

Práctica 6

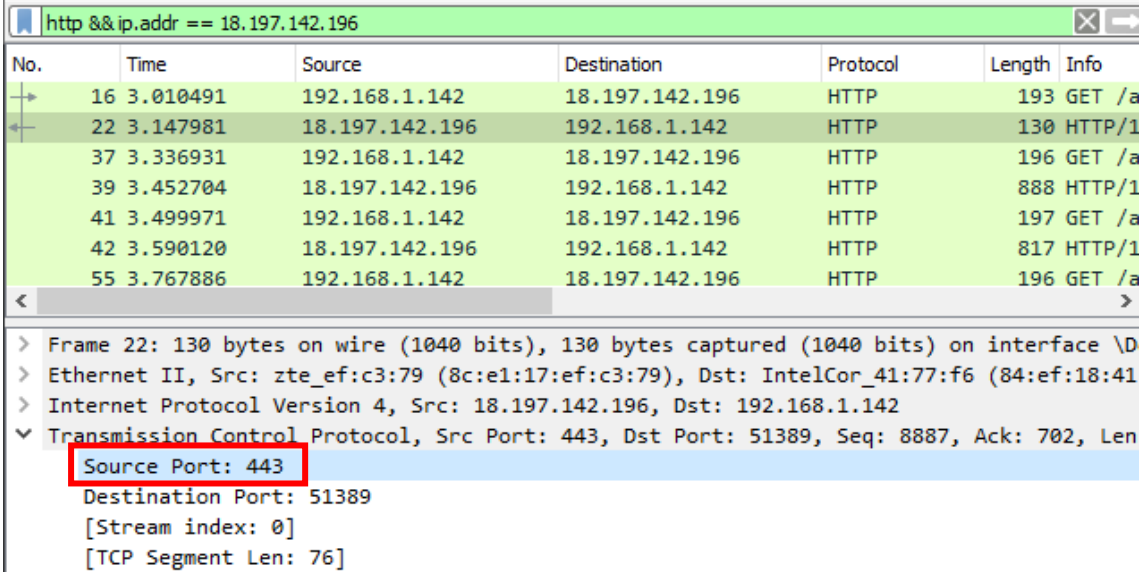
Alumno 1: López Pérez, Marta

Titulación: Grado de Ingeniería Informática

PC de la práctica: 012

Ejercicio 1. ¿Cuál es el puerto utilizado por el servidor? ¿Es el normal de HTTP (80)? ¿Por qué?

Utiliza el puerto 443. No usa el 80 porque usa HTTPS y el puerto que corresponde a HTTPS es 443.



No.	Time	Source	Destination	Protocol	Length	Info
16	3.010491	192.168.1.142	18.197.142.196	HTTP	193	GET /a
22	3.147981	18.197.142.196	192.168.1.142	HTTP	130	HTTP/1
37	3.336931	192.168.1.142	18.197.142.196	HTTP	196	GET /a
39	3.452704	18.197.142.196	192.168.1.142	HTTP	888	HTTP/1
41	3.499971	192.168.1.142	18.197.142.196	HTTP	197	GET /a
42	3.590120	18.197.142.196	192.168.1.142	HTTP	817	HTTP/1
55	3.767886	192.168.1.142	18.197.142.196	HTTP	196	GET /a

> Frame 22: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface \D
> Ethernet II, Src: zte_ef:c3:79 (8c:e1:17:ef:c3:79), Dst: IntelCor_41:77:f6 (84:ef:18:41
> Internet Protocol Version 4, Src: 18.197.142.196, Dst: 192.168.1.142
▼ Transmission Control Protocol, Src Port: 443, Dst Port: 51389, Seq: 8887, Ack: 702, Len
Source Port: 443
Destination Port: 51389
[Stream index: 0]
[TCP Segment Len: 76]

Ejercicio 2. Observe el número de conexiones realizadas. ¿Cuántas hace? ¿Usa una conexión permanente (en la misma conexión hace varias peticiones) o no permanente (solo realiza una por conexión)? En caso de ser permanente, ¿qué cabecera de la petición indica que queremos que sea permanente?

Se realizan 31 conexiones (A la hora de capturar he realizado 2 rondas con 5 preguntas cada una).

