Activitat sm8act01: Breu estudi del protocol HTTP

A) Busca informació i respon les següents preguntes

1. Quin és el número i tipus de port utilitzat per defecte per un servidor web?. A on trobarem aquesta informació?.

   - El número i tipus de port és el 80/tcp

   - Dins del camp Destination Port d'un segment TCP si ho envia un programa client (Firefox), o del camp Source Port d'un segment TCP si ho envia un progrma servidor (Apache).

2. Quin són els documents dins dels quals està definit el funcionament del protocol HTTP. Indica qui crea aquests document. el tipus i número dels documents.

   - Els document RFC7230 a RFC7240 defineixen el funcionament del protocol HTTP/1.1 i deixen obsolet l'antic document RFC 2616.

   - Posteriorment, van apareixer els document 9110 i 9112 que deixen obsolets:
     - Porcions del document RFC7230.
     - Els documents RFC7231, RFC7232, RFC7233 i RFC7235

3. Respon les següents preguntes:

   a) Que significa que el protocol HTTP sigui un protocol client-servidor?

   Significa que el protocol HTTP segueix el model general de peticions i respostes entre un client i un servidor que opera intercanviant missatges HTTP. Per tant, el funcionament consisteix en una màquina client que emet una petició dirigida a un servidor per accedir a un recurs. Aquest servidor, si té accés a aquest recurs, genera una resposta que és enviada al client.

   b) Què significa que HTTP és un protocol sense estat (stateless)?

   El sistema no té memòria de les peticions realitzades anteriorment i cada cop que s'inicia una de nova, encara que sigui idèntica a una més antiga realitzada prèviament, es torna tractar novament en la seva totalitat. Així doncs, les peticions i respostes anteriors no tenen cap influència en les posteriors. És com si mai s'haguessin fet.

4. Respon les següents preguntes:

   a) Quines són les 4 parts d'un missatge HTTP?

   - La línia inicial, que pot ser de petició o d'estat
   - La capçalera
   - La línia en blanc de finalització de la capçalera
   - El cos (body) del missatge

   b) Quina és la informació que pots trobar a la línia inicial d'un missatge HTTP?. Quins són els 2 tipus de línies inicials existents?. Quin tipus de programa cadascun dels tipus de línies incials?.

   Es pot trobar una línia de petició si el missatge ha estat generat per un programa client o una línia d'estat si ha estat generat per un programa servidor.

   c) Com comença una lína de petició?. I una lína d'estat?
   - Petició: GET/POST/PUT/DELETE + Recurs demanat + HTTP/1.1
   - Estat: HTTP/1.1 + Codi de resposta(2xx/3xx/4xx/5xx) + Missatge resposta (OK, Not Found,..)

d) Què és la capçalera HTTP?. En quines parts està dividida?

- La capçalera d'un missatge HTTP de petició o resposta passa informació addicional i metadades a l'altra banda de la comunicació sobre la petició o resposta que s'està realitzant. Per exemple, un client pot enviar informació a un servidor indicant quina és la llengua prioritària en la resposta en el cas que el rescurs demanat al servidor es trobi disponible en més d'una llengua.
- La capçalaera d'un missatge HTTP estarà dividida en camps de capçalera (headers). Per exemple, el camp Accept-Language ho enviaria un client al servidor per indicar a quina llengua vol donar prioritat en la resposta. Els camps de capçalera són case-sensitive, tenen un nom i un valor. Hi ha un camp per cada línia de la capçalera HTTP. Es separa el nom del valor amb : , i cada camp finalitza amb <CR><LF>.

e) Quina informació dóna el camp de capçalera User-Agent?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?.
- El camp User-Agent és un camp que trobarem dins d'un missatge de petició.
- El camp User-Agent permet enviar informació que identifica l'aplicació (versió i fabricant) i el sistema operatiu sore el qual s'executa l'aplicació que està fent la petició a un servidor. Generalment serà un navegador però, també potser qualsevol altre programa que s'utilitzi per accedir a un servei web.

f) Quina informació dóna el camp de capçalera Host?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?
- El camp Host ens el trobarem dins d'un missatge de petició.
- Indica el nom de domini i port del host que té el servei web al qual volem accedir. El port per defecte si no s'inclou és el 80/tcp per http i 443/tcp per https.
- És un camp especialment útil quan una mateixa màquina servidora hostatja webs de diversos dominis. Quan això passa, diversos dominis tenen una mateixa IP però el camp host permet indicar al servidor a quina web en concret volen accedir.

g) Quina informació dóna el camp de capçalera Connection?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?.
- Ens el trobarem tant dins d'un missatge de petició com d'un de resposta.
- La capçalera Connection controla si la connexió de xarxa roman oberta després que finalitzi la transacció actual. Si el valor enviat és keep-alive, la connexió és persistent i no es tanca, cosa que permet sol·licituds posteriors al mateix servidor a la mateixa connexió. Això evita el procés de tancament de la connexió i haver de tornar a començar el procés d'obrir la connexió una altra vegada si ens cal enviar una nova petició.

h) Quina informació dóna el camp de capçalera Server?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?.
- El camp Server ens el trobarem dins d'un missatge de resposta.
-La capçalera de resposta del servidor HTTP descriu el sistema opertiu, programari i versió utilitzats per gestionar una sol·licitud i generar una resposta. Per exemple, un valor típic és "Apache/2.4.1 (Unix)" indicant que la resposta ha estat generada per un servidor web Apache.
- Els benificis d'aquest camp són fer anàlisi de les connexions i trobar problemes d'interconnexió o que els clients pugin evitar problemes associats a les limitacions d'un determinat servidor o d'alguna de les seves versions.

i) Quina informació dóna el camp de capçalera Strict-Transport-Security?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?.
- La capçalera HTTP Strict-Transport-Security (sovint abreujada com a HSTS) és de resposta.
- Informa els navegadors que només s'ha d'accedir a l'amfitrió mitjançant HTTPS i que qualsevol intent futur d'accedir-hi mitjançant HTTP s'ha d'actualitzar automàticament a HTTPS i que en el futur no permetrà errors de connexió segura, com ara, utilitzar un certificat no vàlid.
- HSTS identifica un amfitrió només pel seu nom de domini.

j) Quina informació dóna el camp de capçalera Date?. Aquest camp el trobaràs dins d'un missatge de petició o de resposta?.
- Date és una capçalera de resposta i de petició
- Indica lal data i horaen la qual es va crear el missatge enviat.

5. Indica de quina manera el client faria una petició de la pàgina estàtica index3.html que és troba a la carpeta "exam" que penja de l'arrel de l'arbre de directoris del servidor web. S'ha d'indicar mètode, fitxer, protocol i versió del protocol.

