consistir sólo en una cabecera) se incluye la información transmitida entre cliente y servidor. El cuerpo puede contener datos de un formulario, una imagen, un documento, etc.



Composición de los Mensajes

Ejemplo de mensaje de Solicitud (Request)

```
Frame 62: 939 bytes on wire (7512 bits), 939 bytes captured (7512 bits) on interface 0
Ethernet II, Src: CadmusCo_81:bd:ef (08:00:27:81:bd:ef), Dst: CompalBr_60:50:42 (dc:53:7c:60:50:42)
Internet Protocol Version 4, Src: 192.168.1.133 (192.168.1.133), Dst: 80.252.91.41 (80.252.91.41)
Transmission Control Protocol, Src Port: 1570 (1570), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 885
Hypertext Transfer Protocol
  [truncated]GET /BurstingPipe/adserver.bs?cn=int&iv=2&int=31061146~0~7124303~6899537154779705264^VsR~0~0~01020^VsRag~0~0~01020^VsRad~0~0~01020^VsIAB~
  Host: bs.serving-sys.com\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36\r\n
  Origin: http://elpais.com\r\n
  Accept: */*\r\n
  Referer: http://elpais.com/\r\n
  Accept-Encoding: gzip, deflate, sdch\r\n
  Accept-Language: es-ES,es;q=0.8\r\n
  Cookie: s_14978431=3400540521139074505; C5=; D2=; eyeblaster=FLV=19&RES=0; A6=01Laxtjcqh000ewc00004000001RXMuJcqh000eKJ000000000015fT4jCqh000mVb00001000\r\n
  [Full request URI [truncated]: http://bs.serving-sys.com/BurstingPipe/adserver.bs?cn=int&iv=2&int=31061146~0~7124303~6899537154779705264^VsR~0~0~01020^VsRag~0~0~01020^VsRad~0~0~01020^VsIAB~
  [HTTP request 1/1]
  [Response in frame: 73]
```

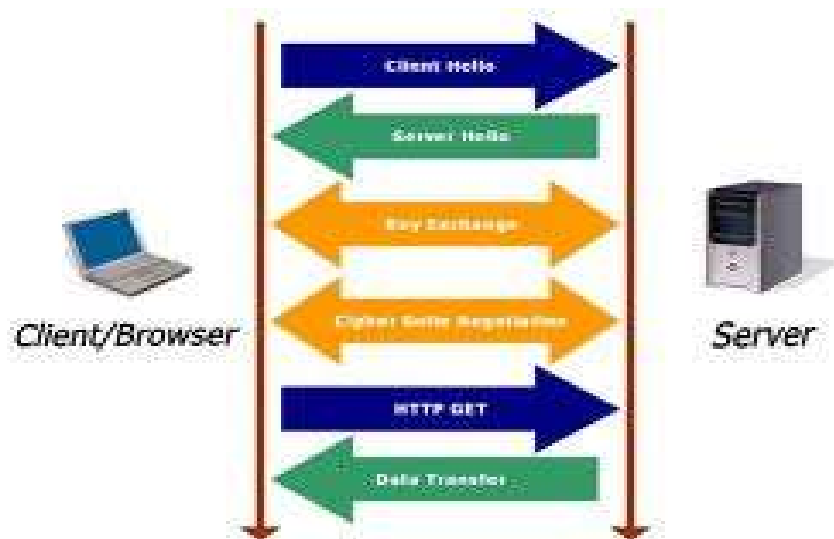
Mensajes de Solicitud

- Cuando un usuario pulsa sobre un enlace, el navegador se encarga de enviar un mensaje de solicitud al servidor correspondiente. Este mensaje contiene en la primera línea de su cabecera el **método** de solicitud empleado, el **identificador del recurso** solicitado y la **versión** del protocolo empleado. En la cabecera del mensaje y a continuación de esta primera línea, opcionalmente pueden figurar muchos campos que aportan información del cliente hacia el servidor.

