



Universidad rey Juan Carlos

Escuela Técnica Superior de Ingeniería de Telecomunicación

Sistemas Telemáticos

Practica 5

HTTP

Iván Moreno Martín

2º GITT

Contenido

1. Comunicación cliente-servidor HTTP.....	3
2. Diferentes tipos de respuesta de un Servidor	5
3. Formularios en HTTP	6
4. Cookies.....	10
4.1. Almacén de Cookies en el navegador Firefox	10
4.2. Envío de Cookies en un mensaje HTTP.....	11
5. Conexión a través de proxy HTTP	13
5.1 Caché en el proxy	15
6. Caché en el navegador	16
6.1. Sitio web ICANN.....	16
6.2. Sitio web IETF.....	16
6.3 Sitio web MECD	17

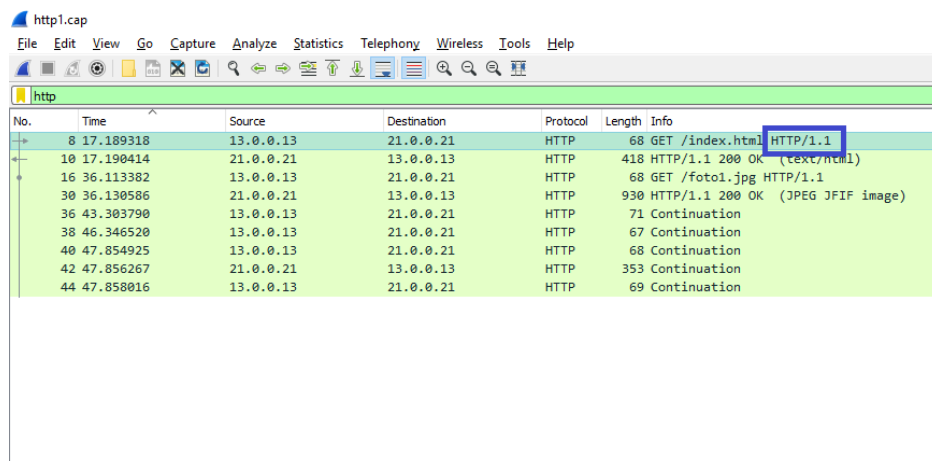
1. Comunicación cliente-servidor HTTP

1. Indica qué dirección IP es la de la máquina cliente HTTP y cuál la del servidor.

Para saber quién es quién en la conexión en este caso vamos a analizar las tramas http que vamos a filtrar en Wireshark. En este caso vemos que el cliente lanza una petición GET con origen 13.0.0.13 por tanto esta será su dirección IP. Por otro lado comprobamos que el destino es la dirección 21.0.0.21, esa será la dirección IP del servidor.

2. Indica qué versión HTTP están utilizando cliente/servidor

Como vemos en la siguiente imagen la versión de http que se está usando es la 1.1



No.	Time	Source	Destination	Protocol	Length	Info
8	17.189318	13.0.0.13	21.0.0.21	HTTP	68	GET /index.html HTTP/1.1
10	17.190414	21.0.0.21	13.0.0.13	HTTP	418	HTTP/1.1 200 OK (text/html)
16	36.113382	13.0.0.13	21.0.0.21	HTTP	68	GET /foto1.jpg HTTP/1.1
30	36.130586	21.0.0.21	13.0.0.13	HTTP	930	HTTP/1.1 200 OK (JPEG JFIF image)
36	43.303790	13.0.0.13	21.0.0.21	HTTP	71	Continuation
38	46.346520	13.0.0.13	21.0.0.21	HTTP	67	Continuation
40	47.854925	13.0.0.13	21.0.0.21	HTTP	68	Continuation
42	47.856267	21.0.0.21	13.0.0.13	HTTP	353	Continuation
44	47.858016	13.0.0.13	21.0.0.21	HTTP	69	Continuation

3. Indica si están utilizando conexiones persistentes o no persistentes y el número de conexiones que se ven en el fichero de captura.

Como estamos usando la versión 1.1 de http suponemos que las conexiones son persistentes. Con la versión 1.1 también podemos usar conexiones no persistentes pero en este caso el cliente incluye en las cabeceras **connection: close** y el servidor en su respuesta también incluirá en su cabecera **connection: close**. Como podemos ver en este caso en concreto no se observa que ni en la petición ni en la respuesta se incluya esta cabecera.

4. ¿Cuántas peticiones GET observas desde el cliente?

En este caso podemos observar dos peticiones. La primera de establecimiento de conexión y la segunda petición es de solicitud de un recurso, es este caso de foto1

5. ¿Cuántas URLs crees que ha escrito el usuario en el navegador para obtener dicha captura? ¿Cuáles? ¿Por qué?