GET /exam/index3.html HTTP/1.1
Mètode: GET, fitxer: /exam/index3.html i protocol HTTP versió 1.1

6. Indica l'esquema URI per accedir a un fitxer que s'anomena ex1m08uf1.pdf que es troba a la carpeta /home/daw2/examen d'un servidor FTP al qual s'ha d'accedir amb el login daw i password m08uf1pr1. El nom del servidor és ftp.fjeclot.edu i escolta pel port 21.

ftp://daw:m08uf1pr1@ftp.fjeclot.edu:21/home/daw2/examen/ex1m08uf1.pdf

7. Indica quines són les diferències existents entre utilitzar un mètode POST i un mètode PUT
- El mètode POST s'utilitza bàsicament per crear nous recursos dins del servidor (creació d'un nou fitxer, creació d'un nou registre d'un base de dades, etc...).
- El codi de la resposta del servidor normalment serà un 201 si s'ha creat un nou recurs, però també pot ser un 200 per dir al client que posi la resposta a la caché, o un 303 el resultat de processar la petició dóna com a resultat un recurs ja existent. Podria enviar també 206, 304 o 416.
- El mètode POST no és idempotent. NO es garantitza que si repeteixo més d'una vegada la petició no es generin tants recursos com peticions s'han fet.
- El mètode PUT s'utilitza bàsicament per actualitzar completament un recurs dins del servidor (actualitzar un registre d'una base de dades, etc...).
- El codi de la resposta del servidor normalment serà un 200 o 204 si s'ha actualitzat el recurs. Hi ha altres respostes possibles per errors: 400, 409, 415.
- El mètode PUT és idempotent. Es garantitza que si repeteixo més d'una vegada la petició no es generin tants recursos com peticions s'han fet.
- Tot i que aquesta pregunta és sobre les diferències, és interessant remarcar que tots 2 tenen en comú que les dades s'envien dins del cos del missatge i el programa dins del servidor al quan volem accedir s'envien dins de la línia d'estat.

8. Indica quins són els grups de missatges diferents que pot enviar un servidor al client i quins són els propòsits de cadascun d'ells.

Un servidor pot enviar 5 tipus de missatge a un client. La informació del tipus de missatge es troba dins d'un codi numèric que es troba al principi de línia d'estat (la línia inicial en un servidor) del misstage de resposta del servidor. D'acord amb el document RFC9110– Capítol 15 el 5 de grups o tipus de missatges són els següents:

1xx --> Informació: La petició ha estat rebuda i el procés s'està duent a terme.
2xx --> Èxit (o sigui, ha funcionat correctament): La petició s'ha rebut correctament, s'ha pogut entendre i s'ha acceptat.
3xx --> Redireccionament: L'User-Agent (o sigui, el client que en general és el navegador) ha de fer una sèrie d'accions addicionals per completar la petició. Per exemple, si la pàgina web demanada a canviat de servidor, el client rep un missatge 301 i una nova URI. El client enviarà automàticament al nou servidor.
4xx --> Error del client: La petició no s'ha fet correctament, o el recurs no existeix, o no es té autorització per accedir-hi, etc..
5xx --> Error del servidor: El servidor és conscient de que ha comès un error o és incapaç de dur a terme el mètode demanat. Per exemple, el servidor enviarà el el codi 501 si no té suport (o no reconeix) el mètode que el client ha demanat en la línia de petició, o el 505 si el servidor no dóna suport per una determinada versió del protocol HTTP que el client ha demanat en la línia de petició.

9. Indica quins mètodes farian que el servidor respongués amb el missatge 201.

Qualsevol mètode que serveixi per crear nous recursos dins del servidor, o sigui, que bàsicament serà una resposta a un petició utilitzant el mètode POST. Estrictament parlant, d'acord amb el document RFC 9110 – Capítol 9 el mètode PUT també pot dona com a resposta un 201 però pel moment, ens limitarem a l'ús general d'aquests 2 mètodes i en general, POST és per crear i PUT per actualitzar.

10. Respon amb verdader o fals les següents afirmacions:

a) El mètode POST és menys segur perquè envia dades dins de la URL. Raona la resposta.
FALSA. Tot i que el mètode POST no és segur i el GET si que ho és, la raó per la qual un mètode és segur no té res a veure a com s'envien les dades dins del missatge HTTP. A més a més, POST envia les dades dins del cos del misstge no dins de la URL a la línia de petició.

b) Les peticions de tipus POST no poden ser desades a les adreces d'interès (bookmark)
VERITAT. Les respostes a una petició POST no es desen a les adreces d'interés perquè d'acrod amb les especificacions (documents RFC) del protocol HTTP 1.0 i 1.1, l'aplicació no té cap manera de saber si el servidor retornarà la mateixa resposta a la mateixa petició en el futur.

c) Una petició de tipus POST envia les dades dins del cos del missatge
VERITAT. Així està especificat als documents RFC que expliquen el funcionament del protocol HTTP.

d) Les peticions de tipus GET no admeten qualsevol tipus de dades (texte, imatge, video, etc..)
FALS. El mètode GET envia les dades dins de la línia de petició juntament amb l'URI. La línia de petició treballa amb caràcters USASCII d'acord amb els documents RFC que han de ser interpretats pel servidor i l'aplicació web. També, d'acord amb aquests documents existeixen restriccions en el format de l'URI que un format d'àudio, vídeo o imatge no poden assegurar que es compliran. També poden haver problemes amb el límit de bits que accepta un servidor com a part de l'URI de la línia de petició. De manera que el mètode GET no s'hauria d'utilitzar per enviar dades que no siguin ASCII.

11. Quin és el significat de la següen resposta d'un servidor:

```
HTTP/1.1 302 Found
Location: http://www.example.com/test/index2.php
```

El recurs ha canviat de localització temporalment. La nova URI és aquella indicada al camp Location. Si l'User-Agent ho pot fer, hauria de fer una nova petició a la nova localització.