Métodos http

Método	Objetivo de la Solicitud
OPTIONS	Información relativa a opciones de comunicación de un recurso
GET	Solicita un recurso (documento HTML, script PHP, imagen, etc.)
HEAD	Obtiene metainformación acerca de un recurso
POST	Envía un mensaje con datos para ser procesados por un recurso
PUT	Envía un documento para ser almacenado en el servidor
DELETE	Solicita la eliminación de un recurso en el servidor
TRACE	Genera información de diagnóstico relativa a la comunicación
CONNECT	Método reservado para comunicaciones a través de proxys

Métodos más empleados:



Comando	Descripción
GET	Solicita el recurso ubicado en la URL especificada
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Envía datos al programa ubicado en la URL especificada
PUT	Envía datos a la URL especificada
DELETE	Borra el recurso ubicado en la URL especificada

Métodos más empleados:

- El método **GET** se emplea para solicitar información y, a la vez, en la misma petición enviar variables con sus correspondientes valores. Estas parejas de variables y valores se concatenan en el URI de la petición a continuación del recurso precedidas por el carácter de cierre de interrogación ('?').
- Con el método **POST**, los datos del formulario son enviados en el cuerpo del mensaje y no en la cadena de la solicitud como ocurría con GET. Por tanto, en el URI solicitado, la cadena de la consulta (query string) suele estar vacía y exclusivamente suele figurar el nombre del recurso que realizará el procesamiento de dicho formulario.

Encabezados

Nombre del encabezado	Descripción
Accept	Tipo de contenido aceptado por el navegador (por ejemplo, <i>texto/html</i>). Consulte Tipos de MIME
Accept-Charset	Juego de caracteres que el navegador espera
Accept-Encoding	Codificación de datos que el navegador acepta
Accept-Language	Idioma que el navegador espera (de forma predeterminada, inglés)
Authorization	Identificación del navegador en el servidor
Content-Encoding	Tipo de codificación para el cuerpo de la solicitud
Content-Language	Tipo de idioma en el cuerpo de la solicitud
Content-Length	Extensión del cuerpo de la solicitud
Content-Type	Tipo de contenido del cuerpo de la solicitud (por ejemplo, <i>texto/html</i>). Consulte Tipos de MIME
Date	Fecha en que comienza la transferencia de datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor
From	Permite especificar la dirección de correo electrónico del cliente
From	Permite especificar que debe enviarse el documento si ha sido modificado desde una fecha en particular
Link	Vínculo entre dos direcciones URL
Orig-URL	Dirección URL donde se originó la solicitud
Referer	Dirección URL desde la cual se realizó la solicitud
User-Agent	Cadena con información sobre el cliente, por ejemplo, el nombre y la versión del navegador y el sistema operativo

CUERPO DEL MENSAJE

- El cuerpo de un mensaje pueden ser muy largo, y pueden contener cualquier cosa
- El tipo lo indica el header "Content-Type"
 - text/plain = documento en texto plano
 - text/html = documento texto HTML
 - audio/mpeg = audio en formato MP3
 - image/jpeg = imagen en formato JPEG
 - application/pdf = documento formato PDF
 - application/x-shockwave-flash = Adobe Flash
 - application/octet-stream = secuencia de bytes

PETICIÓN GET



```
GET /charla/ajax/hellohtml.html HTTP/1.1
Host: ostertagxp
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en-US,es-CL;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0)
Connection: Keep-Alive
```


RESPUESTA HTTP



HTTP/1.1 200 OK

Server: Sun Java System Application Server 9.1

Date: Mon, 15 Oct 2007 12:40:42 GMT

Last-Modified: Mon, 15 Oct 2007 12:22:23 GMT

Content-Type: text/html; charset=iso-8859-1

<HTML><HEAD>...</HEAD><BODY>.....</BODY></HTML>

Respuesta HTTP

Una respuesta HTTP es un conjunto de líneas que el servidor envía al navegador. Está constituida por:

- **Una línea de estado:** es una línea que especifica la versión del protocolo utilizada y el estado de la solicitud en proceso mediante un texto explicativo y un código. La línea está formada por tres elementos que deben estar separados por un espacio:
 - la **versión** del protocolo utilizada
 - el **código de estado**
 - el **significado** del código
- **Los campos del encabezado de respuesta:** es un conjunto de líneas opcionales que permiten aportar información adicional sobre la respuesta y/o el servidor. Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.
- **El cuerpo de la respuesta:** contiene el documento solicitado.