En esta captura podemos ver que solo escribe una URL ya que la imagen se encuentra en el index.html

6. Fíjate en el contenido de la página index.html que se ha descargado el cliente. ¿Qué crees que ocurrirá cuando el navegador se haya descargado index.html?

Como el cliente no ha mandado ningún mensaje de cierre de conexión el servidor se queda a la escucha hasta que le llegue una nueva petición.

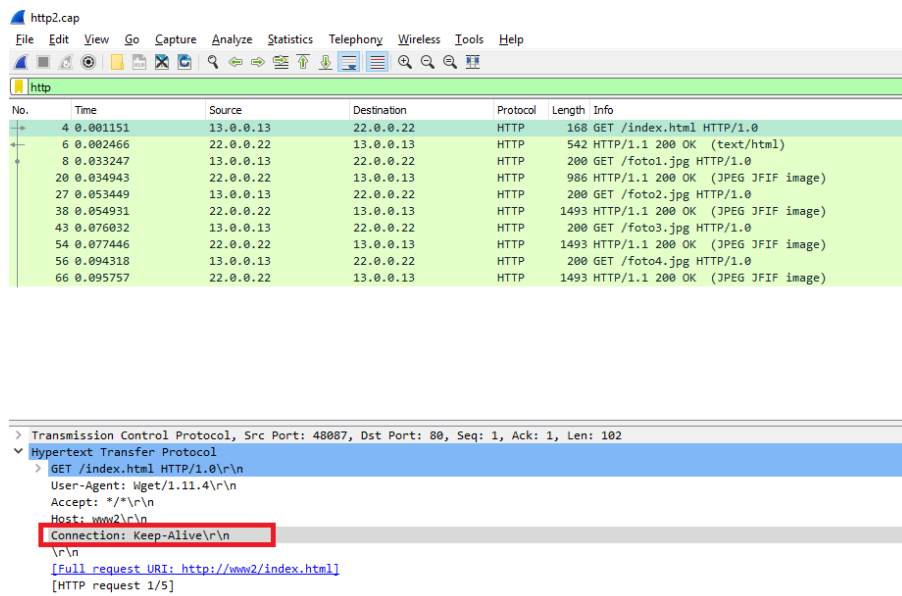
Abre la captura http2.cap y responde a las siguientes preguntas:

7. Indica qué versión HTTP están utilizando cliente/servidor

Como vemos la versión que se está usando en este caso es 1.0 por lo que podríamos pensar en un principio que la conexión podría ser no persistente.

8. Indica si están utilizando conexiones persistentes o no persistentes y el número de conexiones que se ven en el fichero de captura. ¿Por qué?

Para saber si estamos en un caso u otro comprobamos si existen mensajes KEEP-ALIVE en las cabeceras.

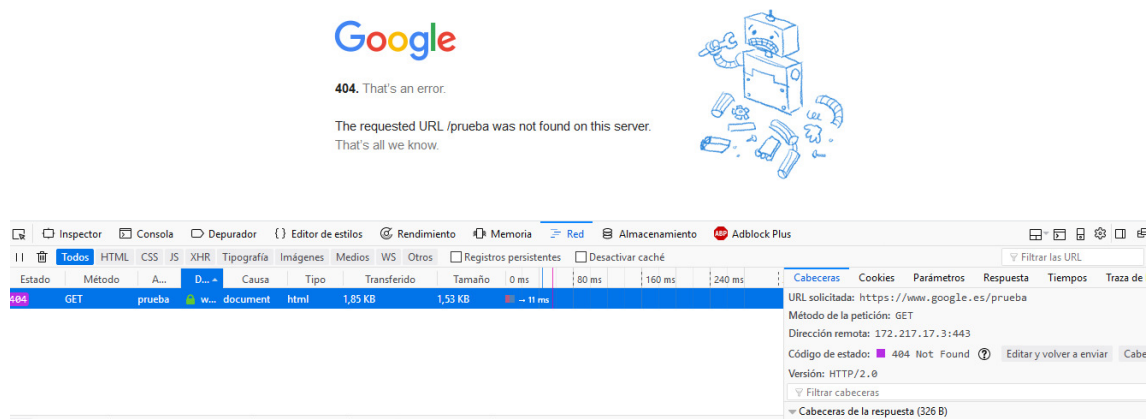


En este caso en concreto vemos que se está usando conexiones persistentes ya que en las solicitudes y respuestas se incluye la cabecera KEEP-ALIVE

2. Diferentes tipos de respuesta de un Servidor

1. Dentro de esa pestaña carga la página <http://www.google.es/prueba>. Selecciona dentro de las herramientas del desarrollador la petición GET y la pestaña Cabeceras. Fíjate en el campo Código de estado que indica el tipo de respuesta recibida y explica su contenido.