B) Peticions i respostes HTTP treballant amb Telnet i wireshark

1. Posa en marxa **Wireshark** i filtra el contingut de les captures de manera que només es mostrin els continguts dels missatges **HTTP**. A continuació, estableix una connexió al port **80/tcp** del servidor **www.collados.org** utilitzant **telnet** de la següent manera:

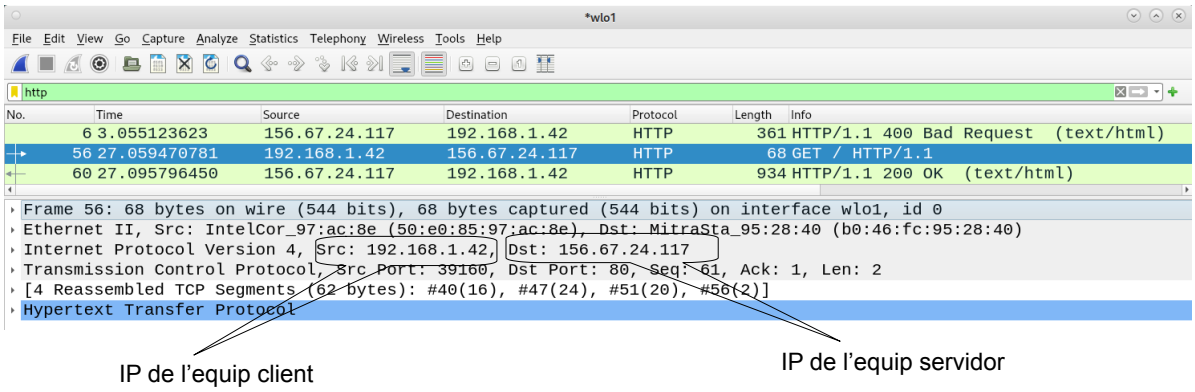
   telnet www.collados.org 80

   Ara, aprofita la connexió per envia el següent missatge **HTTP** des del teu equip al servidor:

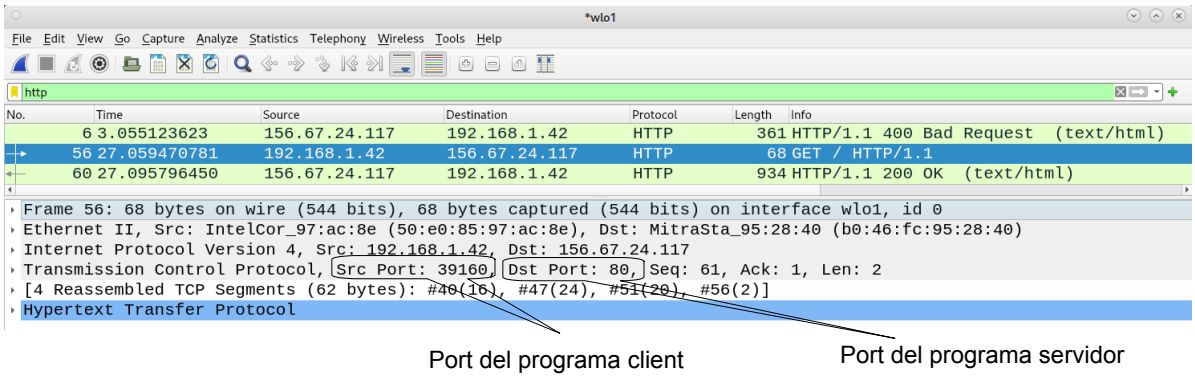
   ```
   GET  /  HTTP/1.1
   Host:  www.collados.org
   User-Agent:  telnet
   ```

   Mostra el resultat obtingut amb **Wireshark**, i troba:

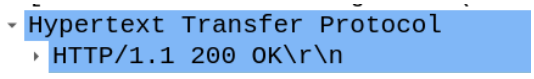
   - L'adreça IP del teu ordinador i la del servidor dins de la capçalera IP del missatge enviat pel programa telnet.



   -El port (tipus i valor) utilitzat pel programa telnet i pel servidor web



   - Quin codi de resposta dóna la capçalera HTTP enviada pel servidor? Què significa?



   El codi de resposta és el 200 OK i això vol dir que al recurs demanat ha estat trobat i s'envia com a a part del cos del missatge HTTP.

- Que indica la capçalera *Server* de la resposta donada pel servidor? Explica el resultat que has obtingut.

Dins de la resposta del servidor, trobarem el camp server amb el valor nginx:

```
Server: nginx\r\n
```

El header de resposta Server té informació sobre el programari que utilitza el servidor per manegar la petició realitzada pel client. Aquesta informació pot ser utilitzada per l'User-Agent per millorar les comunicacions entre els dos programes com diu la documentació RFC. En aquest cas és el servidor gws (Google Web service), el servidor propi de Google.

- Que indica la capçalera *Content-Type* de la resposta donada pel servidor? Explica el resultat que has obtingut.

Dins de la resposta del servidor, trobarem el camp server amb el valor gws:

```
Content-Type: text/html\r\n
```

El header de resposta Content-Type té informació sobre el tipus i codificació de les dades que es troben dins del cos del missatge enviat pel servidor. En aquest cas, ens indica que s'està enviant text i que aquest text és un codi HTML.

b) Comprova si la connexió es tanca immediatament després de rebre la resposta del servidor. Hi ha alguna manera de tancar la connexió?.

```
▸ Line-based text data: text/html (6 lines)
    <HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">\n
    <TITLE>301 Moved</TITLE></HEAD><BODY>\n
    <H1>301 Moved</H1>\n
    The document has moved\n
    <A HREF="http://www.google.com/">here</A>.\r\n
    </BODY></HTML>\r\n
```

No s'observa el missatge *Connection close by foreing host* de manera que la connexó no es tanca

La connexió no es tanca immediatament perquè no s'observa el missatge Connection close by foreing host al final del cos del missatge (després del codi HTML).. Això vol dir que des del terminal tindríem l'oportunitat de generar un nou missatge HTTP.

Per tancar la connexió podríem:

- Fer noves peticions afegint el campm Connection amb el valor close.
- Enviar una petició errònia fent per exemple Ctrl-D o escrivint qualsevol cosa que no sigui protocol HTTP. El resultat seria una resposta del tipus 400 i un tancament de la connexió.
- Esperar a que es tanqui la connexió per part del servidor de manera automàtica després de passar un temps.

2. Estableix novament una connexió via telnet al port **80/tcp** de **www.collados.org** i envia el següent missatge **HTTP**:

```
GET / HTTP/1.1
Host: www.collados.prg
User-Agent: telnet
Connection: close
```