Ejemplo de mensaje de Respuesta (Response)

```
⊕ Internet Protocol Version 4, Src: 80.252.91.41 (80.252.91.41), Dst: 192.168.1.133 (192.168.1.133)
⊕ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 1570 (1570), Seq: 1, Ack: 886, Len: 675
⊖ Hypertext Transfer Protocol
  ⊕ HTTP/1.1 200 OK\r\n
    Cache-Control: no-cache, no-store\r\n
    Pragma: no-cache\r\n
  ⊕ Content-Length: 0\r\n
    Content-Type: text/html\r\n
    Expires: Sun, 05-Jun-2005 22:00:00 GMT\r\n
    Set-Cookie: A6=01Laxtjcqh000ewc00004000001RXmujcqh000ekJ000000000015fT4jcqh000mVb000001000001sviqjcqh000ekJ000001jcqi; expires=Thu, 07-Jan-2016 02:26:07
    Set-Cookie: u2=480824df-cbe0-4af5-8404-46d5ed3b2d6945d05g; expires=Thu, 07-Jan-2016 02:26:07 GMT; domain=.serving-sys.com; path=/\r\n
    Access-Control-Allow-Credentials: true\r\n
    Access-Control-Allow-Origin: http://elpais.com\r\n
    X-Powered-By: ASP.NET\r\n
    P3P: CP="NOI DEVa OUR BUS UNI"\r\n
    Date: Fri, 09 Oct 2015 06:26:07 GMT\r\n
    Connection: close\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.047135000 seconds]
    [Request in frame: 62]
```

Encabezados de respuesta

Nombre del encabezado	Descripción
Content-Encoding	Tipo de codificación para el cuerpo de la respuesta
Content-Language	Tipo de idioma en el cuerpo de la respuesta
Content-Length	Extensión del cuerpo de la respuesta
Content-Type	Tipo de contenido del cuerpo de la respuesta (por ejemplo, <i>texto/html</i>). Consulte Tipos de MIME
Date	Fecha en que comienza la transferencia de datos
Expires	Fecha límite de uso de los datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor
Location	Redireccionamiento a una nueva dirección URL asociada con el documento
Server	Características del servidor que envió la respuesta

Códigos de Respuestas HTTP

- 100-199 Informativos
 - Reservados para el futuro
- 200-299 Éxito
 - 200 = OK
- 300-399 Redirección
 - 301 = Se movió permanentemente
 - 302 = Se movió temporalmente
 - 304 = No ha cambiado
- 400-499 Error del cliente
 - 400 = Requerimiento inválido
 - 401 = No está autorizado
 - 403 = Está prohibido
 - 404 = No se encontró
- 500-599 Error del servidor
 - 500 = Error interno del servidor
 - 501 = No está implementado

Los códigos de respuesta

- Son los códigos que se ven cuando el navegador no puede mostrar la página solicitada, cuando se envía correctamente, etc. El código de respuesta está formado por tres dígitos: el primero indica el estado y los dos siguientes explican la naturaleza exacta del error.

Código	Mensaje	Descripción
10x	Mensaje de información	Estos códigos no se utilizan en la versión 1.0 del protocolo
20x	Éxito	Estos códigos indican la correcta ejecución de la transacción
200	OK	La solicitud se llevó a cabo de manera correcta
201	CREATED	Sigue a un comando POST e indica el éxito, la parte restante del cuerpo indica la dirección URL donde se ubicará el documento creado recientemente.
202	ACCEPTED	La solicitud ha sido aceptada, pero el procedimiento que sigue no se ha llevado a cabo
203	PARTIAL INFORMATION	Cuando se recibe este código en respuesta a un comando de GET indica que la respuesta no está completa.
204	NO RESPONSE	El servidor ha recibido la solicitud, pero no hay información de respuesta
205	RESET CONTENT	El servidor le indica al navegador que borre el contenido en los campos de un formulario
206	PARTIAL CONTENT	Es una respuesta a una solicitud que consiste en el encabezado <i>range</i> . El servidor debe indicar el encabezado <i>content-Range</i>