Como podemos observar en este caso al solicitar un recurso podemos ver una respuesta **404 page not found** (no existe el recurso que hemos solicitado). También podemos ver que la versión que se está usando es http versión 2.



2. En esa misma pestaña carga la página <http://www.wikipedia.com>. Explica que ocurre en la primera petición GET y a partir de las líneas de cabecera que ves en la respuesta, explica la segunda petición GET.

En la primera petición GET se manda un mensaje de redirección en ese caso es un **301 Moved permanently**, es decir se detecta Wikipedia.com y te redirige a Wikipedia.org.

En la siguiente petición GET se puede ver que hay una nueva solicitud esta vez con la URL Wikipedia.org en la cabecera, la respuesta a esa solicitud ahora es 200 OK con lo que nos quiere decir que se ha encontrado esa URL.

3. Formularios en HTTP

1. Indica qué tipo de conexiones HTTP utilizan el cliente y servidor e indica el número de conexiones entre cliente y servidor que aparecen en la captura.

Como podemos ver la primera solicitud http que lanza el cliente es con versión 1.0 lo que nos lleva a pensar que las conexiones no serán persistentes además no aparece ninguna cabecera con KEEP-ALIVE ni en la solicitud ni en la respuesta, además vemos en el servidor la cabecera **Connection: close**, lo que se confirma que son conexiones no persistentes.

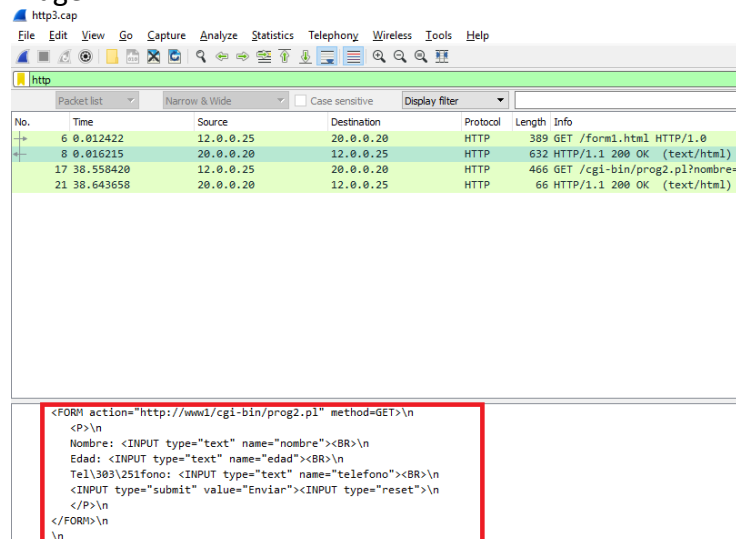
Si realizamos un filtrado para que nos aparezcan solo los paquetes HTTP vemos que en esta captura solo nos aparecen dos solicitudes y dos respuestas, además sabemos que son conexiones no persistentes, por lo que se puede deducir que son dos conexiones.

2. Busca en la captura el segmento donde el servidor le envía al cliente un formulario. Indica los nombres de los campos del formulario que rellenará el usuario.

El nombre de los campos que el cliente rellenará serán nombre, edad y teléfono

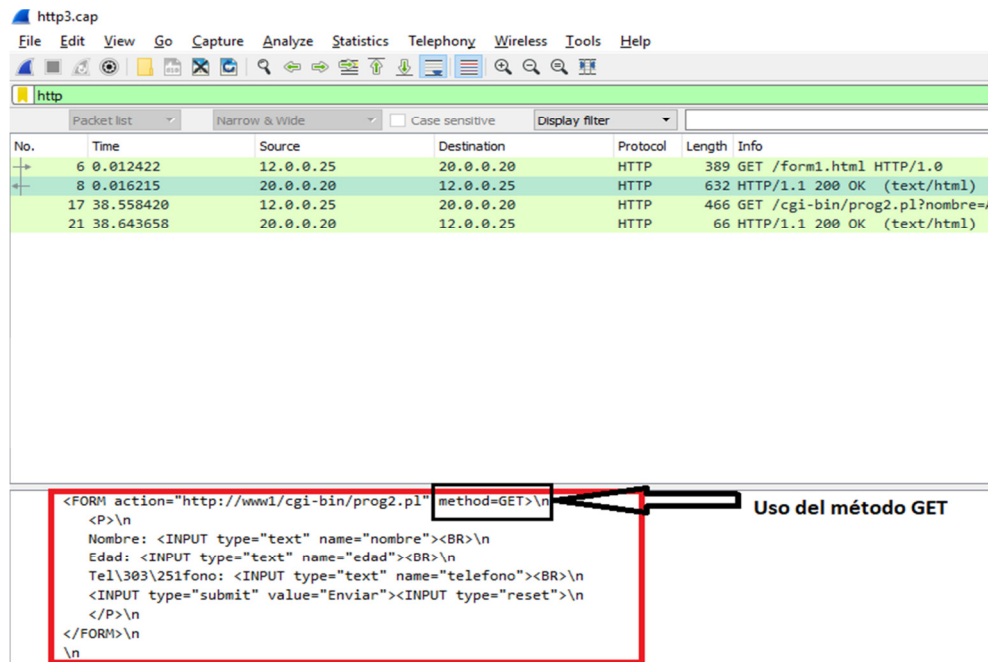
3. Indica si es el cliente o el servidor el que decide cómo debe enviar el cliente los datos del formulario (GET/POST). ¿Qué método están usando en este caso? ¿Cómo lo sabes?