a) Comprova des de telnet si la connexió es tanca immediatament després de rebre la resposta del servidor. Per què?

```
collados@sh4:~$ telnet www.collados.org 80
Trying 156.67.24.117...
Connected to www.collados.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.collados.org
User-Agent: telnet
Connection: close

HTTP/1.1 200 OK
Server: nginx
Date: Tue, 20 Oct 2026 10:41:22 GMT
Content-Type: text/html
Content-Length: 1931
Connection: close
Strict-Transport-Security: max-age=31536000; includeSubdomains;preload
Last-Modified: Tue, 22 Sep 2026 10:58:55 GMT
Accept-Ranges: bytes
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="ca">
  <head>
        <meta http-equiv="content-type" content="text/html; charset=windows-1252">
    <title>P□gina Inicial del domini collados.org</title>
  </head>
  <body style="background-color: silver;">
        <font face="Arial">
                <p align="left"><font size="+3"><b>P□gina inicial de www.collados.org</b></font></p>
        <b><font size="+1">Cicles Formatius de Grau Superior (ASIX/DAW)</font></b><br>
        <b><u>ASIX1-DAW1</u></b><br>
        <a href="http://www.collados.org/asix1/index_asix1.html">ASIX1/DAW1/DAM1-SM1/EH1(Open-source operating systems/Linux Lab) </a><br>
        <b><u>ASIX2</u></b><br>
        <a href="http://www.collados.org/asix2/index_asix2_eh3.html">ASIX2-EH3 (Hardware Lab)</a><br>
        <!--<a href="http://www.collados.org/asix2/index_asix2_sm5.html">ASIX2-SM5 (Arquitecures de maquinari) - Finalitzat</a><br>-->
        <a href="http://www.collados.org/asix2/index_asix2_pj6.html">ASIX2-PJ6 (Administraci□de xarxes i sistemes operatius)</a><br>
        <!--<a href="http://www.collados.org/asix2/index_asix2_pj7.html">ASIX2-PJ7 (Disseny i simulaci□de LANs) - Finalitzat</a><br>-->
        <a href="http://www.collados.org/asix2/index_asix2_sm6.html">ASIX2-SM6 (Fonaments de xarxes)</a><br>
        <a href="http://www.collados.org/asix2/index_asix2_eh4.html">ASIX2-EH4 (Network Lab)</a><br>
        <a href="http://www.collados.org/asix2/index_asix2_pj9.html">ASIX2-PJ9 (DevOps i Cloud Computing)</a><br>
        <b><u>DAW2</u></b><br>
        <a href="http://www.collados.org/daw2/index_daw2_pj6.html">DAW2-PJ6 (Back-End amb tecnologies PHP  i gestors de continguts)</a><br>
        <a href="http://www.collados.org/daw2/index_daw2_sm8.html">DAW2-SM8 (Desplegament d'aplicacions web)</a><br>
        </font>
        <br><br>
  </body>
</html>
Connection closed by foreign host.
```

S'observa el missatge *Connection close by foreing host* de manera que la connexió es tanca un cop rebuda la resposta

D'acord amb els documents RFC, la capçalera **Connection: close** ja sigui dins d'una petició o d'una resposta, significa que es tancarà la connexió un cop el procés de petició/resposta actual s'hagi completat. En aquest cas, això suposa que el client telnet es tancarà immediatament un cop rebuda la resposta del servidor i per tant no tindrem l'oportunitat d'intentar fer el redireccionament manualment.

b) Amb el resultat obtingut troba:

- Què hi ha dins del cos del missatge HTTP?

En el cos del missatge trobem el codi HTML del recurs / que en aquest cas és la pàgina index.html que es troba a l'arrel dels recursos servits per servidor quan s'estableix una connexió amb www.collados.org.

- Què hi ha dins del cos del missatge HTTP?

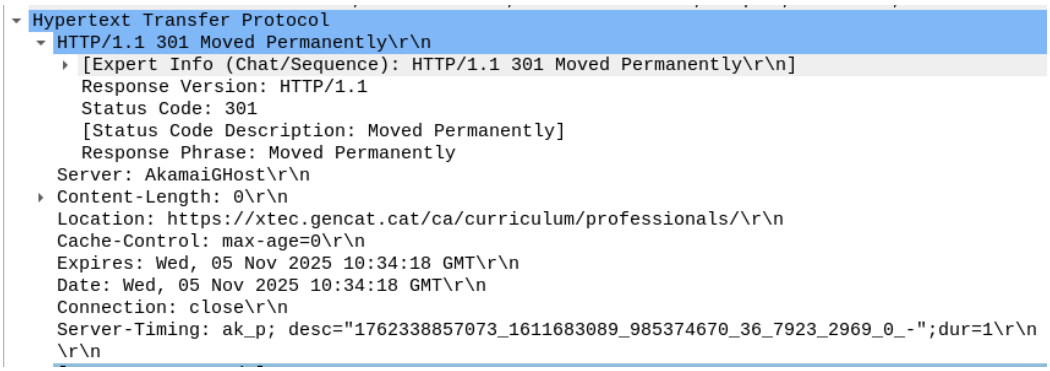
La mida del cos del missatge és de 1931 bytes a partir de la informació que es troba dins del camp Content-Length i que coincideix amb la mida del fitxer index.html en el servidor.

3. Posa en marxa **Wireshark** i filtra el contingut de les captures de manera que només es mostrin els continguts dels missatges HTTP. Estableix una connexió al port **80/tcp** de **www.google.com** i envia el següent missatge **HTTP**:

```
HEAD  /ca/curriculum/professionals/  HTTP/1.1
Host: xtec.gencat.cat
User-Agent: telnet
Connection: close
```

Mostra el resultat obtingut amb **Wireshark**, i troba:

a) Quina reposta dóna el servidor. Indica el significat del codi de la reposta i del camp Location.



D'acord amb el valor del camp Location, la nova URI (o URL) permanent (perquè el missatge és de tipus 301) del recurs demanat és: https://xtec.gencat.cat/ca/curriculum/professionals/.

b) Indica quina és la nova adreça del servidor.
El recurs demanat es troba ara aquí: https://xtec.gencat.cat/ca/curriculum/professionals/. La diferència es troba en el protocol que ara és HTTPS.

c) Indica el programa servidor de pàgines web utilitzat per xtec.gencat.cat.
Utilitza el programa servidor web AkamaiGHost.

d) Quina és la diferència entre haver utilitzat HEAD i no GET?
La resposta a una petició HEAD no inclou mai un cos de missatge i per tant, no podríem rebre la pàgina inicial del servidor.
Ara bé, en aquest cas concret no hi ha cap diferència perquè la resposta és un redireccionament i no inclou cap cos del missatge.

e) A quina hora s'ha generat el missatge HTTP en el servidor?
El dimecres dia 5 de Novembre a les 10:34:18 del meridià de Greenwich (GMT) que en aquesta època de l'any (5 de Novembre) va 1 hora endarrerit respecte del nostre horari.

f) Es tanca la connexió un cop rebuda la resposta. Per què?
Sí es tanca perquè el camp Connection de la resposta té un valor igual a close.

g) Per què el camp **Content-Length** és igual a 0?
Perquè la resposta indica que el recurs s'ha mogut a una nova localització i no es troba en el lloc demanat. El servidor indica la nova localització i per tant no pot enviar el recurs perquè no existeix. El servidor espera que el client faci una nova petició indicant la localització correcta per servir el contingut demanat.