Los códigos de respuesta

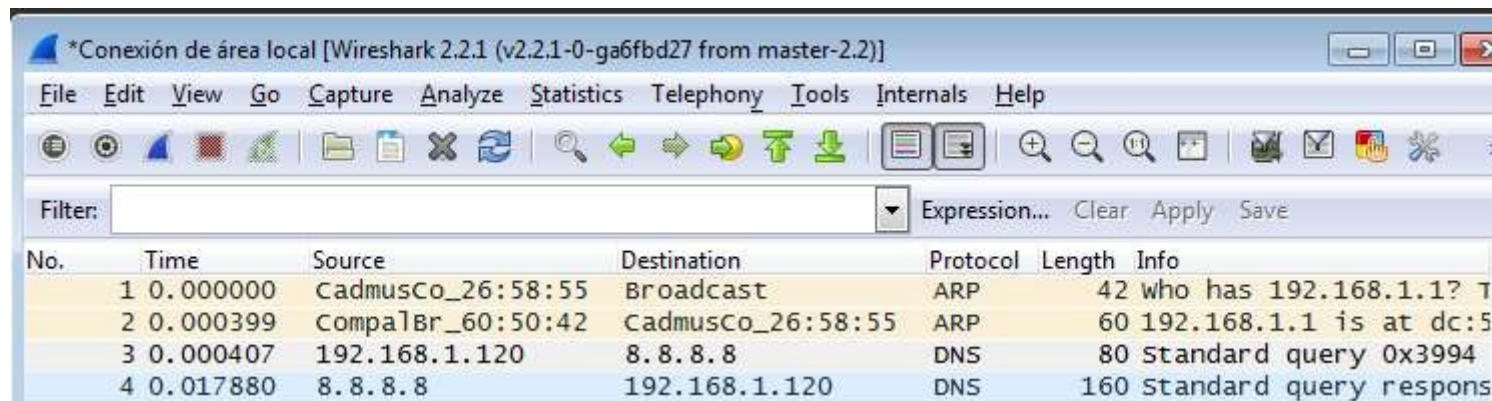
30x	Redirección	Estos códigos indican que el recurso ya no se encuentra en la ubicación especificada
301	MOVED	Los datos solicitados han sido transferidos a una nueva dirección
302	FOUND	Los datos solicitados se encuentran en una nueva dirección URL, pero, no obstante, pueden haber sido trasladados
303	METHOD	Significa que el cliente debe intentarlo con una nueva dirección; es preferible que intente con otro método en vez de GET
304	NOT MODIFIED	Si el cliente llevó a cabo un comando GET condicional (con la solicitud relativa a si el documento ha sido modificado desde la última vez) y el documento no ha sido modificado, este código se envía como respuesta.
40x	Error debido al cliente	Estos códigos indican que la solicitud es incorrecta
400	BAD REQUEST	La sintaxis de la solicitud se encuentra formulada de manera errónea o es imposible de responder
401	UNAUTHORIZED	Los parámetros del mensaje aportan las especificaciones de formularios de autorización que se admiten. El cliente debe reformular la solicitud con los datos de autorización correctos
402	PAYMENT REQUIRED	El cliente debe reformular la solicitud con los datos de pago correctos
403	FORBIDDEN	El acceso al recurso simplemente se deniega
404	NOT FOUND	Un clásico. El servidor no halló nada en la dirección especificada. Se ha abandonado sin dejar una dirección para redireccionar... :)

Los códigos de respuesta

50x	Error debido al servidor	Estos códigos indican que existe un error interno en el servidor
500	INTERNAL ERROR	El servidor encontró una condición inesperada que le impide seguir con la solicitud (una de esas cosas que les suceden a los servidores...)
501	NOT IMPLEMENTED	El servidor no admite el servicio solicitado (no puede saberlo todo...)
502	BAD GATEWAY	El servidor que actúa como una puerta de enlace o proxy ha recibido una respuesta no válida del servidor al que intenta acceder
503	SERVICE UNAVAILABLE	El servidor no puede responder en ese momento debido a que se encuentra congestionado (todas las líneas de comunicación se encuentran congestionadas, inténtelo de nuevo más adelante)
504	GATEWAY TIMEOUT	La respuesta del servidor ha llevado demasiado tiempo en relación al tiempo de espera que la puerta de enlace podía admitir (excedió el tiempo asignado...)

Análisis de una petición http

- Primero se ejecutará el wireShark y desde un navegador se lanzará una petición de conexión a una página web.
- Las primeras tramas que aparecerán corresponden a ARP, ya que es necesario determinar la MAC de la IP destino.
- Posteriormente veremos las peticiones DNS ya que se conoce la URL pero no la IP destino



*Conexión de área local [Wireshark 2.2.1 (v2.2.1-0-ga6fbd27 from master-2.2)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