En este caso será el servidor el que decide cómo se debe enviar los datos como vemos en la siguiente imagen:



También vemos que el método usado es GET, ya que para rellenar el formulario lo hará a través de la URL.

4. Busca en la captura el segmento donde el cliente le envía los datos del formulario al servidor y comprueba que se está realizando con el método GET



5. Fíjate cómo se llama el programa del servidor que va a recibir esos datos.

Como vemos en la imagen superior el programa se llama prog2.pl y aparece en el propio formulario.

6. ¿Dónde viajan los datos que el cliente le envía al servidor? ¿Cuáles son esos datos?

Los datos del cliente viajan en la URL y estos datos son como hemos indicado con anterioridad el nombre, la edad y el teléfono.

7. Indica qué cabecera es la que representa el tipo de contenido del mensaje que el cliente envía al servidor con los datos del formulario.

La cabecera que representa el tipo de contenido del mensaje es la siguiente:

Accept: text/html, text/plain, application/x-troff-man, application/x-tar, application/x-gtar, text/*, application/x-debian-package, audio/basic, */*;q=0.01

Abre la captura http4.cap y responde a las siguientes preguntas:

9. Busca en la captura el segmento donde el servidor le envía al cliente un formulario. Indica los nombres de los campos del formulario que rellenará el usuario.

Los campos del formulario que rellenará el cliente serán nombre, NIF y Edad

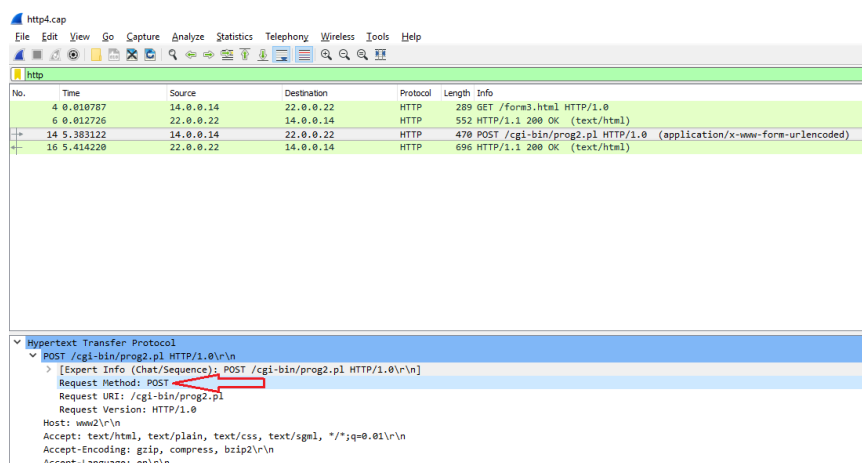
10. Indica si es el cliente o el servidor el que decide cómo debe enviar el cliente los datos del formulario (GET/POST). ¿Qué método están usando en este caso? ¿Cómo lo sabes?

Line-based text data: text/html (9 lines)

```
<FORM action="http://www2/cgi-bin/prog2.pl" method=POST>\n
<P>\n
Nombre: <INPUT type="text" name="nombre"><BR>\n
NIF: <INPUT type="text" name="nif"><BR>\n
Edad: <INPUT type="text" name="edad"><BR>\n
<INPUT type="submit" value="Enviar"><INPUT type="reset">\n
</P>\n
</FORM>\n
\n
```

Como en el caso anterior será el servidor el que decide cómo se enviarán esos datos. Para este caso se usará el método POST.

11. Busca en la captura el segmento donde el cliente le envía los datos del formulario al servidor y comprueba que se está realizando con el método POST.