4. Posa en marxa **Wireshark** i filtra el contingut de les captures de manera que només es mostrin els continguts dels missatges HTTP. Estableix una connexió al port **80/tcp** de **www.google.com** i envia el següent missatge **HTTP**:

```
GET / HTTP/1.1
Host: www.google.cat
User-Agent: telnet
Connection: close
```

Observa el resultat obtingut amb **Wireshark**, i troba:

a) Fixa't que a la petició el camp ***Host*** és **www.google.cat**. Pot ser diferent el valor de **Host** i el de la connexió via telnet?. Per que?.
- Sí, perquè el valor de la connexió de telnet serveix per indicar quin és el servidor al qual vull connectar-me. En aquest cas, el nom i domini serveixen per poder consultar a un servidor DNS, trobar la seva IP i establir la connexió.

- D'altra banda. un servidor web pot hostatjar multiples dominis. Amb l'adreça IP només establim la connexió però el servidor necessita el camp host per saber quin és el domini i per tant el lloc dins del servidor a on es troben els continguts que volem obtenir.

b) Què ha passat quan hem demanat l'accés a **www.google.cat** d'acord amb el missatge **HTTP** rebut des del servidor?
- El codi de resposta és el 301 Moved Permanently. Això vol dir que al recurs demanat se li ha assignat una nova URI (o URL) de manera permanent. En el futur, s'hauria d'utilitzar l'adreça que es troba en el camp Location.

- En el camp Location es troba la nova URI que és http://www.google.com

c) Què significa pel navegador que el camp ***Expires*** valgui el mateix que el camp ***Data*** i que la directiva ***max-age*** del camp de capçalera ***Cache-Control*** sigui igual a **604800**?
- Expires igual a la Data vol dir que la resposta caduca immendiatament i per tant no es pot reutilitzar, o sigui, no es pot desar dins de la memòria cau (caché). D'altra banda la directiva max-age indica que es pot reutilitzar i per tant, sí es pot desar dins de la memòria cau durant 604800 segons, o sigui, 7 dies. Passats 7 dies, el navegador no l'ha d'usar directament, sinó que ha de revalidar-la amb el servidor.

- D'acord amb els documents RFC pel protocol HTTP/1.1, quan hi són tots dos, el navegador ignora Expires i fa cas de max-age. Cache-Control té prioritat sobre Expires.

5. Analitza la següent "conversa" HTTP entre un client i un servidor i descriu allò que està passant:

```
GET /private/index2.html HTTP/1.1
Host: localhost
```

El client demana amb el mètode GET el recurs index2.html que es troba al directori private que penja de l'arrel de l'arbre de directoris que utilitza el servidor per emmagatzemar els recursos. La comunicació amb el servidor utilitzarà la versió 1.1 del protocol HTTP. El valor Host és localhost, de manera que s'està intentant realitzar una connexió a un servidor que s'executa en el mateix ordinador des del qual s'executa el client.

```
HTTP/1.1 401 Authorization Required
Server: HTTPd/1.0
Date: Sat, 27 Nov 2004 10:18:15 GMT
WWW-Authenticate: Basic realm="Secure Area"
Content-Type: text/html
Content-Length: 311
```

El servidor envia el codi 401 en la línia d'estat del missatge de resposta. Això significa que:

- La petició no s'ha dut a terme perquè manquen les credencials d'autenticació necessàries per accedir al recurs demanat.
- El servidor haurà d'enviar obligatòriament el camp de capçalera WWW-Authenticate indicant un o més mètodes d'autenticació (en aquest cas només el mètode "Basic" o mètode d'autenticació d'accés bàsic) i els paràmetres (realm="Secure Area") necessaris per poder accedir al recurs demanat quan es torni intentar accedir novament des del client.

El mètode "Basic" és una tècnica molt simple, sense encriptació i s'utilitza generalment sobre HTTPS. Quan s'utilitza un navegador com a client, en el moment de rebre la resposta apareixerà una finestra emergent (pop-up) per poder escriure les credencials (usuari i contrasenya). El realm defineix un conjunt de recursos del servidor que estan protegits i requereixen credencials per accedir-hi. Tots els recursos dins d'un realm comparteixen les mateixes credencials.

```
GET /private/index2.html HTTP/1.1
Host: localhost
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

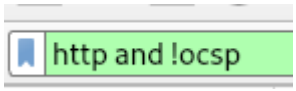
L'usuari escriu les credencials dins de la finestra emergent i el client genera una nova petició amb les dades donades pel client, indicant el mètode d'autenticació (Basic) i a continuació les credencials codificades (que no vol dir encriptades) en Base64 en comptes d'ASCII dins del camp Authoritzation.

```
HTTP/1.1 200 OK
Server: HTTPd/1.0
Date: Sat, 27 Nov 2004 10:19:07 GMT
Content-Type: text/html
Content-Length: 10476
```

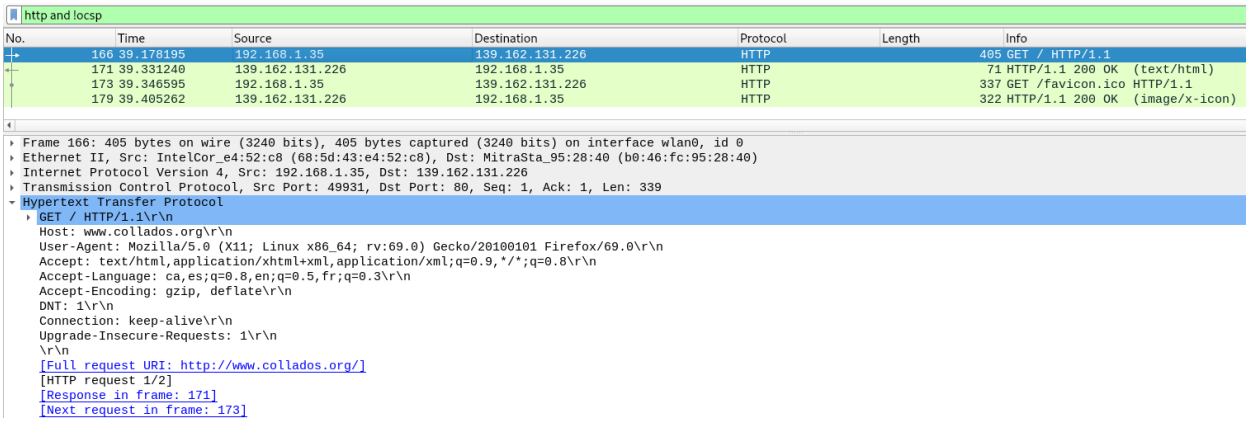
Les credencials enviades són correctes i el servidor envia el codi 200 OK indicant que la petició del client s'ha rebut, entès i acceptada, i que per tant s'envia el recurs dins del cos del missatge de resposta. El programa servidor és HTTPd versió 1.0. El missatge de resposta s'ha creat el 27 de Novembre del 2004 a les 10:19:07 GMT. El contingut és de tipus text/html (veure RFC 2854), o sigui text amb el conjunt de caràcters UTF-8 preferiblement i el contingut és HTML, i la mida del cos del missatge és de 10476 bytes.

6. Fes les accions que es demanen i respon preguntes:

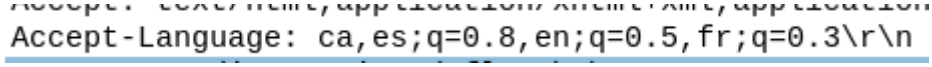
a) Descarrega la captura de wireshark http://www.collados.org/daw2/sm8/httpwireshark i fes un filtre per visualitzar només missatges HTTP i descartar missatge OSCP.



b) Obre el missatge amb la petició inicial realitzada pel navegador al servidor.



c) Quin és el propòsit del valor **q**?. Quin és l'efecte del valor en aquest cas concret?



El camp Accept-Language és utilitzat per l'User-Agent per indicar al servidor la (o les llengües) que són les preferides en la resposta del servidor.

És un valor entre 0 i 1 (no ha de tenir més de 3 decimals). Com més gran és el valor associat, mes preferent és una llengua. Si no s'assigna cap valor, llavors la preferència està marcada per l'ordre de la llista.

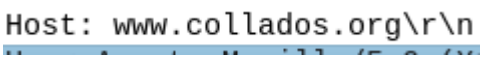
En aquest cas concret, la preferència d'idioma seria primer el català, segon l'anglès i tercer el francès. En el cas d'una web que pogués donar la resposta en diversos idiomes, aquest seria l'ordre de preferència.

d) Quin navegador i versió s'ha utilitzat?



Firefox versió 69.0

e) A quin host virtual hem accedit?



Al lloc o host virtual www.collados.org

f) De quin tipus era la petició?. Era de lectura o escriptura?