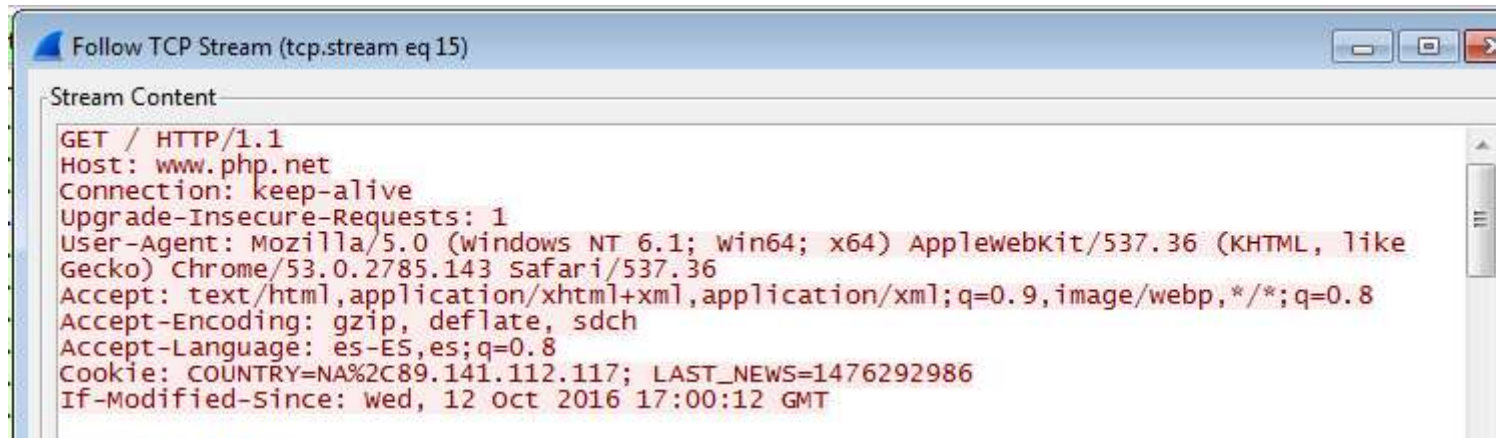
Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CadmusCo_26:58:55	Broadcast	ARP	42	who has 192.168.1.1? T
2	0.000399	Compa1Br_60:50:42	CadmusCo_26:58:55	ARP	60	192.168.1.1 is at dc:5
3	0.000407	192.168.1.120	8.8.8.8	DNS	80	standard query 0x3994
4	0.017880	8.8.8.8	192.168.1.120	DNS	160	standard query respons

Petición / REQUEST

117	5.855187	192.168.1.120	83.137.20.135	HTTP	537 GET / HTTP/1.1
126	5.895612	83.137.20.135	192.168.1.120	TCP	60 80→50126 [ACK] Seq=1 Ack=484 win=15744 Len=0
127	5.928055	83.137.20.135	192.168.1.120	TCP	1514 [TCP segment of a reassembled PDU]
128	5.929006	83.137.20.135	192.168.1.120	TCP	1514 [TCP segment of a reassembled PDU]
129	5.929016	192.168.1.120	83.137.20.135	TCP	54 50126→80 [ACK] Seq=484 Ack=2921 win=65536 Len=0
130	5.929192	83.137.20.135	192.168.1.120	TCP	1514 [TCP segment of a reassembled PDU]
131	5.929193	83.137.20.135	192.168.1.120	HTTP	974 HTTP/1.1 200 OK (text/html)
Source Port: 50126					
Destination Port: 80					
[Stream index: 15]					
[TCP Segment Len: 483]					
Sequence number: 1 (relative sequence number)					
[Next sequence number: 484 (relative sequence number)]					
Acknowledgment number: 1 (relative ack number)					
Header Length: 20 bytes					
⊕ Flags: 0x018 (PSH, ACK)					
window size value: 256					
[calculated window size: 65536]					
[window size scaling factor: 256]					
Checksum: 0x2c2e [unverified]					
[Checksum Status: Unverified]					
Urgent pointer: 0					
⊕ [SEQ/ACK analysis]					
⊖ Hypertext Transfer Protocol					
⊖ GET / HTTP/1.1\r\n					
⊕ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]					
Request Method: GET					
Request URI: /					
Request Version: HTTP/1.1					
Host: www.php.net\r\n					
Connection: keep-alive\r\n					
Upgrade-Insecure-Requests: 1\r\n					
User-Agent: Mozilla/5.0 (Windows NT 6.1; win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36\r\n					
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n					
Accept-Encoding: gzip, deflate, sdch\r\n					
Accept-Language: es-ES,es;q=0.8\r\n					
⊕ Cookie: COUNTRY=NA%2C89.141.112.117; LAST_NEWS=1476292986\r\n					
If-Modified-Since: wed, 12 oct 2016 17:00:12 GMT\r\n					
01d0	41 53 54 5f 4e 45 57 53	3d 31 34 37 36 32 39 32	AST_NEWS=1476292		
01e0	39 38 36 0d 0a 49 66 2d	4d 6f 64 69 66 69 65 64	986. If-Modified		
01f0	2d 53 69 6e 63 65 3a 20	57 65 64 7c 20 31 32 20	-Since: wed, 12		