12. Fíjate cómo se llama el programa del servidor que va a recibir esos datos.

El nombre del programa que va a recibir esos datos es prog2.pl

13. Indica qué cabecera es la que representa el tipo de contenido que el cliente envía al servidor y cuál es su valor.

La cabecera que representa el tipo de contenido que el cliente envía al servidor es:

Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01\r\n

14. Indica en qué parte del mensaje van los datos del formulario que el cliente le envía al servidor.

Los datos del formulario que el cliente manda al servidor van en el cuerpo del mensaje.

15. Explica si en este caso es necesario la cabecera Content-Length en el mensaje HTTP que el cliente envía al servidor con los datos del formulario. ¿Por qué?

El content-length es el tamaño en bytes del cuerpo del mensaje. Como estamos en la versión 1.0 las 16 cabeceras son opcionales. Por tanto no es obligatoria.

16. Observa si el servidor le manda alguna respuesta cuando recibe los datos del formulario del cliente. En caso afirmativo localiza el número de segmento y observa en las cabeceras HTTP: tipo de contenido, longitud y cuerpo del mensaje

El servidor manda una respuesta en el segmento 16, Content-Type: text/html. El contenido del mensaje es el siguiente:

```
<html>
<head><title>Resultado </title></head>
  <body>
    <h2>Hola Jaime, con NIF 1111, de 23 a\303\261os de edad </h2>
    Te he enviado tus datos en forma de cookies
  </body>
</html>
```

Es decir el cliente vería: Hola Jaime, con NIF 1111, de 23 años de edad

4. Cookies

4.1. Almacén de Cookies en el navegador Firefox

2. Pulsa sobre la pestaña "Cookies" para poder ver de forma más clara el contenido de las cookies. Copia los campos importantes. Fíjate que no hay fecha de expiración, eso quiere decir que la Cookie se eliminaría cuando se cierre el navegador.

Cookies de la petición:

```
__utma65738302.107415360.1526313316.1526313316.1526313316.1
__utmb65738302.3.10.1526313316
__utmt 1
__utmz65738302.1526313316.1.1.utmcsr=(direct)utmccn=(direct)utmcmd=(none)
```

ASPSESSIONIDCCSRBSCD: PNJOJAGAIQJKECAEKEGJOELG

Cabeceras de la petición

```
GET /resolucion.asp?resolucion=1366*768 HTTP/1.1
Host: www.ayto-fuenlabrada.es
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.ayto-fuenlabrada.es/
Cookie: __utma=65738302.107415360.1526313316.1526313316.1526313316.1;
__utmb=65738302.3.10.1526313316;
__utmz=65738302.1526313316.1.1.utmcsr=(direct)utmccn=(direct)utmcmd=(none);
__utmt=1; ASPSESSIONIDCCSRBSCD=PNJOJAGAIQJKECAEKEGJOELG
Connection: keep-alive
```

Cabeceras de la respuesta

```
HTTP/1.1 200 OK
Date: Mon, 14 May 2018 16:04:57 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Length: 0
Content-Type: text/html
Cache-control: private
Via: 1.1 192.168.101.71
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

4. Vuelve a la pestaña donde tienes cargada la página <http://www.ayto-fuenlabrada.es/> y pulsa sobre la segunda petición GET que aparece, observarás que en esa petición web se envían todas las cookies almacenadas en el navegador. Usa también la pestaña Cookies para poder ver mejor los valores que se envían.

