Trabajo Práctico de Autenticación Básica

Alan G. Aquino R.

Mayo de 2025

Objetivo General

Desarrollar un sistema cliente-servidor básico que permita la autenticación de usuarios a través de una conexión de red, aplicando principios fundamentales de comunicación entre procesos, protocolos y gestión de sesiones.

Requisitos mínimos

Servidor:

1. Escuchar peticiones de conexión por un puerto determinado.

2. Recibir credenciales (usuario y contraseña) desde el cliente.

3. Validar credenciales contra un archivo local (usuarios.txt) o una estructura de

datos predefinida.

4. Enviar respuesta al cliente indicando si el acceso fue aceptado o denegado.

Cliente:

1. Conectarse al servidor remoto (o local) por IP y puerto.

2. Enviar nombre de usuario y contraseña.

3. Recibir y mostrar la respuesta del servidor (éxito o error).

Formato del archivo usuarios.txt (en el servidor):

usuario1:clave123

juan:pass456

admin:adminpass

Extras opcionales:

1. Permitir múltiples conexiones concurrentes (hilos o procesos).

2. Implementar un sistema de logs de conexiones.

3. Permitir registro de nuevos usuarios.

4. Cifrar la contraseña antes de enviarla (hash o cifrado básico).

5. Uso de sockets UDP como alternativa (para analizar limitaciones).

Desarrollo

El desarrollo del sistema se realizó de forma progresiva, aplicando los contenidos de la materia de Sistemas Operativos en el contexto de una arquitectura cliente-servidor. El objetivo fue crear una aplicación robusta, funcional y didáctica, que permitiera gestionar la autenticación de usuarios mediante conexión en red, con posibilidad de ampliación mediante extras opcionales.

Etapa 1: Elección del entorno y lenguaje

Inicialmente se intentó implementar el sistema en C++ utilizando Dev-C++, luego Visual Studio 2022 y finalmente Code::Blocks, pero se encontraron dificultades relacionadas con la configuración de Winsock en sistemas Windows y problemas de compatibilidad. Tras varias pruebas, se optó por utilizar Python en combinación con Visual Studio Code, por su facilidad de uso, velocidad de desarrollo y claridad para trabajar con sockets.

Etapa 2: Sistema básico (TCP, login)

Se desarrolló una versión básica que:

1. Incluye un servidor TCP, que escucha en un puerto y valida credenciales contra un archivo de texto (usuarios.txt).

2. Incluye un cliente TCP, que solicita usuario y contraseña, los envía al servidor, y muestra el resultado.

3. Usa el protocolo TCP para asegurar entrega confiable.

Una vez establecida la conexión básica, se probó el sistema con credenciales válidas e inválidas, confirmando su funcionamiento.

Etapa 3: Implementación progresiva de los 5 extras opcionales

1. Múltiples conexiones (multithreading):

El servidor fue modificado para aceptar múltiples clientes concurrentemente, utilizando la biblioteca threading, lo que permite que varios usuarios accedan sin bloquear al resto.

2. Registro de nuevos usuarios:

Se amplió el protocolo para permitir al cliente elegir entre "Iniciar sesión" o "Registrarse". El servidor valida que el usuario no exista y, de ser así, lo agrega al archivo usuarios.txt.

3. Logs de actividad:

Cada conexión, intento de login o registro es registrado con fecha, IP, tipo de acción y resultado en el archivo logs.txt. Esto brinda trazabilidad al sistema.

4. Cifrado de contraseñas (hash SHA-256):

El cliente aplica hash SHA-256 a las contraseñas antes de enviarlas, y el servidor compara hashes, nunca almacenando contraseñas reales.

5. Versión UDP:

Se desarrolló un cliente y un servidor paralelos utilizando sockets UDP. Esta implementación permite analizar la diferencia entre comunicación con y sin conexión (TCP vs UDP), evidenciando las limitaciones de UDP (falta de fiabilidad, sin control de flujo, sin confirmación de conexión).

Etapa 4: Capturas, documentación y validación

Durante el desarrollo se realizaron numerosas pruebas de conexión, simulación de errores, y se capturaron imágenes para respaldar el funcionamiento.

Resumen

El trabajo permitió poner en práctica conocimientos clave sobre comunicación entre procesos, sockets, estructuras de control, archivos, manejo de errores y seguridad básica. Además, se aprendió a tomar decisiones tecnológicas, resolver errores reales, y a aplicar conceptos teóricos a problemas concretos.