Petición / REQUEST



The image shows a screenshot of a network packet capture tool window titled "Follow TCP Stream (tcp.stream eq 15)". The window displays the "Stream Content" for an HTTP GET request. The text is as follows:

```
GET / HTTP/1.1
Host: www.php.net
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (windows NT 6.1; win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/53.0.2785.143 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: es-ES,es;q=0.8
Cookie: COUNTRY=NA%2C89.141.112.117; LAST_NEWS=1476292986
If-Modified-Since: wed, 12 Oct 2016 17:00:12 GMT
```

Respuesta/RESPONSE

131	5.929193	83.137.20.135	192.168.1.120	HTTP	974	HTTP/1.1 200 OK (text/html)
132	5.929202	192.168.1.120	83.137.20.135	TCP	54	50126->80 [ACK] Seq=484 Ack=5301 Win=65536 Len=0
133	5.929598	192.168.1.120	83.137.20.135	TCP	54	50126->80 [FIN, ACK] Seq=484 Ack=5301 Win=65536 Len=0
134	5.930468	83.137.20.135	192.168.1.120	TCP	60	80->50126 [FIN, ACK] Seq=5301 Ack=484 Win=15744 Len=0
135	5.930478	192.168.1.120	83.137.20.135	TCP	54	50126->80 [ACK] Seq=485 Ack=5302 Win=65536 Len=0
138	5.974181	83.137.20.135	192.168.1.120	TCP	60	80->50126 [ACK] Seq=5302 Ack=485 Win=15744 Len=0
Frame 131: 974 bytes on wire (7792 bits), 974 bytes captured (7792 bits) on interface 0						
Ethernet II, Src: CompalBr_60:50:42 (dc:53:7c:60:50:42), Dst: CadmusCo_26:58:55 (08:00:27:26:58:55)						
Internet Protocol Version 4, Src: 83.137.20.135, Dst: 192.168.1.120						
Transmission Control Protocol, Src Port: 80, Dst Port: 50126, Seq: 4381, Ack: 484, Len: 920						
[4 Reassembled TCP Segments (5300 bytes): #127(1460), #128(1460), #130(1460), #131(920)]						
Hypertext Transfer Protocol						
+ HTTP/1.1 200 OK\r\n						
Date: Wed, 12 Oct 2016 17:34:42 GMT\r\n						
Server: Apache\r\n						
X-Powered-By: PHP/5.4.45\r\n						
Last-Modified: Wed, 12 Oct 2016 17:10:12 GMT\r\n						
Content-Language: en\r\n						
X-Frame-Options: SAMEORIGIN\r\n						
Set-Cookie: LAST_NEWS=1476293682; expires=Thu, 12-Oct-2017 17:34:42 GMT; path=/; domain=.php.net\r\n						
Link: <http://php.net/index>; rel=shorturl\r\n						
Content-Encoding: gzip\r\n						
Vary: Accept-Encoding\r\n						
+ Content-Length: 4835\r\n						
Connection: close\r\n						
Content-Type: text/html; charset=utf-8\r\n						
\r\n						
[HTTP response 1/1]						
[Time since request: 0.074006000 seconds]						
[Request in frame: 117]						
Content-encoded entity body (gzip): 4835 bytes -> 22616 bytes						
File Data: 22616 bytes						
190	33 35 0d 0a 43 6f 6e 6e	65 63 74 69 6f 6e 3a 20	35..Conn	ection:		
1a0	63 6c 6f 73 65 0d 0a 43	6f 6e 74 65 6e 74 2d 54	close..C	ontent-T		
1b0	79 70 65 3a 20 74 65 78	74 2f 68 74 6d 6c 3b 20	ype: tex	t/html;		

Respuesta/RESPONSE

```
HTTP/1.1 200 OK
Date: Wed, 12 Oct 2016 17:34:42 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Last-Modified: Wed, 12 Oct 2016 17:10:12 GMT
Content-language: en
X-Frame-Options: SAMEORIGIN
Set-Cookie: LAST_NEWS=1476293682; expires=Thu, 12-Oct-2017 17:34:42 GMT; path=/;
domain=.php.net
Link: <http://php.net/index>; rel=shorturl
Content-Encoding: gzip
Vary: Accept-Encoding
Content-Length: 4835
Connection: close
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>PHP: Hypertext Preprocessor</title>

  <link rel="shortcut icon" href="http://nl3.php.net/favicon.ico">
  <link rel="search" type="application/opensearchdescription+xml" href="http://php.net/
phpnetimprovedsearch.ssr" title="Add PHP.net search">
```

HTTP 2.0

Los objetivos principales de HTTP/2 son **reducir la latencia** permitiendo una multiplexación completa de solicitudes y respuestas, minimizar la sobrecarga de protocolo mediante una compresión eficiente de campos de encabezados de HTTP y agregar compatibilidad con la priorización de solicitudes y push de servidor. Con el fin de implementar estos requisitos, existe serie de mejoras de protocolo, como nuevos mecanismos de control de flujo, manejo de errores y actualizaciones.