```
GET /resolucion.asp?resolucion=1366*768 HTTP/1.1
Host: www.ayto-fuenlabrada.es
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.ayto-fuenlabrada.es/
Cookie: ASPSESSIONIDCCSRBSCD=PNJOJAGAIQJKECAEKEGJOELG; __utmc=65738302
Connection: keep-alive
```

4.2. Envío de Cookies en un mensaje HTTP

Abre la captura http5.cap y responde a las siguientes preguntas:

1. Indica qué cookies envía el servidor al cliente:

Set-Cookie: Nombre=Pepe Lozano; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT;
Domain=elcortebritanico.com; Path=/facturas;\r\n

Set-Cookie: Nif=22034J; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT;
Domain=elcortebritanico.com; Path=/facturas;\r\n

Set-Cookie: Edad=22; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT;
Domain=elcortebritanico.com; Path=/facturas;\r\n

Set-Cookie: Carrito=objeque-bienvenida; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT;
Domain=elcortebritanico.com; Path=/tienda;\r\n

Set-Cookie: Sesion=1003; Expires=Tuesday, 31-Dec-2030 23:12:40 GMT;
Domain=elcortebritanico.com; Path=/;\r\n

Es decir manda las cookies nombre, NIF, edad, Carrito, Sesion

2. Indica qué cookies enviaría el cliente al servidor cuando acceda a la página con la URL: <http://elcortebritanico/tienda/index.html>

Las cookies que envía el cliente son aquellas que cumplan estas tres condiciones:

- 1) Aún no han expirado
- 2) Son del mismo Domain (servidor) al que va a hacer la petición
- 3) La URL que se va a solicitar empieza por el PATH para el que es válida

3. ¿Y si el cliente accediera en el año 2025 a dicha URL?

- 1) Comprobamos que las cookies no caducan hasta el 31 de diciembre de 2030.
- 2) Todas pertenecen a un mismo Domain, en este caso a elcortebritanico.com
- 3) Y también cumple la condición de que la URL que se va a solicitar empieza por el PATH para el que es válida.

Por tanto el cliente mandará todas sus cookies.

4. ¿Y si el cliente accediera en el año 2035 a dicha URL?

Para este caso existen dos posibilidades. La primera de ellas es que el cliente se conectase antes del martes del 31 de diciembre a las 23:12:40 GMT. En ese caso se enviarían todas las cookies. En caso contrario las cookies habrían expirado por lo que el servidor tendría que mandar nuevas cookies al cliente con una nueva fecha de expiración.

Abre la captura http6.cap y responde a las siguientes preguntas:

5. Indica qué cookies el cliente está enviando al servidor.

En este caso podemos observar en la captura que se envían dos cookies, la primera con la versión y la segunda se manda con dos atributos en concreto con el nombre y con el NIF.

6. ¿Por qué crees que le envía dichas cookies?

Se mandan esas cookies porque ha existido una conexión previa con el servidor.

7. Escribe un ejemplo de las posibles cabeceras que le habría enviado dicho servidor al cliente previamente.

Set-Cookie: Nombre = Andres; expires= ...; domain=/index.html; path=/
Set-Cookie: NIF = 123456789A; expires= ...; domain =/index.html ;path=/;

8. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: <http://www2/dir1/dir2/index.html> enviaría esas cookies, más o menos?

Se mandarían las mismas cookies siempre y cuando no hayan expirado, además estamos accediendo a la misma página, aunque a un recurso distinto dentro de dir1

9. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: <http://www2/index.html> enviaría esas cookies, más o menos?

Se mandarían las mismas cookies aunque el recurso esta propiamente en la página principal

10. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: <http://www/index.html> enviaría esas cookies, más o menos?

En este caso no porque estamos accediendo a una URL distinta, y quizás sean otro tipo de cookies.

5. Conexión a través de proxy HTTP

Abre la captura http7.cap y responde a las siguientes preguntas:

1. Indica qué dirección IP es el cliente, el proxy y el servidor final.

El cliente siempre se conecta con el proxy, después el proxy se conecta con el servidor final. El camino a la inversa es servidor final contesta al proxy y el proxy manda la respuesta al cliente. Por tanto

La dirección del cliente es: 13.0.0.13

La dirección del proxy es: 23.0.0.23

La dirección del servidor final es: 22.0.0.22

2. ¿Qué diferencia la petición HTTP que realiza el cliente de la petición que realiza el proxy?

Las peticiones a un proxy se distinguen porque incluyen la URL completa en la primera línea del mensaje de petición. Por otro lado las peticiones de un proxy a un servidor no incluyen la URL completa