Usa una conexión permanente, porque indica el nombre de la conexión: keep-alive.

ip.addr == 18.197.142.196					
No.	Time	Source	Destination	Protocol	Length
589	52.467482	192.168.1.142	18.197.142.196	HTTP	1
590	52.550184	18.197.142.196	192.168.1.142	TCP	8
591	52.577713	18.197.142.196	192.168.1.142	HTTP	8
592	52.579789	192.168.1.142	18.197.142.196	HTTP	1
593	52.670442	18.197.142.196	192.168.1.142	HTTP	8
594	52.671193	192.168.1.142	18.197.142.196	TLSv1.2	8
595	52.671286	192.168.1.142	18.197.142.196	TCP	8

>	Frame 592: 197 bytes on wire (1576 bits), 197 bytes captured (1576 bits) on interface
>	Ethernet II, Src: IntelCor_41:77:f6 (84:ef:18:41:77:f6), Dst: zte_ef:c3:79 (84:ef:18:41:77:f9)
>	Internet Protocol Version 4, Src: 192.168.1.142, Dst: 18.197.142.196
>	Transmission Control Protocol, Src Port: 51419, Dst Port: 443, Seq: 630, Ack: 5
>	Transport Layer Security
>	Hypertext Transfer Protocol
>	GET /api/planets/40/ HTTP/1.1\r\n
	User-Agent: \r\n
	Accept: application/json\r\n
	Host: swapi.dev\r\n
	Connection: keep-alive\r\n
	\r\n
	[Full request URI: https://swapi.dev/api/planets/40/]
	[HTTP request 2/2]
	[Prev request in frame: 589]
	[Response in frame: 593]

Ejercicio 3. Observe una respuesta, ¿cómo se identifica dónde acaban las cabeceras HTTP y empieza el recurso?

Se identifican con una línea en blanco que separa el recurso de las cabeceras. Justo después de la línea empieza el recurso (datos).

http &&ip.addr == 18.197.142.196

No.	Time	Source	Destination	Protocol
16	3.010491	192.168.1.142	18.197.142.196	HTTP
22	3.147981	18.197.142.196	192.168.1.142	HTTP
37	3.336931	192.168.1.142	18.197.142.196	HTTP
39	3.452704	18.197.142.196	192.168.1.142	HTTP
41	3.499971	192.168.1.142	18.197.142.196	HTTP
42	3.590120	18.197.142.196	192.168.1.142	HTTP
55	3.767886	192.168.1.142	18.197.142.196	HTTP

> Transport Layer Security

▼ Hypertext Transfer Protocol

> HTTP/1.1 200 OK\r\n

Server: nginx/1.16.1\r\n

Date: Sun, 21 Jun 2020 20:21:49 GMT\r\n

Content-Type: application/json\r\n

Transfer-Encoding: chunked\r\n

Connection: keep-alive\r\n

Vary: Accept, Cookie\r\n

X-Frame-Options: SAMEORIGIN\r\n

ETag: "fd80e7a2dd56103df8fa993d41db2c48"\r\n

Allow: GET, HEAD, OPTIONS\r\n

Strict-Transport-Security: max-age=15768000\r\n

\r\n

[HTTP response 2/2]

[Time since request: 0.090149000 seconds]

[\[Prev request in frame: 37\]](#)

[\[Prev response in frame: 39\]](#)

[\[Request in frame: 41\]](#)

[Request URI: https://swapi.dev/api/planets/33/]

> HTTP chunked response

File Data: 395 bytes

> JavaScript Object Notation: application/json

Cabeceras

Línea en blanco

Ejercicio 4. Describa el significado de las cabeceras de una petición y una respuesta (sin incluir las que empiecen por x-).

DESCRIPCIÓN DEL CÓDIGO

En cada uno de los métodos a completar se han realizados las mismas operaciones.

- 1º. Todas las posibles excepciones están capturadas
- 2º. Creamos la URL con new URL() y las conexiones.
- 3º. Añadimos las cabeceras y lo deserializamos.

En el código main he añadido una pregunta extra del tipo masAlto que se llama masPeso, que te dice quién pesa más de dos personas.

```
21
22     do{
23         masAlto(sw);
24         quienVive1(sw);
25         quienVive2(sw);
26         masPeso(sw);
27         System.out.println("Desea otra ronda (s/n)?");
28         respuesta = sc.nextLine();
29     }while(respuesta.equals("s"));
30     sc.close();
31
32 }
33
34 // Genera un número entre 0 y max-1 que no haya sido usado previamente (los usados vienen en l)
35 public static Integer getRandomMass(int max, List<Integer> l)
36 {
37     if(max == l.size()) return null;
38
39     Integer p = rand.nextInt(max);
40     while(l.contains(p)){
41         p = (p+1)%max;
42     }
43     return p;
44 }
45
46 private static void masPeso(SWClient sw)
47 {
48     // Obteniendo la cantidad de gente almacenada
49     int date = sw.getHeightOfPerson("people");
50     if(date == 0)
51     {
52         System.out.println("No se encontraron personas.");
```