HTTP/2 no modifica la semántica de HTTP 1.1. Todos los conceptos centrales, como los métodos de HTTP, códigos de estado, URIs y campos de encabezados, permanecen vigentes.

<https://es.wikipedia.org/wiki/HTTP/2>

HTTP 2.0

- HTTP/2 es la evolución del protocolo de la capa de aplicación, HTTP. Se centra en hacer un uso más eficiente de los recursos de red. No cambia la característica fundamental de HTTP, la semántica. Todavía hay solicitudes, respuestas, cabeceras y todo los elementos típicos de HTTP/1.
- HTTP/2 es un **protocolo binario**, al contrario que HTTP 1.1 que es texto plano. La intención para HTTP 1.1 es que sea legible (por ejemplo capturando el tráfico de red) mientras que para HTTP/2 no.
- HTTP/2 es capaz de llevar **múltiples streams** de datos sobre la misma conexión TCP, evitando la clásica solicitud lenta "head-of-line blocking" de HTTP 1.1 y evitando generar múltiples conexiones TCP para cada solicitud/respuesta (KeepAlive parcheó el problema en HTTP 1.1 pero no lo resolvió completamente).

HTTPS

- El sistema HTTPS utiliza un cifrado basado en SSL (**Secure Socket Layers**) /TLS (***Transport Layer Security***) 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, password, etc.) 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
- Al usar HTTPS una página web adopta una codificación con certificado digital SSL con el que crea un canal cifrado más seguro para el tráfico de datos cliente (navegador) - servidor.
- HTTPS no evita que terceros puedan observar nuestras comunicaciones sino que crea un sistema de cifrado que hace que el mensaje solo pueda ser entendido por el destinatario, además de garantizar que el receptor de los datos es quien dice ser.

Diferencias con HTTP

- HTTPS trabaja por defecto por el puerto 443 TCP, y antes de enviar los datos realiza algunas acciones previas. Para hacer esta negociación, el cliente, envía al servidor las opciones de cifrado, compresión y versión de SSL junto con algunos bytes aleatorios.

El servidor, escoge las opciones de cifrado, compresión y versión de SSL entre las que ha ofertado el cliente y le envía su decisión y su certificado.

Ambos negocian la clave secreta llamada y usando esta clave y las opciones pactadas se envía la información encriptada de tal manera que de ser interceptada no se puede descifrar.

