Secure TCP String Search Server

This project implements a secure, multithreaded TCP server in Python that performs exact string match lookups in a large text file. It supports SSL/TLS encryption, dynamic configuration, unit testing, detailed logging, and Linux daemonization for production deployments.

Features

SSL/TLS encryption with configurable certificates

Dynamic configuration reload on each query

Multithreaded client connection handling

High-performance exact line matching (tested with files up to 250,000 lines)

Unit tested using `unittest`

Ready for deployment as a Linux daemon

Fully configurable via `server/config.cfg`

Deploying as a Linux Daemon (Systemd)

Step 1: Create a Systemd Service File

Open a new service file:

```bash
sudo nano /etc/systemd/system/tcp-server.service
```

Paste the following configuration:

```ini
```

```
[Unit]

Description=Secure TCP String Search Server

After=network.target


[Service]

EnvironmentFile=/tcp_server_project/.env

ExecStart=/usr/bin/python3 /tcp_server_project/server/server.py

WorkingDirectory=/tcp_server_project/

Restart=always

User=nobody


[Install]

WantedBy=multi-user.target
```

Note: Replace `/tcp_server_project/` with the absolute path to your project directory.

Configuration

All configuration options are defined in `server/config.cfg`.

To override the search file path via an environment variable:

```bash
export SEARCH_FILE_PATH=/root/200k.txt
```

Generating SSL Certificate and Key

To enable SSL/TLS, generate a self-signed certificate and private key using OpenSSL.

Step 1: Install OpenSSL

```bash
sudo apt install openssl
```

Step 2: Create an OpenSSL Configuration File

Navigate to your project directory:

```bash
cd tcp_server_project
```

Create a new OpenSSL config file:

```bash
nano openssl.cnf
```

Paste the following content:

```ini
[ req ]
default_bits       = 2048
prompt             = no
default_md         = sha256
distinguished_name = dn
```

```
[ dn ]
C  = US
ST = State
L  = City
O  = Organization
OU = Unit
CN = localhost
```

Step 3: Generate Certificate and Private Key

```bash
openssl req -new -x509 -days 365 -nodes    -out server/server-cert.pem    -keyout server/server-key.pem
-config openssl.cnf
```

This will:

- Create a self-signed certificate: `server/server-cert.pem`
- Create a private key: `server/server-key.pem`

Installation & Setup

Step 1: Install Dependencies

```bash
sudo apt update
sudo apt install python3-pip -y
pip install matplotlib
```

Starting the Server

Option 1: Manually Start the Server

```bash
python3 server/server.py
```

Option 2: Enable and Start with Systemd

```bash
sudo systemctl daemon-reexec
sudo systemctl enable tcp-server
sudo systemctl start tcp-server
```

Check the server status:

```bash
sudo systemctl status tcp-server
```

Running Tests

To run unit tests:

```bash
python3 -m unittest discover -s tests
```

Connecting a Client

To connect a client to the server:

```bash
python3 client/client.py --host 135.181.96.160 --port 44445
```

Running the Performance Benchmark

To execute the benchmark script:

```bash
python3 report/benchmark.py
```