ECS 152A Project 1 Report

Brandon Ng Joon Hoe 924335301
Kevin Shefkiu 924335327

Part 2: Proxy Server

---

1. Explanation of Initial Attempts

Our first version of the proxy was a simple message forwarder. It read JSON data from a client and forwarded it to the server. The basic version worked well for forwarding messages, however, it wasn't able to send a response back. In particular, we noticed that sometimes the ports on our computer "bugged" and threw an error. We later discovered that our local network wasn't allowing port forwarding on port 7000 because it was already occupied by another process. After several attempts and some web research, we were able to fix it by killing the process running on those ports. Another issue we faced was how to unpack a JSON in python. We eventually found out that our client was sending incomplete data to the proxy, which prevented it from establishing a connection with the server because we were sending the wrong data.

2. Expected Results

In simple steps, the proxy should do the following:

- Accept a message from a client and read the contents.
- Check if the server is blocklisted.
- If not, forward the message to the server and then return the server's response back to the client.
- If the IP is blocked, send an alert to the client and do not forward the message to the server.

3. Issues Faced

During the development, we encountered a few challenges:

- Handling the JSON input and output. At first the we had issues because our program crashed because the data we sent was invalid
- Establishing the connection between client and server through the proxy. We had issues handling the ports.
- Implementing the check for blocklisted IP.

4. How to run the code

Open your terminal and navigate to the directory where the 3 files are located. Run the following commands in this exact order:

```
python server_Kevin_924335327_Brandon_924335301.py
```

```
python proxy_server_Kevin_924335327_Brandon_924335301.py
```

```
python client_Kevin_924335327_Brandon_924335301.py
```

Make sure that the ports on the program are set up correctly and that your network allows connection on those ports.

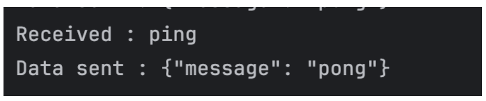
5. Implementation

The proxy listens on a specific IP and PORT for incoming connections. Once the client connects to the proxy, it sends a JSON data containing the following data: SERVER_IP, SERVER_PORT, MESSAGE. The proxy then receives the data and, before forwarding it to the server, extracts the data and checks if the SERVER_IP is on the blacklist. If it is, he doesn't forward the message and instead sends an error back to the client.

If the IP is valid, the proxy creates a new connection with the server and forwards the message. Once the server receives the message, it unpacks it and sends a response back to the proxy, which forwards it back to the client.

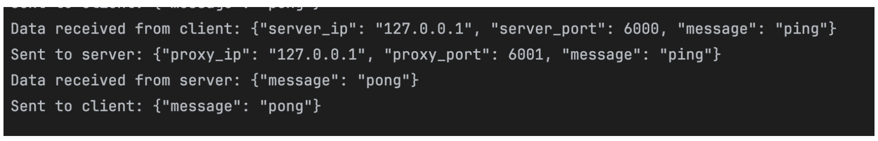
This design ensures that the server never sees the client's IP and PORT, as required by the guidelines.

Server



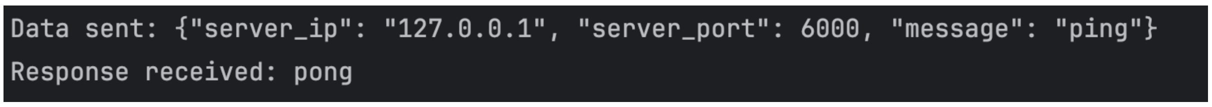
```
Received : ping
Data sent : {"message": "pong"}
```

Proxy



```
Data received from client: {"server_ip": "127.0.0.1", "server_port": 6000, "message": "ping"}
Sent to server: {"proxy_ip": "127.0.0.1", "proxy_port": 6001, "message": "ping"}
Data received from server: {"message": "pong"}
Sent to client: {"message": "pong"}
```

Client



```
Data sent: {"server_ip": "127.0.0.1", "server_port": 6000, "message": "ping"}
Response received: pong
```