NGINX → Servidor WEB que ha superat Apache.

Protocolo http (puerto 80) (la evolución és la capa extra https puerto 443)

Clientes Servidores Web

-(Navegadores) -Apache, NGNIX

-Lynx→ Navegador tipo texto

-Wget

-curl

El protocol http és stateless -> Petición → respuesta y **termina la comunicación**,

però como conseguimos que se mantenga la sesión??? (con cookies). Se van intercambiando cookies de sesión (login etc)

Culturilla: <http://info.cern.ch> → 1er servidor web (www) del mundo, aún encendido.

ping info.cern.ch

nc 188.184.21.10 80

PROTOCOLO: Conjunto de reglas que definen cómo pedir las cosas y cómo se devuelven.

*nc 188.184.21.10 80*

*GET /index.html HTTP/1.0^M (si tarda mucho TIMEOUT) (parte del protocolo)*

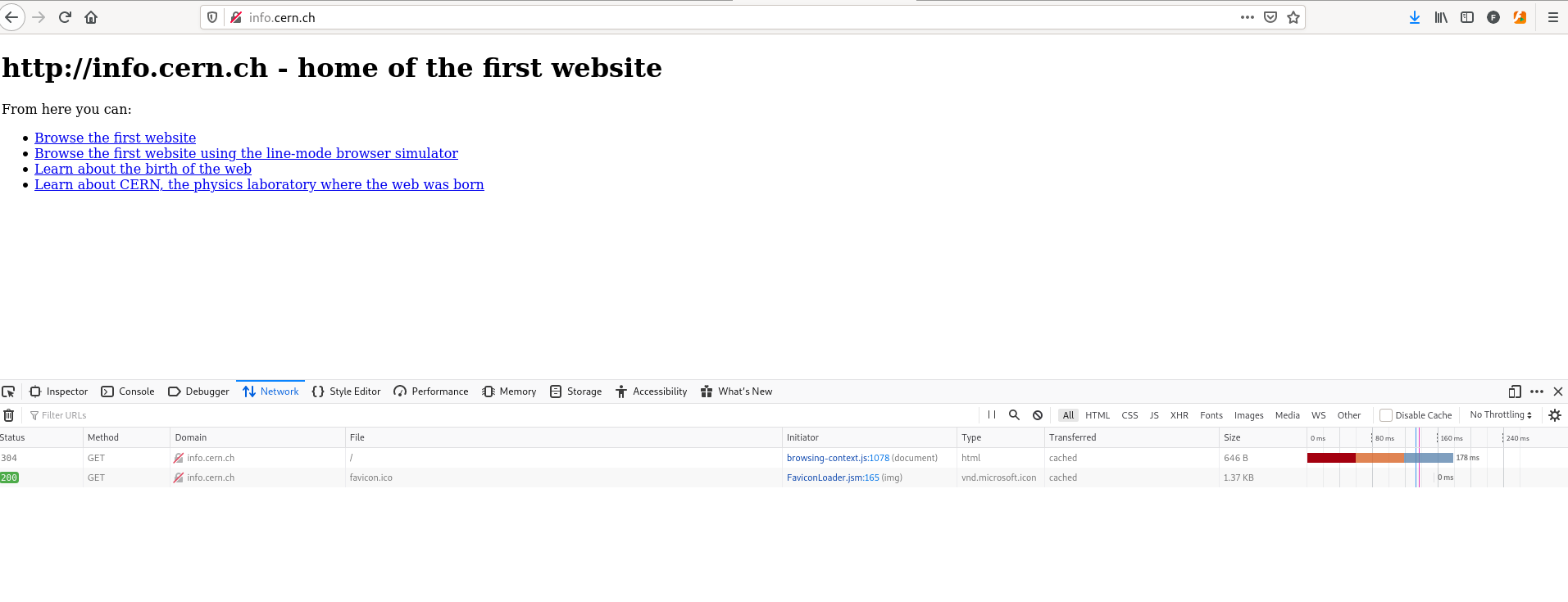
*host: info.cern.ch*

*printf 'GET /index.html http/1.0\r\nHost:info.cern.ch\r\n\r\n' | nc info.cern.chn 80*

------------

Vamos al navegador WEB, pulsamos F12:

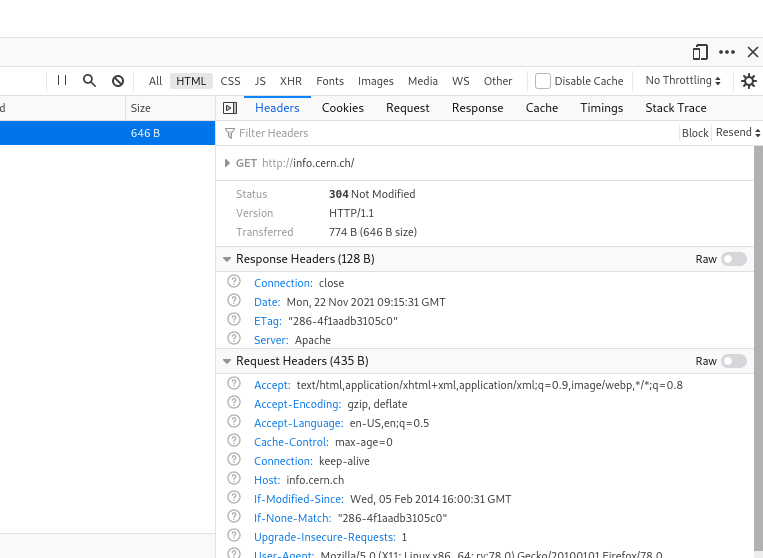
Pestaña network i damos a recargar.



1ero obtiene la IP

Después envía la petición.

Los servidores pueden alojar varias webs !!





Los servidores WEB van con API ( no sólo envia WEBs, también envian videos, imagenes etc..) con lo cual és más correcto llamarles **servidores http**.

Arxiu configuració:

ls -la **/etc/nginx/sites-enabled/default**

És un enlace (soft-link) a /etc/nginx/sites-available/default

Ponemos :

systemctl restart ngnix:

Creamos una web sencilla con html.

Ahora vamos a nuestro PC (PC cliente):

Printf 'GET /page.html HTTP/1.0\r\n\r\n' | nc 192.168.122.113 80

Lo mismo con firefox 192.168.122.113, no hace falta poner el puerto (:80), pero no funciona

pero miramos pulsando F12, vemos que no podemos por XXXXXXXXXXx

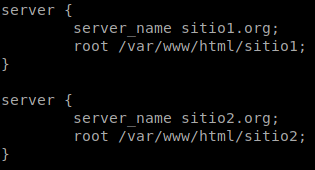
Funcionar si le especificamos el recurso /192.168.122.113/page.html

----------------

Leer buena praxis de diseño web --> Google css

Hazlo así para no enviar cabeceras innecesaria (paja) !!

TENIAMOS 2 CARPETAS EN /var/www/html/ CON UN FICHERO CADA UNA TIPO HTML (pag.html i pag2.html)

****

Printf ' GET /pag.html

AQUÍ ES EL RECURSO DENTRO DEL SERVIDOR

¿ Qué página devuelve este comando ?

Printf ' GET /pag.html HTTP/1.0\r\n\r\n' | nc sitio2.org 80

¿ Qué página devuelve este comando ?

Printf ' GET /pag2.html HTTP/1.0\r\n\r\n' | nc sitio2.org 80

NO devuelve la página (nc resuelve la IP=192.168.11.113:80),

Està la pàgina 2 en el ??? no.

Tiene que especificar el host donde está el recurso !

Printf ' GET /pag2.html HTTP/1.0\r\nHost:sitio2.org\r\n' | nc sitio2.org 80

AHORA SI VA

Sitio2 es un dominio inventado para el nombre del servidor.

Printf ' GET /pag2.html HTTP/1.0\r\nHost:sitio2.org:pag1\r\n' | nc sitio2.org 80

**DARA ERROR PQ NO HAY PAG1 !!**

Es lo mismo **nc sitio1.org 80** que **nc sitio2.org 80**. Devuelven la misma IP del servidor !

URL → Unite Resource Locator→ Donde obtener un recurso (localización fichero)

URI → Unite Resource Identifier → No dice donde, dice el identificador

content://resources.imagen1.jpeg (yo quiero esto, buscate la vida para enseñarmelo, mirara con otro program en una bbdd)

MÁS COSAS QUE SE AÑADEN EN UNA URL:

Después de recurso en URL →

index.htm? **Luego Pares clave&valor** (són querys de busqueda)

¿Qué más se añade ?

index.html? Pares clave&valor (són querys de busqueda) + **FRAGMENT**

[**http://user:pass**@servidor:8000](http://user:pass@servidor:8000)

------------------------------------------------ 29/11/21-------------------------------

<https://gerardfp.github.io/>

nc 192.168.122.169 80 Petición conexión TCP por puerto 80

Ahora ponemos:

GET /index.html HTTP/1.0^M^M | nc 192.168.122.169 80

En el archivo:

server {  **←Primero busca aquí, si no encuentra..**

root /var/www/html

location /a/ { **← Luego busca aquí**

root /tmp/;

}

}

I reinciamos nginx

Desde shell → printf 'GET /a/index.html HTTP/1.0\r\n\r\n' | nc 192…

Si le damos devuelve "Not found " en las cabeceras

mkdir /tmp/a/

printf 'yesss'>/tmp/a/index.html

ARA SI, surt yesss !!

mkdir /tmp/b/

printf 'yesss'>/tmp/b/index.html

ARA NO !, b no esta definit !!

