**UNIVERSIDAD NACIONAL DE CÓRDOBA**

**FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES.**

**ASIGNATURA**

Comunicaciones de Dato

**CARRERA**

INGENIERIA EN COMPUTACION

**Trabajo Práctico I: Capa de Aplicación**

**Integrantes:**

* Contrera, Iván Elias
* Alaniz Ferreyra, Alvaro
* Yépez Hinostroza, Franz Alexánder
* Malano,Leandro

Introducción

Para el siguiente trabajo práctico utilizaremos una máquina virtual en la que se montará un Servidor Web corriendo bajo el sistema operativo Ubuntu Server 16.04.3 LTS.

Seguido de definir el socket del programa interpretado por python, se hará uso de los métodos Get y Post del protocolo HTTP y se procederá a analizar el tráfico por medio del sniffer WireShark.

Del lado del cliente, se utilizará la herramienta netcat para hacer uso de la capa de transporte.

El servidor tiene la ip privada 192.168.1.45 (que puede variar según la red en la que esté conectado) y el puerto habilitado para el script de python es el 4000.

Al usar WireShark se filtró la búsqueda para que sólo mostrase del protocolo de aplicación HTTP.

Método Get sin parámetros

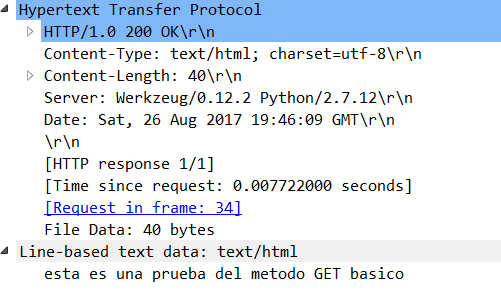
La siguiente imagen muestra lo que capturó WireShark en la sección de capa de aplicación ante la petición de Get del cliente.



Se observa en la sección de Host el socket del servidor.

En la sección User-Agent se puede apreciar que la petición fue hecha desde el Navegador Mozilla versión 5.0, corriendo desde un Sistema Operativo Windows.

La respuesta de la petición hecha se muestra a continuación:



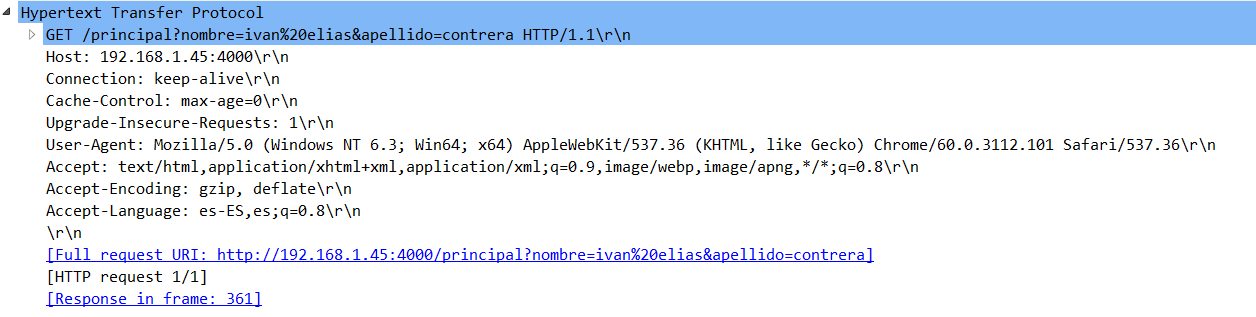
HTTP /1.0 200 = El código 200 hacer referencia a que fue una petición correcta.

Content-Length: 40 = el tamaño de la cadena que envía, en bytes.

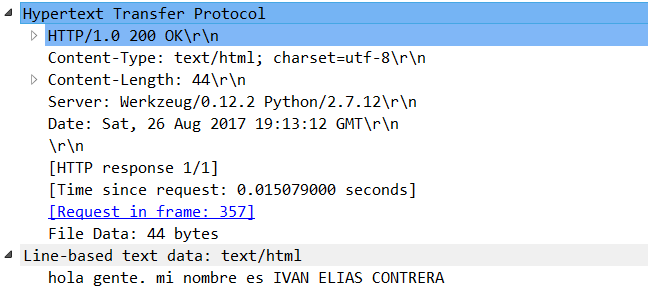
Se observa que el pedido se ubica en la trama 34, según la ordenación del WireShark.

Debajo de la sección correspondiente a la de capa de aplicación se observa la respuesta del servidor, en texto plano.

Método Get con parámetros



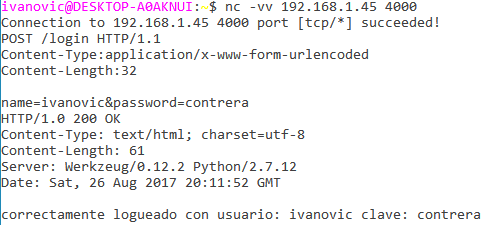
Se observa que en este caso los parámetros son pasado en texto plano sin ninguna clase de encriptación, por que que quedaría perfectamente visible por ejemplo en el historial del navegador.



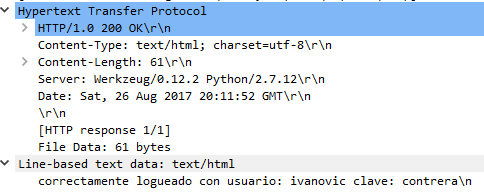
Método Post

El método Post del protocolo HTTP , a diferencia del Get, su contenido va en el body de la petición, por lo que no aparecería su contenido en la url de un navegador.

En la siguiente imagen se adjunta la petición y respuesta mediante la herramientas netcat:



Captura de la respuesta por WireShark



RESPUESTAS:

1. Otros metodos de peticion importantes son:

* HEAD: Pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
* PUT: suele utilizarse junto con herramientas de publicación web. Esto permite a un usuario cargar un objeto en una ruta específica (directorio) en un servidor web determinado. Las aplicaciones que necesitan cargar objetos en servidores web también emplean el método PUT.
* DELETE: permite a un usuario o a una aplicación borrar un objeto de un servidor web.
* TRACE: Este método solicita al servidor que en la respuesta meta todos los datos que reciba en el mensaje de petición. Se utiliza con fines de depuración y diagnóstico ya que el cliente puede ver lo que llega al servidor y de esta forma ver lo que añaden al mensaje los servidores intermedios.
* OPTIONS: Devuelve los métodos HTTP que el servidor soporta para un URL específico. Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso específico.
* CONNECT: Se utiliza para saber si se tiene acceso a un host, no necesariamente la petición llega al servidor.
* Ademas existen estos otros: PATCH, SEARCH, COPY, LOOK, UNLOOK, MOVE, MKCOL, PROPFIND, PROPPATCH, MERGE y LABEL.

1. JSON (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionarlos. una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías. Un uso común de JSON es para leer datos desde un servidor web, y mostrarlos en una página web.
2. API REST, básicamente, es un modelo de arquitectura de software muy vinculado desde sus origenes (siglo 20) al protocolo HTTP. Consiste en una serie de normas para mejorar la comunicación cliente-servidor (VER SI FALTA ALGO)

CONCLUSION: FDFGFGHFDHGDS