SISTEMAS DE INFORMAÇÃO 2025/1 - Redes de Computadores I
PROFESSOR: Douglas Chagas
ALUNOS: Gisele Oliveria e João Víttor Costa Leite
DATA: 09/06/2025
DATA DE ENTREGA PREVISTA: 11/06/2025

Relatório do projeto - Varredura Passiva em Redes IEEE 802.11 a/b/g/n/ac

Visão Geral do Projeto

Este projeto consiste na criação de um sistema de varredura de redes Wi-Fi utilizando o comando `netsh wlan show networks mode=bssid` do Windows. As informações coletadas são armazenadas em um banco de dados PostgreSQL hospedado em uma máquina virtual (VM) Ubuntu 22.04 utilizando o VirtualBox.

A aplicação Java foi executada na máquina local com sistema operacional Windows, sendo necessária a instalação do driver JDBC do PostgreSQL (pgJDBC).

Ambiente de Execução

Host (Windows):

- Sistema Operacional: *Windows 10*
- Comando de varredura: `netsh wlan show networks mode=bssid`
- Driver JDBC: *`postgresql-42.7.5.jar`*

VM (VirtualBox):

- Sistema Operacional: Ubuntu Server 22.04
- Banco de dados: PostgreSQL

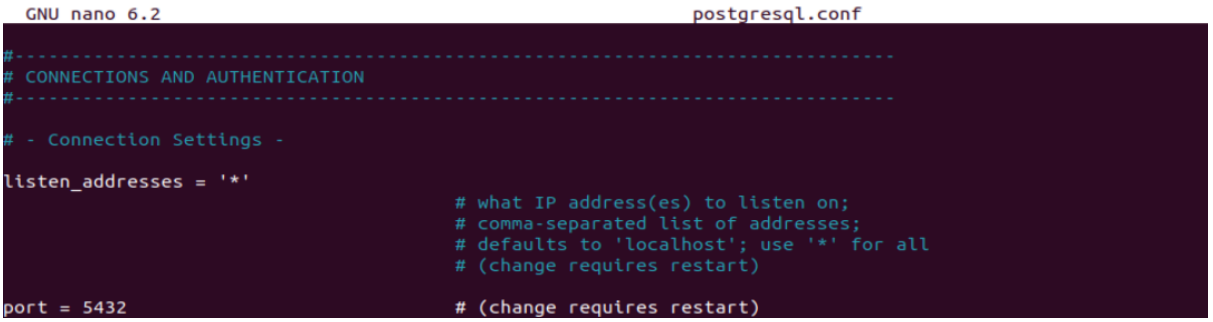
Configuração do PostgreSQL na VM Ubuntu

Ao instalar o PostgreSQL na VM Ubuntu 22.04, foi necessário habilitar o acesso externo para que o script Java na máquina Windows pudesse se conectar via JDBC.

Configuração do arquivo postgresql.conf

Localização:
/etc/postgresql/14/main/postgresql.conf

Ação Realizada:
- #listen_addresses = 'localhost' (removido)
+ listen_addresses = '' (adicionado)*

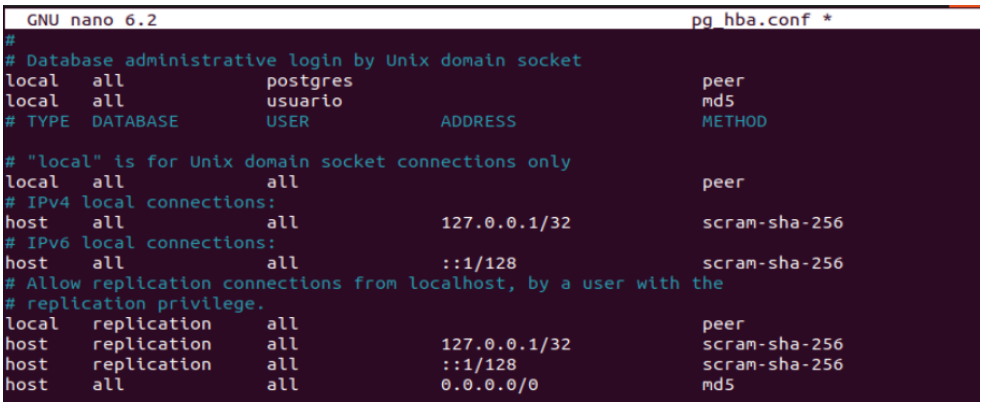


Esta modificação permite que o servidor PostgreSQL aceite conexões de qualquer IP.