Tenim sitioa i sitob dins de /var/www/html/

(els iconos de les 2 webs son els mateix i altres coses)

Descarreguem un icone amb wget i enganxem la URL de la imagen (del recurs)

Ara tenim un fitxer que es diu icon.svg i el posem a carpeta creada imatges.

Ara tenim

Vim sitioa/page.html

<h1> This is sitioa</h1>

<img src='images/icon.svg'>

Vim sitioa/page.html

<h1> This is sitiob</h1>

<img src='images/icon.svg'>

Ahora

vim /etc/nginx/sites-enabled/default

server {

server\_name sitio a.com;

root /var/[www/html](http://www.html)/sitioa;

}

server {

server\_name sitio b.com;

root /var/[www/html](http://www.html)/sitiob;

}

Anem al navegador sitioa.com/index.html

Funciona PERO no es descarga la imagen !!

Ahora rectifiquen:

a

vim /etc/nginx/sites-enabled/default

server {

server\_name sitioa.com;

root /var/[www/html](http://www.html)/sitioa;

location /images/

root /var/www/html;

}

server {

server\_name sitiob.com;

root /var/[www/html](http://www.html)/sitiob;

location /images/

root /var/www/html;

}

**Reinciciem ngnix i apareix la imatge !!**

ARA fem PER (expresion regulars, exemples)

location ~ \.(jpg | jpeg)$ {

root /var/www/html/;

}

location ~ ^icon\.(jpg | jpeg)$ { ← no se si esta bé el PER

root /var/www/html/;

}

location / { ← busca todo lo que haya barra para abajo(arbol)/atrás(directorio)

}

location =/images ← busca exactamente aquí.

~ empieza el PER \. Es el punto, $ que termina la expresión.

ES MEJOR HACERLO ASI PARA NO PONER LA RUTA /images/

EN LAS WEBS html !!!!!

ASÏ QUEDARIA:

Vim sitioa/page.html

<h1> This is sitio a</h1>

<img src='/icon.svg'>

Vim sitioa/page.html

<h1> This is sitio b</h1>

<img src='/icon.svg'>

----------------------------------------------------------------------

Como configurar SSL (añadir a http capa TLS)

**CLIENT: 443 (HTTPS) SERVIDOR**

**PETICIO**

**FIREFOX ------------------------------------------------------------------> SERVIDOR**

**clau**

**<-----------------------------------------------------------------**

**contingut**

**<-----------------------------------------------------------------**

Https és http + capa TLS.

Lo primero que manda el servidor al cliente es la clave pública. **(.pem)**

Luego !!

Luego manda el contenido encriptado.

Server {

listen 443 ssl;  *(si añadimos ssl encriptamos)*

*ssl \_certificate* /etc/ssl/cert/site.pem;

ssl\_certifiacte key /etc/ssl/private/site.key;

*server\_name sitioa.com;*

Ahora generaremos las claves en el servidor.

Orden:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -subj '/CN=mysite.com' -keyout /etc/ssl/private/site.key -out /etc/ssl/certs/site.pem

La guarda en /etc/ssl/private/

Una .pem i la otra .key la **.pem es la publica** i **la key la privada**. (la PEM serà la clau que el nostre servidor li enviarà al client)

Si intentamos acceder Firefox nos dirà que NO ES SEGURO. No està certificada !!

Imaginemos que alguien intercepta MIT (ment in the middle) NO sirve de nada

Para eso existen las claves públicas encriptadas que venden las CA (empresa certificadoras) Firefox o cualquier navegador tienen estas claves.

Privacy and Security → Podemos ver todas las empresas que tiene certificación.

(tenemos que demostrar que la empresa es nuestra, poniendo en la web pública dentro de un html un hash o algo)

Let's Encrypt → Esta empresa genera certificados gratis

Si nos ponemos en modo promiscuo (con la tarjeta de red ? SI !!) capturamos todos los datos aunque no vayan dirigidos a nosotros.(sniffing) i el siguiente paso es con un servidor DHCP poder hacer un Men in the Middle. Entonces els https sin certificado no serviria de nada !!.

Vamos al virtual manager, vemos la interface(virtb0 ) i capturaremos el trafico de esa interfaz

1er comentamos esto:

server {

#listen 443 ssl;  *(si añadimos ssl encriptamos)*

*#ssl \_certificate* /etc/ssl/cert/site.pem;

#ssl\_certifiacte key /etc/ssl/private/site.key;

*server\_name sitioa.com;*

}

(desde local(hem d'instalar **tcpdump**))

**sudo tcpdump -i virtb0 -w http.pcap**  (esto se pone a snifar todos los paquetes)

Anem al navegador i pulsem enter a <http://sitioa/.html> (sin www que no vamos a internet)

Ara con WireShark podemos abrir el archivo i mirarlo. **(el http.pcap)**

Filtramos por (http) y vemos la info sin cifrar.

Ahora si lo habilitamos vemos que todo el protocolo http va encriptado (no sólo el http)

Las peticiones GET también !!!!