**Trabajo Práctico Final Sockets TCP/IP (Cliente - Servidor)**

\* [Bercovsky Nicolas](https://github.com/Bercovein)

\* [Marcos Bertolotti](https://github.com/MarcosBertolotti)

1. ¿Qué es un puerto?

Diversos programas TCP/IP pueden ejecutarse simultáneamente en Internet. Cada uno de estos programas funciona con un protocolo.

Por lo tanto, a cada una de estas aplicaciones puede serle asignada una dirección única en equipo, codificada en 16 bits llamada puerto a donde se dirigen los datos. Si se trata de una solicitud enviada a la aplicación, la aplicación se denomina aplicación servidor. Si se trata de una respuesta, entonces hablamos de una aplicación cliente.

1. ¿Cómo están formados los endpoints?

Los sistemas de ciberseguridad Endpoint, que se pueden adquirir como software o como un dispositivo dedicado, sirven para descubrir, gestionar y controlar los dispositivos que solicitan acceso a la red corporativa de nuestra empresa.

Los elementos que lo forman pueden incluir un sistema operativo, un cliente VPN y una licencia anti-virus.

Los dispositivos Endpoint trabajan bajo un modelo cliente/servidor en el que un servidor o gateway de gestión centralizada acoge el programa de seguridad, mientras que un programa cliente de acompañamiento se instala en cada dispositivo de red. En un sistema SaaS(Software as a Service) el servidor host y los programas de seguridad se mantienen a distancia por el proveedor de la solución, y cuando un cliente intenta conectarse a la red, el programa servidor valida las credenciales del usuario y escanea el dispositivo para asegurarse de que cumple con las políticas de ciberseguridad corporativas definidas antes de permitir el acceso a la red.

1. ¿Qué es un socket?

Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto. Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.

- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Para ello son necesarios los tres recursos que originan el concepto de socket:

* Un protocolo de comunicaciones, que permite el intercambio de octetos.
* Un par de direcciones del Protocolo de Red (Dirección IP, si se utiliza el Protocolo TCP/IP), que identifica la computadora de origen y la remota.
* Un par de números de puerto, que identifica a un programa dentro de cada computadora.

1. ¿A qué capa del modelo TPC/IP pertenecen los sockets? ¿Porque?

Los sockets se pueden ver como una interfaz de comunicación entre un proceso de la capa de aplicación y la capa de transporte del modelo TCP/IP.

1. ¿Cómo funciona el modelo cliente-servidor con TCP/IP Sockets?

El servidor debe negociar con su Sistema Operativo un puerto (casi siempre bien conocido) donde esperar las solicitudes. El servidor espera pasivamente las peticiones en un puerto bien conocido que ha sido reservado para el servicio que ofrece. El cliente también solicita, a su sistema operativo, un puerto no usado desde el cual enviar su solicitud y esperar respuesta. Un cliente ubica un puerto arbitrario, no utilizado y no reservado, para su comunicación.

En una interacción se necesita reservar solo uno de los dos puertos, asignados un identificador único de puerto para cada servicio, se facilita la construcción de clientes y servidores.

Los servidores por lo general son más difíciles de construir que los clientes pues aunque se implantan como programas de aplicación deben manejar peticiones concurrentes, así como reforzar todos los procedimientos de acceso y protección del sistema computacional en el que corren, y protegerse contra todos los errores posibles. El cliente y el servidor pueden interactuar en la misma máquina.

1. ¿Cuáles son las causas comunes por la que la conexión entre cliente/servidor falle?

Pueden presentarse varias fallas, que el cliente no encuentre al servidor, que la petición del cliente se pierda dentro del servidor o en la red , que el servidor se caiga al procesar un mensaje, que la respuesta del servidor a una petición se pierda o que el cliente haga crash al recibir un mensaje.

El diseño modular de una aplicación cliente/servidor permite que la aplicación sea tolerante a fallos, estos pueden ocurrir sin causar la caída de la aplicación completa. Por lo cual, uno o más servidores pueden fallar sin parar el sistema total mientras que los servicios proporcionados por los servidores caídos estén disponibles en otros servidores activos.

1. Diferencias entre sockets UDP y TCP

Las propiedades de un socket dependen de las características del protocolo en el que se implementan. Generalmente la comunicación con sockets se realiza mediante un protocolo de la familia TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet). Los dos más utilizados son: TCP (Protocolo de Control de Transmisión) y UDP (Protocolo de Datagrama de Usuario).

Cuando se implementan con el protocolo TCP, los sockets tienen las siguientes propiedades:

* Orientado a conexión.
* Se garantiza la transmisión de todos los octetos sin errores ni omisiones.
* Se garantiza que todo octeto llegará a su destino en el mismo orden en que se ha transmitido. Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información.

El protocolo UDP es un protocolo no orientado a la conexión. Sólo se garantiza que si un mensaje llega, llegue bien. En ningún caso se garantiza que llegue o que lleguen todos los mensajes en el mismo orden que se mandaron. Esto lo hace adecuado para el envío de mensajes frecuentes pero no demasiado importantes, como por ejemplo, mensajes para los refrescos (actualizaciones) de un gráfico.

1. Diferencia entre sync & async sockets?

Synchronous: La mensajería sincrónica involucra a un cliente que espera a que el servidor responda a un mensaje. La mensajería síncrona es una comunicación bidireccional. Es decir, el cliente envía un mensaje al servidor y el servidor recibe este mensaje y responde al cliente. El cliente no enviará otro mensaje hasta que reciba una respuesta del servidor.

Asynchronous: La mensajería asíncrona involucra a un cliente que no espera un mensaje del servidor. Un evento se utiliza para desencadenar un mensaje desde un servidor. Entonces, incluso si el cliente está inactivo, la mensajería se completará con éxito. La mensajería asíncrona significa que es una comunicación unidireccional y el flujo de comunicación es solo una forma.