Universidad del Valle de Guatemala

Facultad de ingeniería

Redes

Catedrático: Jorge Yass

Proyecto MCP

Diego Linares 221256

Repositorios

- Cliente y Host: https://github.com/DiegoLinares11/MCP-Host
- Servidor Remoto (Supabase Admin Helper): https://github.com/DiegoLinares11/MCP-Host/tree/main/src/remoteMCP
- MCP local: https://github.com/DiegoLinares11/Proyecto1-MCP
- Demostración de funcionamiento: https://youtu.be/UbPiTl54kO4

MCP Local – Host

El MCP Local implementado funciona como cliente principal que orquesta:

- Herramientas de FileSystem y Git.
- Herramientas de Supabase Admin Helper como crear usuarios, enviar magic links, listar usuarios, actualizar metadata, etc.
- Herramientas de SQLScout para explicar, diagnosticar y optimizar consultas SQL.

El host expone comandos de consola como:

- :tools [FS|Git|SQLScout]
- :load <ruta.sql>
- :explain <SQL>
- :diagnose <SQL>
- :optimize <SQL>
- :apply <DDL>

De esta manera, el host sirve como interfaz natural en lenguaje humano para invocar distintas operaciones.

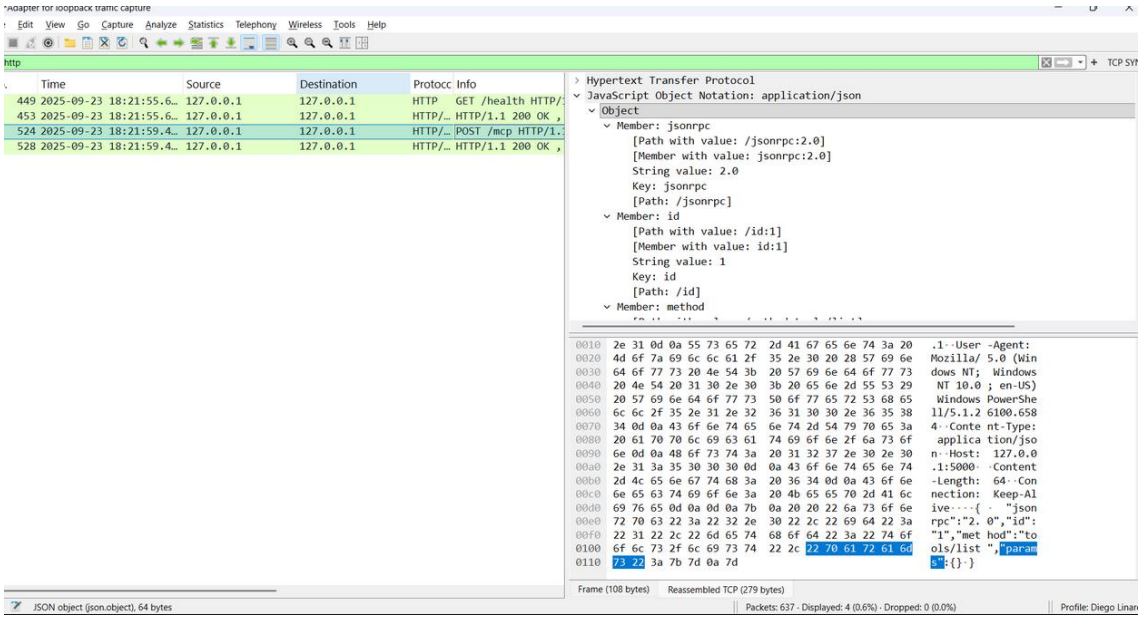
MCP Remoto – Supabase Admin Helper

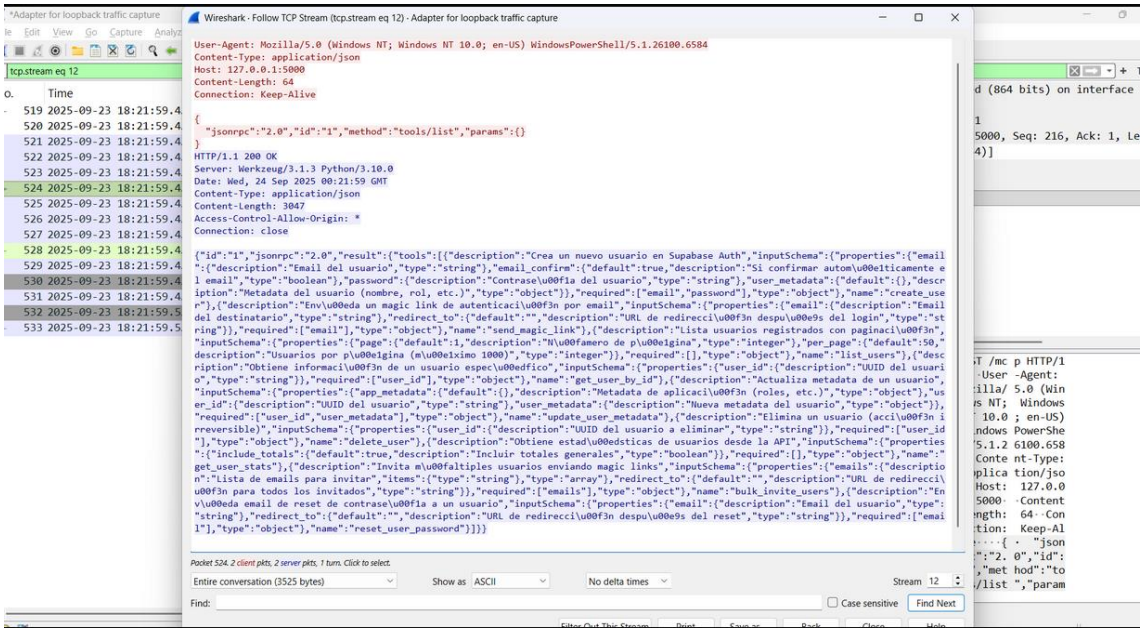
El servidor remoto está hecho en Flask (Python) y se ejecuta en el puerto 5000. Expone herramientas vía JSON-RPC:

Herramienta	Parámetros	Descripción
create_user	email, password	Crea un usuario en Supabase.
list_users	–	Lista todos los usuarios registrados.

get_user_by_id	user_id	Recupera la información completa de un usuario.
update_user_metadata	user_id, user_metadata	Actualiza metadata del usuario.
reset_user_password	email	Envía correo de reseteo de contraseña.
send_magic_link	email	Envía enlace mágico de autenticación.
delete_user	user_id	Elimina un usuario de manera irreversible.
get_user_stats	rango	Estadísticas de usuarios (día, semana, mes).
