

PRÁCTICA 1

Viktor Yosava

vikyosava@uma.es

Redes y Sistemas Distribuidos. Ingeniería de la Salud

Análisis con Wireshark y desarrollo de protocolos de nivel de aplicación (DNS y HTTP).

Información Básica

Se utiliza Wireshark para capturar y analizar los mensajes de la red, proporcionando información acerca de los encabezados de los diferentes protocolos de la arquitectura de red.

Utilizaremos el navegador y diferentes comandos para generar tráfico de diferentes aplicaciones.

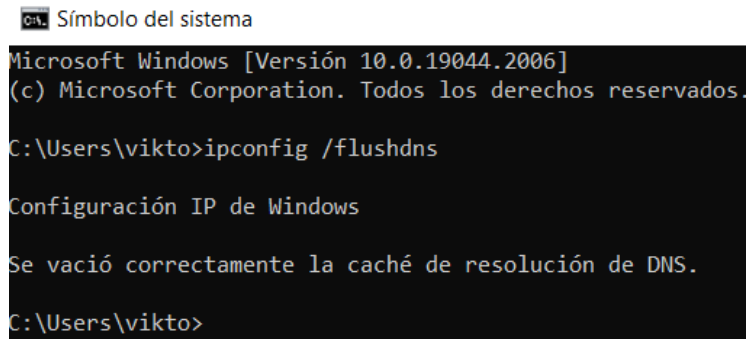
Escenario

Se utiliza Wireshark para capturar y analizar los mensajes DNS y HTTP.

1 Protocolo DNS

Borra la cache con el comando:

ipconfig /flushdns - Ya que el sistema operativo que utilizaré es Windows



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19044.2006]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\vikto>ipconfig /flushdns

Configuración IP de Windows

Se vació correctamente la caché de resolución de DNS.

C:\Users\vikto>
```

Fig. 1. Borramos la cache

Ejercicio 2. Realizar consultas DNS a través del comando NSLOOKUP

Esta parte ha sido realizada con mi máquina personal, accediendo a internet con el Wifi de eduroam desde la facultad.

Responder a las preguntas

4.- ¿Son las respuestas capturadas iguales a las obtenidas cuando las realiza el navegador?

Las respuestas son diferentes, hay mucho menos tráfico de peticiones y respuestas por *nslookup*.

8 4.535773	172.16.137.151	150.214.40.12	DNS	70 Standard query 0x0002 A www.uma.es
9 4.537636	150.214.40.12	172.16.137.151	DNS	112 Standard query response 0x0002 A www.uma.es CNAME ccumali.sci.uma.es A 150.214.40.12
10 4.539590	172.16.137.151	150.214.40.12	DNS	70 Standard query 0x0003 AAAA www.uma.es
11 4.540949	150.214.40.12	172.16.137.151	DNS	124 Standard query response 0x0003 AAAA www.uma.es CNAME ccumali.sci.uma.es AAAA 2001
29 13.589133	172.16.137.151	150.214.40.12	DNS	86 Standard query 0x0001 PTR 12.40.214.150.in-addr.arpa
30 13.595700	150.214.40.12	172.16.137.151	DNS	114 Standard query response 0x0001 PTR 12.40.214.150.in-addr.arpa PTR resolv2.uma.es

Fig. 5. Tráfico para www.uma.es

5.- ¿Cuál es la dirección IP y puerto del servidor DNS local al que envía la consulta?

En los 4 casos: - IP Origen: 172.16.137.83 - IP Destino: 150.214.40.12 - Puerto: 53