Configuração do arquivo pg_hba.conf

Localização:
/etc/postgresql/14/main/pg_hba.conf

Linha adicionada ao final do arquivo:
host all all 0.0.0.0/0 md5

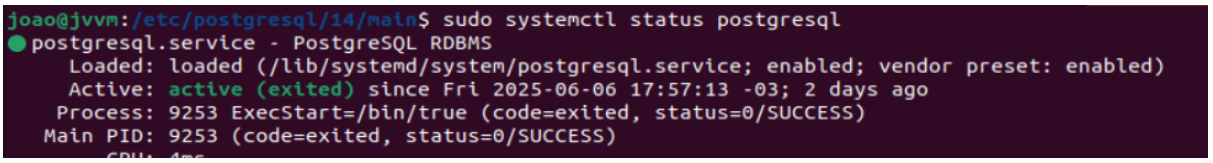


Essa linha define que qualquer IP pode acessar qualquer banco com autenticação por senha (*md5*), o que foi necessário para acesso remoto do cliente JDBC.

Reinício do serviço PostgreSQL

Para aplicar as mudanças, o serviço PostgreSQL foi reiniciado com:
sudo systemctl restart postgresql

Verificou-se que o serviço estava ativo com:
sudo systemctl status postgresql



Estrutura do Banco de Dados

A tabela criada no PostgreSQL se chama *wifi_scan* e possui os seguintes campos:

```
CREATE TABLE wifi_scan (
    id SERIAL PRIMARY KEY,
    ssid TEXT,
    mac_ap TEXT,
    quality_link INT,
    signal_level INT,
    channel INT,
    frequency REAL,
    last_beacon_ms INT,
    beacon_interval INT,
    wifi_security TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Lógica da Aplicação Java

A aplicação Java executa uma varredura a cada 60 segundos e armazena os resultados no banco. Abaixo, os trechos mais relevantes:

1. Execução do comando e leitura da saída

```
Process process = Runtime.getRuntime().exec("netsh wlan show networks
mode=bssid");
BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
```

2. Interpretação das linhas da saída

```
if (line.startsWith("SSID") && line.contains(" : ")) {
    ssid = line.split(" : ")[1].trim();
} else if (line.startsWith("Autenticação")) {
    security = line.split(" : ")[1].trim();
} else if (line.startsWith("BSSID")) {
    bssid = line.split(" : ")[1].trim();
} else if (line.startsWith("Sinal")) {
    signal = Integer.parseInt(line.split(" : ")[1].replace("%", "").trim());
} else if (line.startsWith("Canal")) {
    channel = Integer.parseInt(line.split(" : ")[1].trim());
```

3. Conversão de canal para frequência

```
private static double channelToFrequency(int channel) {
    if (channel >= 1 && channel <= 14) return 2407 + channel * 5;
    else return 5000 + (channel * 5);
}
```

4. Inserção no banco de dados

```
PreparedStatement ps = conn.prepareStatement(
    "INSERT INTO wifi_scan (ssid, mac_ap, quality_link, signal_level,
channel, frequency, last_beacon_ms, beacon_interval, wifi_security) " +
    "VALUES (?,?,?,?,?,?,?,?,?)"
);
ps.setString(1, ssid);
ps.setString(2, bssid);
ps.setInt(3, signal);
ps.setInt(4, signal);
ps.setInt(5, channel);
ps.setDouble(6, channelToFrequency(channel));
ps.setInt(7, 0); // last_beacon_ms fictício
ps.setInt(8, 0); // beacon_interval fictício
ps.setString(9, security);
ps.executeUpdate();
```

5. Delay entre varreduras

```
Thread.sleep(60000); // 60 segundos
```

Considerações Finais

- As informações de segurança da rede (campo *`wifi_security`*) estavam sendo ignoradas devido à lógica de coleta antes do `BSSID`, mas isso foi corrigido.

- Os campos *`last_beacon_ms`* e *`beacon_interval`* não estão disponíveis com o comando `netsh`, então foram mantidos com valor `0`.

- O sistema funciona de forma cíclica, fazendo varreduras automáticas e armazenando em tempo real os dados de Wi-Fi visíveis.