3. Identifica el nombre de la máquina donde se encuentra el servidor HTTP.

Para saber el nombre de la máquina donde se encuentra el servidor HTTP comprobamos el HOST. En esta captura HOST = www2

4. ¿Se puede saber de la petición que realiza el proxy que dicho proxy tiene almacenada en su caché esa página?

Si podemos ver que el servidor le manda un mensaje 304 Not Modified al proxy podemos deducir que el proxy hace una solicitud para comprobar si ha habido algún cambio en el servidor, es decir, para ver si ha que realizar algún cambio en su caché.

Abre la captura http8.cap y responde a las siguientes preguntas:

5. Indica qué tipo de conexiones HTTP utilizan el cliente y servidor e indica el número de conexiones entre cliente y servidor que aparecen en la captura.

La versión que está usando el cliente es 1.0, en su mensaje nos viaja ningún KEEP_ALIVE por lo que deducimos que es una conexión no persistente.

Por otro lado comprobamos que el servidor usa también la versión 1.0 de HTTP: Por ser conexiones no persistentes, en este caso existen dos conexiones entre cliente y servidor, una solicitud de página del servidor y una solicitud de un recurso (una imagen) que se encuentra en el servidor.

6. Explica qué es lo que se está descargando el cliente del servidor HTTP y cuantos objetos se descarga.

El cliente está descargando una imagen del servidor HTTP. En este caso te descargaría la página y la imagen solicitada.

7. Observa en las cabeceras HTTP el tipo de contenido de cada uno de los objetos.

El primero es un objeto es Content-Type = text/html
El segundo objeto es Content-Type = imagen/jpeg

8. ¿Podrás saber si los paquetes capturados se corresponde a la comunicación entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final HTTP? ¿Por qué?

En esta captura podemos ver una comunicación entre cliente y servidor proxy porque la primera solicitud del cliente es con la URL completa.

9. Sabiendo que la comunicación se ha realizado a través de un proxy HTTP, mira las cabeceras HTTP que envía dicho proxy para ver si en ellas existe alguna que muestre cuál es su nombre.

Para saber el nombre del proxy comprobamos en la solicitud del proxy que aparece como Via: www3 en el puerto 8080

Abre la captura http9.cap y responde a las siguientes preguntas:

10. ¿Podrás saber si los paquetes capturados se corresponde a la comunicación entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final HTTP? ¿Por qué?

En la captura http9 vemos que el GET del cliente incluye la URL completa por lo que podemos deducir que es la comunicación con un proxy. Por otro lado el siguiente GET del cliente envía la URI completa lo que confirma que estamos en comunicación entre cliente y servidor. Y por último vemos que en la cabecera HTTP del servidor aparece **Via: www1** que quiere decir que el contenido viene del servidor www1.

5.1 Caché en el proxy

Estudia las capturas **http10.cap** y **http11.cap** y responde a las siguientes preguntas:

2. Indica cuales son las direcciones IP del cliente, proxy y servidor web. ¿Cómo lo sabes?

En la captura **http10.cap** las direcciones:

Dirección IP cliente → 12.0.0.100

Dirección IP servidor → 12.0.0.1

Para este caso en principio parece que es una conexión entre un cliente y un servidor normal. Pero comprobamos que en la primera solicitud el cliente está mandando una URL completa por lo que sería una comunicación entre un cliente y un servidor proxy

GET <http://14.0.0.100/index.html> HTTP/1.0\r\n

Además si comprobamos la respuesta del servidor vemos que aparece:

VIA: 1.0 r1:8080

En la captura **http11.cap** las direcciones:

Dirección IP Proxy → 11.0.0.1

Dirección IP servidor → 14.0.0.100

En este caso se ve claramente que es una conversación entre un Proxy porque en la respuesta del servidor aparece en el mensaje un 304 Not Modify.

3. Explica qué es lo que ocurre en estas capturas.

En la primera captura (**http10.cap**) se accede a una URL a través de un proxy. A continuación se solicita unos recursos (imágenes) que están cacheado en el proxy y este se los manda al cliente.

En la segunda captura se ve una comunicación entre un servidor Proxy y un servidor. Esta comunicación se realiza para comprobar si hay alguna modificación en el recurso solicitado, a continuación el servidor le manda al Proxy un mensaje de que ese recurso solicitado no ha sufrido modificación alguna.

4. Localiza los campos relevantes con respecto al tratamiento de caché que incluye en las líneas de cabecera el servidor. ¿Qué significan?

5. Explica qué ocurre en la segunda consulta que realiza el cliente.