```
> Internet Protocol Version 4, Src: 172.16.137.151, Dst: 150.214.40.12
> User Datagram Protocol, Src Port: 60445, Dst Port: 53
```

Fig. 6. IP Origen, Destino y Puerto DNS

2 Protocolo HTTP

Tarea 1. Análisis de tráfico HTTP

Esta parte ha sido realizada con mi máquina personal, accediendo a internet con el Wifi de eduroam desde la facultad.

Responder a las preguntas

1.- ¿Con qué tipo de petición se envían el nombre y la contraseña?

El nombre y contraseña se envían con una petición de tipo **POST**, como se muestra en la captura de la pregunta 3.-.

2.- ¿Con qué tipo de petición se envían tus opiniones sobre la contraseña?

Las opiniones sobre los distintos campos del formulario se envían a través de una petición de tipo **GET**. Esto se aprecia en la captura que se encuentra bajo la pregunta 4.-.

3.- ¿Cómo se envían los datos con POST? (Formato del cuerpo)

Los datos con POST se envían de la siguiente manera:

1170	28.020058	172.16.137.151	150.214.108.58	HTTP	736	POST /RySD-Web/Formulario HTTP/1.1 (application/x-www-form-urlencoded)
1171	28.022082	150.214.108.58	172.16.137.151	HTTP	433	HTTP/1.1 200 (text/html)
1283	39.781670	172.16.137.151	150.214.108.58	HTTP	607	GET /RySD-Web/Formulario?uno=a+veces&dos=Practica&tres=Windows HTTP/1.1
1284	39.784152	150.214.108.58	172.16.137.151	HTTP	544	HTTP/1.1 200 (text/html)

> Frame 1170: 736 bytes on wire (5888 bits), 736 bytes captured (5888 bits) on interface 0	0000	00 0e ab 4f 2e 6e 9c 29 76 f8 39 6e 08 00 45 00	0000	00 0e ab 4f 2e 6e 9c 29 76 f8 39 6e 08 00 45 00
> Ethernet II, Src: IntelCor_f8:39:6e (9c:29:76:f8:39:6e), Dst: Cisco_4f:00:00 (00:00:00:00:00:00)	0010	02 d2 67 03 40 00 80 06 00 00 ac 10 89 97 96 d5	0010	02 d2 67 03 40 00 80 06 00 00 ac 10 89 97 96 d5
> Internet Protocol Version 4, Src: 172.16.137.151, Dst: 150.214.108.58	0020	6c 3a c3 12 1f 90 3a fa 78 91 87 6d 47 2c 50 18	0020	6c 3a c3 12 1f 90 3a fa 78 91 87 6d 47 2c 50 18
> Transmission Control Protocol, Src Port: 49938, Dst Port: 8080, Seq: 1	0030	02 01 3b 7d 00 00 50 4f 53 54 20 2f 52 79 53 44	0030	02 01 3b 7d 00 00 50 4f 53 54 20 2f 52 79 53 44
> Hypertext Transfer Protocol	0040	2d 57 65 62 2f 46 6f 72 6d 75 6c 61 72 69 6f 20	0040	2d 57 65 62 2f 46 6f 72 6d 75 6c 61 72 69 6f 20
> HTML Form URL Encoded: application/x-www-form-urlencoded	0050	48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20	0050	48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20
> Form item: "nombre" = "Viktor Yosava"	0060	31 35 30 2e 32 31 34 2e 31 30 38 2e 35 38 3a 38	0060	31 35 30 2e 32 31 34 2e 31 30 38 2e 35 38 3a 38
> Form item: "password" = "12345"	0070	30 38 30 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a	0070	30 38 30 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a

Fig. 7. Datos con POST

4.-¿Cómo se reescribe la URL para enviar los datos? (que parámetros se añaden) ¿Cómo se llaman los campos de los formularios?

Podemos observar que la URL se reescribe siguiendo el formato a continuación:

URL-"?"-Número de la Pregunta-"="-Respuesta-"-Número de la Pregunta-"="-Respuesta...

Añadiendo "" al final de cada Respuesta hasta la penúltima.

▼ Hypertext Transfer Protocol
▼ GET /RySD-Web/Formulario?uno=a+veces&dos=Practica&tres=Windows HTTP/1.1\r\n
> [Expert Info (Chat/Sequence): GET /RySD-Web/Formulario?uno=a+veces&dos=Practica&tres=Windows]
Request Method: GET
▼ Request URI: /RySD-Web/Formulario?uno=a+veces&dos=Practica&tres=Windows
Request URI Path: /RySD-Web/Formulario
▼ Request URI Query: uno=a+veces&dos=Practica&tres=Windows
Request URI Query Parameter: uno=a+veces
Request URI Query Parameter: dos=Practica
Request URI Query Parameter: tres=Windows
Request Version: HTTP/1.1

Fig. 8. URL reescrita para envío

Las **Key** serían *Número de la Pregunta*, y *Respuesta* se corresponde a **Value**.

Tarea 2. Desarrollo de una aplicación cliente en Java que interactúe con el recurso “ReverseServlet” en una petición GET.