```
GET / HTTP/1.1\r\n
```

S'ha utilitzat el mètode GET i per tant era una petició de lectura

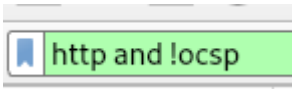
g) Obre el segon missatge de petició del navegador al servidor i comprova que demana?.

```
GET /favicon.ico HTTP/1.1\r\n
```

Demana el fitxer **favicon.ico** que es troba a l'arrel del servidor web, és a dir, a /var/www/html si utilitzen Apache2 i Debian GNU/Linux o c:\xampp\htdocs si utilitzem XAMPP sobre Windows

7. Fes les accions que es demanen i respon preguntes:

a) Descarrega la captura de wireshark http://www.collados.org/daw2/sm8/httpwireshark i fes un filtre per visualitzar només missatges HTTP i descartar missatge OSCP.



b) Obre el missatge amb la resposta del servidor a la petició inicial realitzada pel navegador.

```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Mon, 30 Sep 2019 11:50:34 GMT\r\n
    Server: Apache\r\n
    Content-Encoding: gzip\r\n
    Vary: Accept-Encoding\r\n
    Strict-Transport-Security: max-age=31536000\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Transfer-Encoding: chunked\r\n
    Content-Type: text/html; charset=windows-1252\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.153045000 seconds]
    [Request in frame: 166]
    [Next request in frame: 173]
    [Next response in frame: 179]
    [Request URI: http://www.collados.org/favicon.ico]
  ▸ HTTP chunked response
    Content-encoded entity body (gzip): 1004 bytes -> 2437 bytes
    File Data: 2437 bytes
▸ Line-based text data: text/html (38 lines)
```

c) Quina és la data de creació del missatge HTTP?

```
Date: Mon, 30 Sep 2019 11:50:34 GMT\r\n
```

El dilluns 30 de setembre del 2019 a les 11h 50m 34s del meridià de Greenwich (GMT) que en aquell moment de l'any va 2 hores endarrerit respecte de l'horari de Barcelona.

d) El missatge ha trobat el recurs demanat?. Com ho saps?.

```
▸ HTTP/1.1 200 OK\r\n
```

Sí, perquè a la línia d'estat (status) del missatge de resposta el codi enviat és el 200, que vol dir que el recurs ha estat trobat i s'envia juntament amb el missatge HTTP.

e) Quin tipus de document s'han enviat i quina és la seva codificació?. Com ho saps?

```
Content-Type: text/html; charset=windows-1252\r\n
```

És un document amb codificació de caràcters windows-1252 que permet generar el caràcters necessaris utilitzats a Europa occidental, Amèrica, Ocenia i una part important d'Àfrica. Permet treballar en idiomes com el català, castellà, anglès, francès, portugès, italià, alemany i molts altres idiomes.

f) Què s'envia dins del camp de dades del missatge HTTP?

Un fitxer html. En aquest cas, el fitxer demanat a la petició anterior, que és el fitxer inicial index.html del lloc virtual www.collados.org. Aquest fitxer té una mida de 2437 bytes.

g) Temps que recordarà el navegador que el servidor només és accessible via HTTPS?. Com ho saps?

```
Strict-Transport-Security: max-age=31536000\r\n
```

El camp Strict-Transport-Security (o HSTS) informa al navegador el lloc web només és accessible via HTTPS.

La directiva max-age indica el temps en segon que el navegador ha de recordar que el lloc web només és accessible utilitzant HTTPS.

Així, doncs, el navegador ha de recordar durant 31.536.000 segons (1 any) que el lloc web és només accessible via HTTPS.

h) La connexió es tancarà després d'envia el missatge?. Com ho saps?.

```
Connection: Keep-Alive\r\n
```

No, perquè el camp Connection val Keep-Alive.

8. Comprovant el funcionament de les peticions POST i GET:

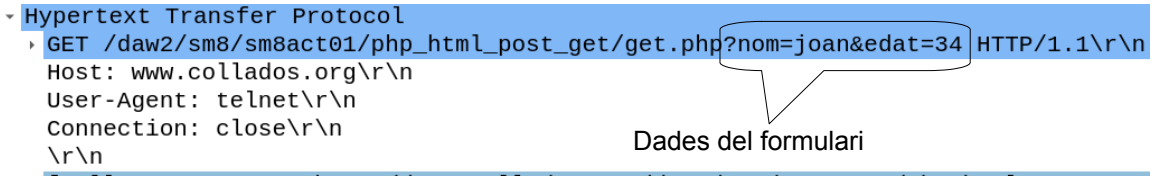
a) Filtre: http

b) Estableix una connexió via **telnet** a **www.collados.org** pel port **80** utilitzant el mètode **GET** i demanant una connexió a l'aplicació **/daw2/sm8/php_html_post_get/index.html** a la qual has d'envia els valors **name=joan** i **edat=34**. El protocol utilitzat serà HTTP/1.1 i al final es **tancarà** la connexió.

