Ihor Nikonov

User guide of Master Node

Introduction

This document explains how to set up and run the **master node** for the proxy system. The master node is responsible for managing slave connections, coordinating SOCKS5 proxy requests, and load balancing.

Prerequisites

1. System Requirements

The master node can be run on any platform that supports **Rust**. However, Ubuntu 20.04+ is recommended for optimal performance.

2. Dependencies

Ensure that Rust is installed on the system:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

source ~/.cargo/env

Getting the Binary

1. Clone the Git repository and navigate to the master node directory.

2. Build the binary:

cargo build –release

This will create the master node binary (net-relay) in the ./target/release directory.

Running the Master Node

Use the following **command-line options** to configure and run the master node:

Command-Line Parameters

Option	Description	Example
-t, --transfer	**Required**: Address for slave connections (e.g., IP:Port)	-t 0.0.0.0:8000
-s, --transfer	**Required**: Address for SOCKS5 client connections (e.g., IP:Port)	-s 0.0.0.0:1080
-p, --proxy_mode	Proxy mode: **stick (1)** or **nonstick (2)**. Default: nonstick	-p stick
-l, --allowed-locations	Comma-separated list of allowed slave locations (countries).	-l “US, CA, DE”
-v, --verbosity	Log verbosity level: **trace**, **debug**, **info**. Default: info.	-v debug

Usage Example

./target/release/net-relay -t 0.0.0.0:8000 -s 0.0.0.0:1080

Description of parameters

`-t` (Master Address)

- This is the address where slaves connect to register with the master node.
- **Example**: `-t 0.0.0.0:8000` listens on all interfaces at port 8000.

`-s` (SOCKS5 Server Address)

- This is the address where the master listens for incoming SOCKS5 connections from clients.
- **Example**: `-s 0.0.0.0:1080` listens on all interfaces at port 1080.

`-p` (Proxy Mode)

- **`stick` (1)**: Ensures requests from the same client always go to the same slave. **IP hashing** ensures clients with the same IP are consistently routed to the same slave.
- **`nonstick` (2)**: Distributes requests across available slaves for load balancing. Uses **weighted round-robin** to assign clients based on the load on each slave.

`-l` (Allowed Locations)

- Restricts slave nodes to specific geographic locations (e.g., countries).
- **Example**: `-l "US,CA"` allows only slaves in the United States and Canada to connect.

`-v` (Verbosity)

- Sets the logging level for debugging and monitoring.
- **Example**: `-v debug` enables detailed logs.
- **`debug`**: Logs traffic flow between clients and slaves.
- **`trace`**: Logs detailed internal operations.

TroubleShooting

1. Firewall Configuration

To deploy the master node on a production server, open the necessary ports in the firewall:

```
sudo ufw allow 8000/tcp
sudo ufw allow 1080/tcp
sudo ufw allow 9090/tcp
```

These ports are for following ones:

- Port **8000**: Slave connections.
- Port **1080**: SOCKS5 client connections.
- Port **9090**: Web monitoring interface.

2. Monitoring Slave Sessions

The master node provides a simple web interface for monitoring at:

```
http://[serverIP]:9090
```

This page displays:

- **Total slave sessions** (including disconnected ones).
- **Active sessions**.
- **Disconnected sessions**.

3. Handling `Too Many Open Files` Errors

If you encounter the `too many open files` error on a new server, increase the file descriptor limit:

```
ulimit -n 65535
```

To make this change permanent, update the `/etc/security/limits.conf` file.