HTTPS

117	13.456178	216.58.210.131	192.168.1.120	QUIC	516 Payload (Encrypted), PKN: 110
118	13.484134	192.168.1.120	216.58.210.131	QUIC	80 Payload (Encrypted), PKN: 90, CID: 13773938175972065013
119	13.677988	192.168.1.120	216.58.210.131	QUIC	243 Payload (Encrypted), PKN: 91, CID: 13773938175972065013
120	13.713669	216.58.210.131	192.168.1.120	QUIC	72 Payload (Encrypted), PKN: 111
121	13.742877	216.58.210.131	192.168.1.120	QUIC	532 Payload (Encrypted), PKN: 112
122	13.768734	192.168.1.120	216.58.210.131	QUIC	80 Payload (Encrypted), PKN: 92, CID: 13773938175972065013
123	13.893741	192.168.1.120	8.8.8.8	DNS	71 Standard query 0x783d A www.bbva.es
124	13.905049	8.8.8.8	192.168.1.120	DNS	87 Standard query response 0x783d A www.bbva.es A 89.107.176.83
125	13.915314	192.168.1.120	89.107.176.83	TCP	66 50154-443 [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
126	13.923540	192.168.1.120	89.107.176.83	TCP	66 50155-443 [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
127	13.924869	89.107.176.83	192.168.1.120	TCP	62 443-50154 [SYN, ACK] Seq=0 Ack=1 win=4380 Len=0 MSS=1460 SACK_PERM=1
128	13.924904	192.168.1.120	89.107.176.83	TCP	54 50154-443 [ACK] Seq=1 Ack=1 win=64240 Len=0
129	13.934308	89.107.176.83	192.168.1.120	TCP	62 443-50155 [SYN, ACK] Seq=0 Ack=1 win=4380 Len=0 MSS=1460 SACK_PERM=1
130	13.934347	192.168.1.120	89.107.176.83	TCP	54 50155-443 [ACK] Seq=1 Ack=1 win=64240 Len=0
131	13.937732	192.168.1.120	89.107.176.83	TLSv1.2	246 Client Hello
132	13.938060	192.168.1.120	89.107.176.83	TLSv1.2	246 Client Hello
133	13.951759	89.107.176.83	192.168.1.120	TLSv1.2	1514 Server Hello
134	13.952223	89.107.176.83	192.168.1.120	TLSv1.2	1514 Server Hello
135	13.952380	89.107.176.83	192.168.1.120	TLSv1.2	1317 CertificateServer Hello Done
136	13.952380	89.107.176.83	192.168.1.120	TLS	54 50154-443 [ACK] Seq=1 Ack=1 win=64240 Len=0
Frame 123: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0					
Ethernet II, Src: CadmusCo_26:58:55 (08:00:27:26:58:55), Dst: CompalBr_60:50:42 (dc:53:7c:60:50:42)					
Internet Protocol Version 4, Src: 192.168.1.120, Dst: 8.8.8.8					
User Datagram Protocol, Src Port: 64666, Dst Port: 53					
Source Port: 64666					
Destination Port: 53					
Length: 37					
Checksum: 0xd266 [unverified]					
[Checksum Status: unverified]					
[Stream index: 3]					
Domain Name System (query)					

HTTPS

131	13.937732	192.168.1.120	89.107.176.83	TLSv1.2	246 Client Hello
132	13.938060	192.168.1.120	89.107.176.83	TLSv1.2	246 Client Hello
133	13.951759	89.107.176.83	192.168.1.120	TLSv1.2	1514 Server Hello
134	13.952223	89.107.176.83	192.168.1.120	TLSv1.2	1514 Server Hello
135	13.952380	89.107.176.83	192.168.1.120	TLSv1.2	1317 CertificateServer Hello Done
136	13.952400	192.168.1.120	89.107.176.83	TCP	54 50154 443 [ACK] Seq=192 Len=0 Win=0

Frame 131: 246 bytes on wire (1968 bits), 246 bytes captured (1968 bits) on interface 0

Ethernet II, Src: CadmusCo_26:58:55 (08:00:27:26:58:55), Dst: CompalBr_60:50:42 (dc:53:7c:60:50:42)

Internet Protocol Version 4, Src: 192.168.1.120, Dst: 89.107.176.83

Transmission Control Protocol, Src Port: 50154, Dst Port: 443, Seq: 1, Ack: 1, Len: 192

Secure Sockets Layer

- [-] TLSv1.2 Record Layer: Handshake Protocol: Client Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 187
 - [+] Handshake Protocol: Client Hello

El certificado del servidor

Un certificado de **clave pública** es un "documento" que certifica que el interlocutor (el servidor HTTPS en el caso de HTTPS) es quien realmente dice ser, esto se hace para evitar que un atacante pueda hacerse pasar por el servidor y recibir la comunicación segura en su lugar.

Estos certificados pueden generarse con herramientas cómo **OpenSSL** y para una mayor seguridad pueden ser firmados por una autoridad certificadora, por ejemplo:

VeriSign

Thawte

GoDaddy

GeoTrust

Aunque también puede ser auto-firmado, en este caso, se evita desembolsar la cantidad de dinero que cobran las autoridades certificadoras manteniendo el cifrado del canal de datos, pero el cliente no tendrá total seguridad de que la información está siendo enviada al servidor correcto.

Direcciones

- <http://es.kioskea.net/contents/internet/http.php3>
- [http://es.wikipedia.org/wiki/Hypertext Transfer Protocol](http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
- <http://www.htmlpoint.com/apache/10.htm>
- <http://www.bignosebird.com/apache/a5.shtml>
- <http://informatica.uv.es/iiguia/IST/Tema2.pdf>
- <http://es.ccm.net/contents/264-el-protocolo-http>
- <http://bibing.us.es/proyectos/abreproy/10967/fichero/Memoria+por+capitulos%252F5-Descripcion+del+protocolo+HTTP.pdf>
- <https://developers.google.com/web/fundamentals/performance/http2/?hl=es>