```
collados@sh4:~$ telnet www.collados.org 80
Trying 156.67.24.117...
Connected to www.collados.org.
Escape character is '^]'.
GET /daw2/sm8/sm8act01/php_html_post_get/get.php?nom=joan&edat=34 HTTP/1.1
Host: www.collados.org
User-Agent: telnet
Connection: close
```

c) Comprova:

- Amb wireshark troba que s'ha creat una petició de tipus GET. Demostra-ho.
- Comprova que les dades s'envien dins de la línia de petició del missatge. Demostra-ho.



d) Estableix una connexió **telnet** a **www.collados.org** pel port **80** utilitzant el mètode **POST** i demanant una connexió a l'aplicació **/daw2/sm8/php_html_post_get/post.php** a la qual has d'envia els valors **name=jaume** i **edat=32**. El protocol utilitzat serà **HTTP/1.1** i al final es **tancarà** la connexió.

```
collados@sh4:~$ telnet www.collados.org 80
Trying 156.67.24.117...
Connected to www.collados.org.
Escape character is '^]'.
POST /daw2/sm8/sm8act01/php_html_post_get/post.php HTTP/1.1
Host: www.collados.org
User-Agent: telnet
Content-Type: application/x-www-form-urlencoded
Content-Length: 17
Connection: close

nom=jaume&edat=32
```

e) Comprova:

- Quins camps has hagut d'afegir?. Què fan?
  - Content-Type i Content-Length
  - Content-type indica el tipus de dades (imatges, html, css, json, pdf, dades d'un formulari, formularis amb fitxers adjunt) enviades. En el cas x-www-form-urlencoded indica que s'envien dades d'un formulari.
  - Content-Length indica la mida en bytes del cos del missatge. En aquet cas 17 bytes, que concideix amb la mida de la cadema "nom=jaume&edat=32". S'haurien pogut afegir els 2 bytes de CRLF i la mida hauria estat 19.
- S'ha esperat abans d'enviar el missatge a tenir les dades?. Per què?
  Sí, a causa del camp Content-Length que fa que el servidor esperi les dades dins del cos del missatge.
- Amb wireshark que s'ha creat una petició de tipus **POST**. Demostra-ho.
- Comprova que les dades s'envien dins del cos del missatge. Demostra-ho.



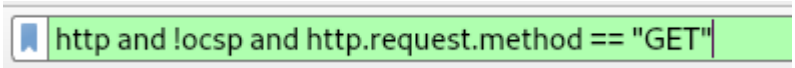
9. Establint múltiples connexions:

a) Treballa amb **Firefox**. Connectat a **http://www.binefa.net/electronica/tutorial04/**. Comprova quantes peticions GET s'han generat per part del client i i quantes respostes respostes ha enviat el servidor.

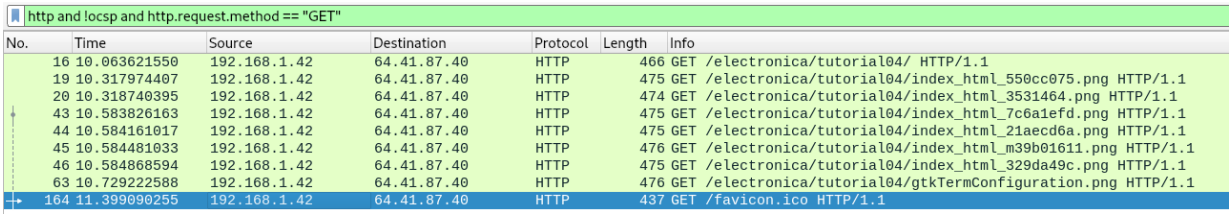
- Després de posar en marxa el Wireshark i deixar-lo fent captures de paquets de xarxa, amb el navegador m'he connectat a **http://www.binefa.net/electronica/tutorial04/**. Un cop rebuda tota la pàgina web, he aturat la captura de paquets.

- La captura de paquets la podeu descarregar d'aquí.

- He fet un filtre per mostrar només les captures de peticions utilitzant el mètode GET. El filtre és aquest:

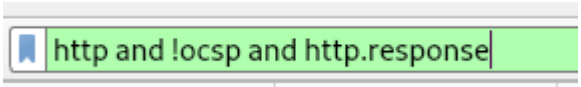


- Wireshark m'ha mostrat les següent captures utilitzant el mètode GET:

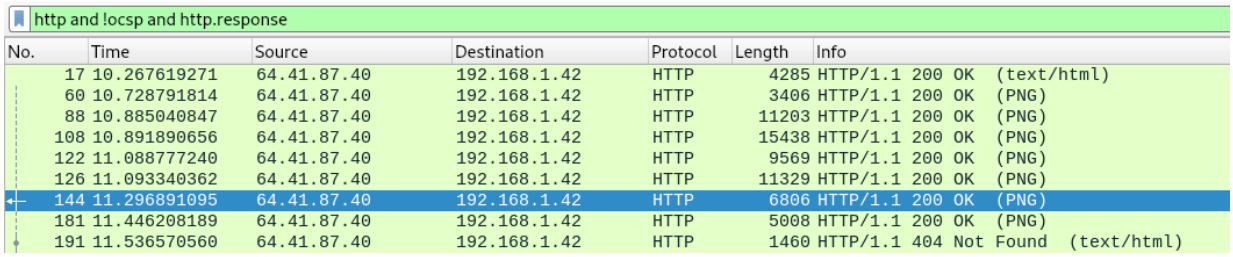


- Pel que es veu a la captura, a més de demanar la pàgina inicial, també demana 7 imatges .png i el fitxer favicon.ico.

- A continuació, he fet un filtre per mostrar les captures de respostes:



- Wireshark m'ha mostrat les següent captures de respostes del servidor a les peticions GET:



- Pel que es veu, s'envia una resposta per enviar la pàgina web i també una resposta per les imatges .png demanades. El missatge 404 Not Found correspon a un missatge d'error del navegador indicant que la petició de favicon.ico és incorrecta perquè el recurs no existeix en el servidor.

b) Indica el motiu pel qual, a part de la petició inicial del client i la resposta inicial del servidor, s'han generat noves peticions i respostes.

Si comprovem el codi HTML enviat al cos del missatge de la primera resposta, veuerem que es demanen una sèrie d'imatges en format **png**. El client demana al servidor aquestes imatges per mitjà de noves peticions utilitzant el mètodes GET perquè no arriben juntament amb el codi HTML i per això es generen aquestes peticions noves. El servidor envia també respostes amb les imatges en el cos del missatge, i per això es generen missatges de resposta nous.

c) Comprova que cada petició ha implicat la utilització d'un nou port per part del client. Per què?

- He fet un filtre per mostrar només les captures de peticions utilitzant el mètode GET. El filtre és aquest:



- Per la captura 19 a on es demana el primer fitxer PNG, el sistema ha obert pel navegador el port 58080/tcp.
- Per la captura 20 a on es demana el segon fitxer PNG, el sistema ha obert pel navegador el port 58082/tcp.
- Per la captura 43 a on es demana el tercer fitxer PNG, el sistema ha obert pel navegador el port 58106/tcp.
- Per la captura 44 a on es demana el quart fitxer PNG, el sistema ha obert pel navegador el port 58122/tcp.
- Per la captura 45 a on es demana el cinquè fitxer PNG, el sistema ha obert pel navegador el port 58098/tcp.
- Per la captura 46 a on es demana el sisè fitxer PNG, el sistema ha obert pel navegador el port 58110/tcp.
- Per la captura 63 a on es demana el setè fitxer PNG, el sistema ha obert pel navegador el port 58134/tcp.

El protocol HTTP 1.1 treballa amb connexions persistents, de manera que, en principi, es pot aprofitar una única connexió TCP per recollir recursos de múltiples URIs. Però estem veient que en aquest cas s'està obrint un port separat per recollir cadascuna de les imatges.

Treballar sense connexions persistents té el problema de que la xarxa es pot congestionar més fàcilment, augmenta el temps de resposta i la carrega de la CPU.

D'altra banda, si per cada imatge establim una connexió diferent fent servir ports TCP diferents, podrem baixar més ràpidament les imatges del servidor, perquè podrem descarregar-les en paral·lel.

10. Treballant amb cookies

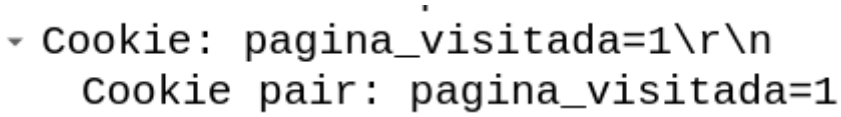
a) Descarrega la captura **http://www.collados.org/daw2/sm8/sm8act01/cookies00.pcapng** i fes un filtre per visualitzar només missatges HTTP.

b) Obre el missatge **No. 6** que és la resposta del servidor a la primera petició del navegador. Què significa el camp **Set-Cookie**?. Quin és el nom de la cookie enviada?. Quant val?. Quan expira?.