6. Viendo los paquetes 14 y 16 de la captura http10.cap indica cómo se puede saber que el contenido proviene de una caché.

En el segmento 14 el servidor proxy recibe una petición de un recurso que no sabe si ha expirado concretamente una imagen entonces se manda una petición al servidor final para comprobar si ese recurso ha expirado y como no ha expirado el servidor manda el recurso para que el proxy se lo mande al cliente, que es lo que ocurre en el segmento 16. Por lo que se puede deducir que el Proxy lo tiene guardado en la caché y además no ha sufrido modificación alguna.

6. Caché en el navegador

6.1. Sitio web ICANN

1. Despliega la pestaña de las cabeceras de la respuesta e indica qué tipo de caché se permite.

En este caso en el parámetro Cache-Control → no caché, no store. En este caso por un lado nos dice que si el recurso se almacenara no podría ser servido desde la caché sin revalidarlo previamente y además el recurso no puede ser almacenado.

2. A la vista de los resultados, ¿qué crees que ocurrirá si se recarga la página en el navegador?

Si recargamos la página en el navegador se tendría que revalidar de nuevo. Es decir como estamos en una conexión persistente debería salirnos lo mismo.

6.2. Sitio web IETF

4. Selecciona la petición GET / y observa en las líneas de cabecera de respuesta qué tipo de caché se permite.

Observamos las líneas de cabecera de respuesta y vemos el parámetro Cache-Control:

Cache-Control: public, max-age = 14400

5. Indica cuándo expirará el recurso en la caché. Relaciona los valores de las cabeceras: Date, Expires y Cache-Control.

Date: Tue, 15 May 2018 10:15:01 GMT
Expires: Tue, 15 May 2018 14:15:01 GMT
Cache-Control: public, max-age=14400

Si convertimos los segundos de max-age obtenemos que el recurso expirara en 4 días, que coincide con el plazo de expiración de Date y Expires ya que ambos tienen el mismo valor.

6. A la vista de los resultados ¿qué crees que ocurrirá si desde otra pestaña se solicita la misma URL?

Si lo hacemos antes del plazo de expiración no se realiza ninguna petición porque el recurso se encuentra en caché, si por otro lado esto mismo lo realizamos cuando ha pasado el plazo de expiración entonces se generará una nueva petición.

7. Abre otra pestaña en el navegador, selecciona de nuevo en las herramientas desarrollador web! alternar herramientas. Teclea en esta nueva pestaña la misma URL: <http://www.ietf.org>. Selecciona de nuevo la petición GET /. Al posicionar el ratón sobre el nombre de la petición GET, observarás que se muestra que el contenido estaba cacheado (fíjate que ahora ya no aparece un círculo verde al lado de GET /, en su lugar hay un círculo gris). Fíjate en las líneas de cabecera: Date, Expires y Last-modified y explica si han cambiado y por qué.

Comprobamos que si han cambiado tanto Date como Expires

Date → Tue, 15 May 2018 10:32:05 GMT
Expires → Tue, 15 May 2018 14:32:05 GMT

El parámetro Last-Modify no debería variar porque en un corto plazo de tiempo no debería haber ninguna modificación en el servidor de la página aunque podría pasar. Por otro lado los otros dos parámetros si cambian porque la fecha (hora en este caso) de acceso al recurso es distinto.

6.3 Sitio web MECD

9. Observa las líneas de cabecera de la respuesta e indica qué tipo de caché se permite.

Como podemos observar acepta cualquier tipo de caché.

Cache-Control: max-age=600, public

10. Indica cuándo expirará el recurso en la caché. Relaciona los valores de las cabeceras: Date, Expires y Cache-Control.

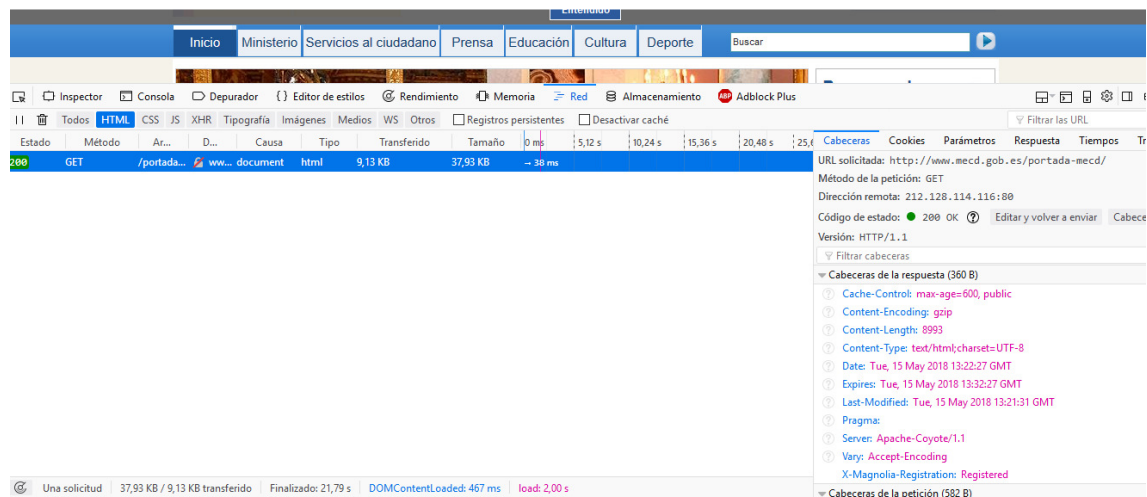
Date: Tue, 15 May 2018 11:08:47 GMT
Expires: Tue, 15 May 2018 11:18:47 GMT
Cache-Control: max-age=600

11. Abre una nueva pestaña del navegador con las herramientas del desarrollador visibles. Indica qué ocurre si tecleas la misma URL.

En este caso no se manda ninguna petición porque no ha caducado todavía si pasamos el plazo seguramente se genera una nueva petición con un nuevo plazo de expiración.

12. Espera el tiempo necesario para que la información caduque en la caché y abre una nueva pestaña con las herramientas del desarrollador visibles. Explica qué sucede cuando vuelves a teclear esta URL.

Como observamos en la imagen se hace una nueva solicitud de la página



13. ¿Se está usando validación fuerte o débil?

En este ejemplo y como vemos en la imagen anterior no tenemos ETAG con lo que tendríamos validación fuerte, pero si que podemos ver que tenemos Last-Modify por tanto concluimos que estamos ante una validación débil.