```
private static void sendGET(String cadena) throws IOException {

    /* COMPLETAR con la url reescrita con el parámetro a enviar */
    String urlRewritten = GET_URL + cadena;

    /* COMPLETAR Crear objeto URL */
    URL obj = new URL(urlRewritten);

    /* COMPLETAR Crear conexión HTTP a la URL creada */
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();

    /* COMPLETAR Establecer el método de la petición GET */
    con.setRequestMethod("GET");
    con.setRequestProperty("Content-Length", "" + cadena.getBytes().length);

    /* COMPLETAR Establecer el encabezado User-Agent */
    con.setRequestProperty("User-Agent", USER_AGENT);

    /* COMPLETAR Obtener el código de estado de la respuesta */
    int responseCode = con.getResponseCode();

    System.out.println("GET Response Code :: " + responseCode);

    System.out.println("GET Connection-Header :: " + con.getHeaderFieldKey(3));

    /* COMPLETAR Obtener la fecha de la petición */
    String date = new Date(con.getDate()).toString();

    System.out.println("GET Date :: " + date);

    if (responseCode == HttpURLConnection.HTTP_OK) { // success

        /* COMPLETAR Leer cuerpo de la respuesta */
        InputStream cuerpo = con.getInputStream();
        BufferedReader in = new BufferedReader(new InputStreamReader(
            cuerpo));
        String inputLine;

        /* Lee el cuerpo de la respuesta */
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();

        String resultado="GET Respuesta :: "+response.toString();

        /* Muestra el cuerpo de la respuesta por pantalla */
        System.out.println(resultado);
    } else {
        System.out.println("GET request not worked");
    }
}
```

Los resultados del **GET** y el **POST** se encuentran en la última imagen de la **Tarea 3** de este apartado.

Tarea 3: Desarrollo de una aplicación Java que interactúe con el recurso “ReverseServlet” en una petición GET.

```
private static void sendPOST(String cadena) throws IOException {

    /* COMPLETAR Crear objeto URL */
    URL obj = new URL(POST_URL);

    /* COMPLETAR Crear conexión HTTP a la URL creada */
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();

    /* COMPLETAR Establecer el método de la petición POST*/
    con.setRequestMethod("POST");

    /* COMPLETAR Establecer el encabezado User-Agent */
    con.setRequestProperty("User-Agent",USER_AGENT);

    // For POST only - START

    /*COMPLETAR INDICAR QUE LA PETICIÓN HTTP TIENE CUERPO*/
    con.setDoOutput(true);

    /*COMPLETAR CREAR stream para ACCEDER AL CUERPO DE LA PETICIÓN HTTP*/
    OutputStream os = (OutputStream)con.getOutputStream();

    /*COMPLETAR escribir parametro y valor en el cuerpo de la petición*/
    os.write(cadena.getBytes());
    os.flush();
    os.close();
    // For POST only - END

    /* COMPLETAR Obtener el código de estado de la respuesta*/

    int responseCode = con.getResponseCode();

    System.out.println("POST Response Code :: " + responseCode);

    // El campo que contiene el Connection-Header es el 4º, o [3] en Java
    System.out.println("GET Connection-Header :: " + con.getHeaderFieldKey(3));

    /* COMPLETAR Obtener la fecha de la petición*/
    String date = new Date(con.getDate()).toString();

    System.out.println("POST Date :: " + date);

    if (responseCode == HttpURLConnection.HTTP_OK) { //success
        /* COMPLETAR Leer cuerpo de la respuesta */
        InputStream cuerpo = con.getInputStream();
        BufferedReader in = new BufferedReader(new InputStreamReader(
            cuerpo));
        String inputLine;

        /* Lee el cuerpo de la respuesta */
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();

        String resultado="POST Respuesta :: "+response.toString();

        /* Muestra el cuerpo de la respuesta por pantalla*/
        System.out.println(resultado);
    } else {
        System.out.println("POST request not worked");
    }
}
```

Descomentamos la parte del código correspondiente al POST y al GET y ejecutamos el código

```
/*Descomentar para tarea 2*/  
sendPOST(cadena);  
System.out.println("POST DONE");
```

Fig. 11. GET descomentado

```
/*Descomentar para tarea 3 */  
sendGET(cadena);  
System.out.println("GET DONE");
```

Fig. 12. POST descomentado

```
Enter text :  
Prueba Viktor Yosava  
POST Response Code :: 200  
GET Connection-Header :: Keep-Alive  
POST Date :: Tue Nov 29 09:02:48 CET 2022  
POST Respuesta ::  
POST DONE  
GET Response Code :: 200  
GET Connection-Header :: Keep-Alive  
GET Date :: Tue Nov 29 09:02:48 CET 2022  
GET Respuesta :: avasoY rotkiV abeurP  
GET DONE
```

Fig. 13. Resultados POST y GET