```
Server: Apache/2.4.38 (Debian)\r\n
Set-Cookie: pagina_visitada=1; expires=Tue, 15-Nov-2022 20:53:25 GMT; Max-Age=2592000; path=/\r\n
```

Aquesta línia és una capçalera de resposta amb la qual el servidor demana al navegador que guardi una cookie. El nom de la cookie és pagina_visitada i el seu valor és 1. La cookie té una durada de 2592000 segons, és a dir 30 dies i d'acord amb la seva data de creació, caducarà el 15-11-2022 a les 20:53:25. La cookie s'enviarà des del client al servidor en qualsevol petició a qualsevol lloc dins de l'arrel per això path=/.

c) Obre el missatge **No. 8** que és la segona petició del navegador. Que significa el camp Cookie?. Quin és el nom de la cookie enviada?. Quant val?. Per què?.



Aquesta és la capçalera de petició que el navegador envia al servidor en les visites posteriors, un cop ha rebut el Set-Cookie. Cookie és el camp a on es desen les cookies que envia el navegador o client.

d) Obre el missatge **No. 4** que és la primera petició del navegador. Pots trobar el camp de capçalera **Cookie**?. Per què?

No, perquè és la primera vegada que es connecta al servidor i per tant no ha pogut rebre encara una cookie.

C) Taula comparativa de mètodes

a) Què significa que un mètode sigui segur?. Quins mètodes són segurs?
D'acord amb el document RFC 9110 secció 9.2.1 els mètodes segurs son aquells que per la seva definició són esencialment de només lectura i que per tant, per ells mateixos, no esperem que facin cap mena de mdificació en l'estat del servidor i els seus recursos. El mètodes GET, HEAD, OPTIONS i TRACE són mètodes segurs.

b) Què significa que un mètode sigui "Cacheable"?. Quins mètodes són "Cacheables"?
D'acord amb el document RFC 9110 apartat 9.2.3 els mètodes "cacheables" són aquells que es permet que les seves respostes s'emmagatzemin per ser utilitzades en el futur. Un lloc típic a on emmagatzemar una resposta és a la caché d'un proxy o un navegador. En general els mètodes cacheables són GET, HEAD i POST però a la realitat, una majoria aclaparadora d'implementacions pràctiques de sistemes de cache només emmagatzemen respostes de tipus GET i HEAD.

c) Fes una taula comparativa dels mètodes GET, PUT, POST, HEAD i DELETE indicant per cada mètode si té les següents propietats:

- La petició envia dades al cos del missatge?
- La petició envia dades dins de la línia de petició
- La resposta a la petició té dades al cos del missatge?
- És un mètode segur?
- És un mètode idempotent?
- És un mètode cacheable?

Mètode HTTP	Petició amb dades al cos del missatge	Dades a la línia de petició	Resposta amb dades al cos del missatge	Segur	Idempotent	Cacheable
GET	NO DEFINIT (*1)	SÍ	SÍ	SÍ	SÍ	SÍ
POST	SÍ	SÍ (*2)	SÍ	NO	NO	SÍ (*4)
PUT	SÍ	SÍ (*2)	SÍ (*3)	NO	SÍ	NO
HEAD	NO DEFINIT (*1)	SÍ	NO	SÍ	Sí	SÍ
DELETE	NO DEFINIT(*1)	SÍ	SÍ (*3)	NO	Sí	NO

(*1) Estrictament parlant, si llegim els documents RFC, no hi cap indicació estricta de si es pot fer o no, però a la pràctica, normalment NO s'envien dades al cos del missatge i és una pràctica que no es recomana perquè ningú no espera que això passi i pot donar lloc a molts problemes en els servidors i clients.

(*2) Estrictament parlant, es pot fer i no hi ha cap limitació si llegim els documents RFC però com a regla genera és un elecció dolenta (mida màxima de les dades, privacitat,comportaments inesperats..).

(*3) Estrictament parlant, l'enviament és opcional si llegim els documents RFC i en principi sí es poden enviar dades. Ara bém normalment, s'envia com a resposta el codi 204 No Content i un cos del missatge sense dades.

(*4) Estrictament parlant, es pot fer cacheable afegint els camps de capçlaera adequats. A la pràctica, gairebé ningú emmagatzema a la caché les respostes a peticions del mètode POST i passa tan rarament que es pot dir que la resposta realment hauria de ser NO.