bulk_invite_users	emails[]	Invita múltiples usuarios con magic link.

Análisis de Wireshark





1. Paquetes de inicialización:
    a. El cliente envía un POST /mcp con { "method":"tools/list" }.
    b. El servidor remoto responde con la lista de herramientas disponibles en Supabase.
2. Paquetes de solicitud:
    a. Ejemplo: update_user_metadata enviado como JSON con parámetros { "phone": "3046" }.
    b. Visible en Wireshark con contenido JSON en claro (porque corre en localhost:5000 sin TLS).
3. Paquetes de respuesta:
    a. El servidor responde con la metadata actualizada (ej. {"phone":"3046"}).
    b. Esto se visualizó en Follow TCP Stream, donde aparece la conversación completa request-response.
4. Flujo de capas:
    a. Aplicación: JSON-RPC sobre HTTP.
    b. Transporte: TCP puerto 5000 con ACKs y segmentación.
    c. Enlace: direccionamiento MAC en la interfaz loopback virtual.
    d. Física: tráfico representado en software, normalmente señales eléctricas en un cable.

Conclusiones y comentarios

- El MCP Host permite interactuar en lenguaje natural con Supabase, Git y FileSystem sin necesidad de escribir comandos bajos.
- Con Wireshark, se comprobó cómo viajan las solicitudes JSON-RPC, confirmando la comunicación cliente-servidor.

Sobre el uso de IA

En este proyecto se apoyó de ChatGPT para:

- Estructurar el flujo de comandos del host.
- Definir la interfaz JSON-RPC del servidor remoto.
- Guiar en la configuración y captura de paquetes en Wireshark.

Referencias

- Model Context Protocol: https://modelcontextprotocol.io/docs/learn/architecture
- Supabase Auth API: https://supabase.com/docs
- Wireshark Filters Guide: https://wiki.wireshark.org/DisplayFilters
- Ejemplos MCP GitHub: https://github.com/modelcontextprotocol