Project README: Reverse Proxy Setup with FRP for AX8 Thermal Camera

Introduction

This README provides an overview of the reverse proxy setup for our project using FRP (Fast Reverse Proxy). The FRP tool is utilized to bypass complex router and local server configurations, enabling the full functionality of our AX8 thermal imaging camera running on a Raspberry Pi to be projected onto the public internet. This allows users to access the camera through our web application hosted on our server.

You can find FRP on its GitHub page: FRP GitHub Repository. FRP is a widely-used open-source project in production environments across many enterprises. We opted for FRP over the more well-known ngrok because FRP, being open-source, offers greater flexibility. Additionally, ngrok tends to flag high-frequency image streams as DDoS attacks, whereas FRP handles them effectively, making it a better choice for our needs.

It's important to note that the README on FRP's GitHub page can be somewhat confusing, leading to configuration bugs across different versions, including the latest one. After testing various versions, we chose FRP 0.48.0 for its relative stability and extensive community support. Below are the configurations for both the FRP client and server.

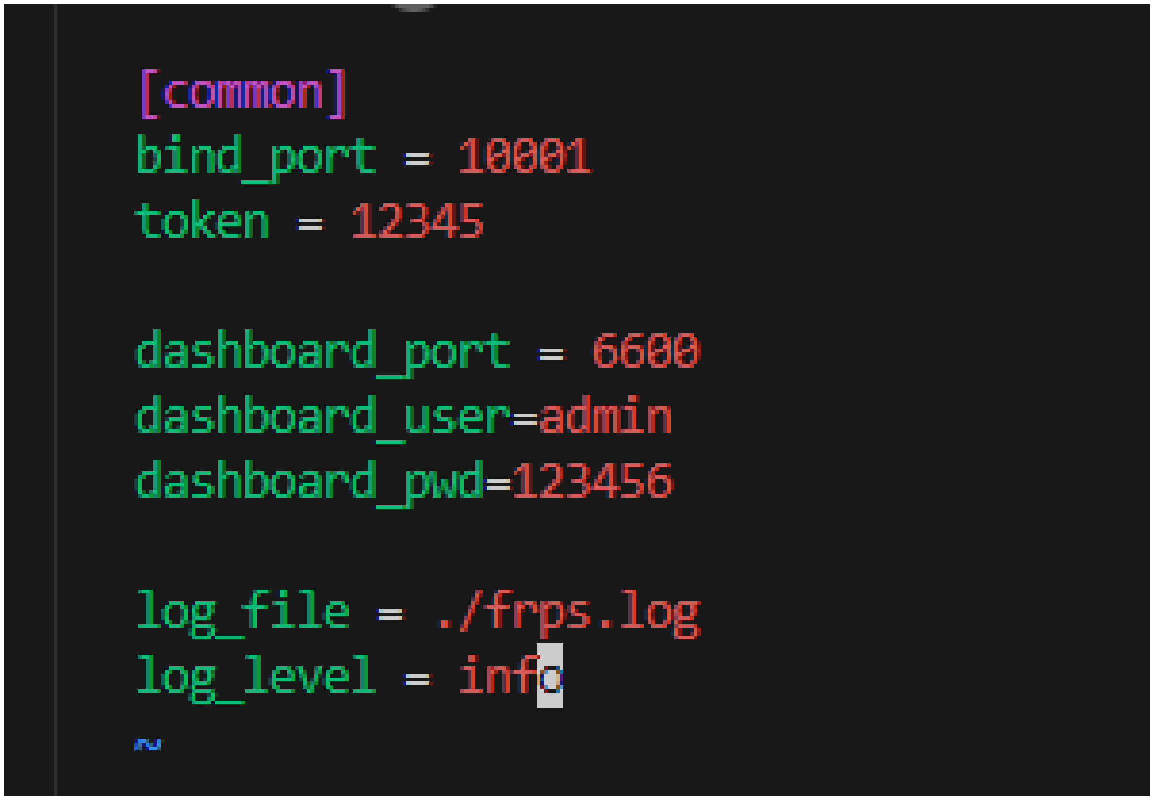
FRP Server Setup

Step 1: Download and Extract FRP

```
wget https://github.com/fatedier/frp/releases/download/v0.48.0/frp_0.48.0_linux_amd64.tar.gz
tar -zxvf frp_0.48.0_linux_amd64.tar.gz
```

Step 2: Configure the FRP Server (`frps.ini`)

Within the extracted `frp` directory, locate the `frps.ini` file and update it as follows:



```
[common]
bind_port = 10001
token = 12345

dashboard_port = 6600
dashboard_user=admin
dashboard_pwd=123456

log_file = ./frps.log
log_level = info
~
```

The `bind_port` is the port on your server that you want the client to connect to.

Step 3: Start the FRP Server

```
./frps -c frps.ini
```

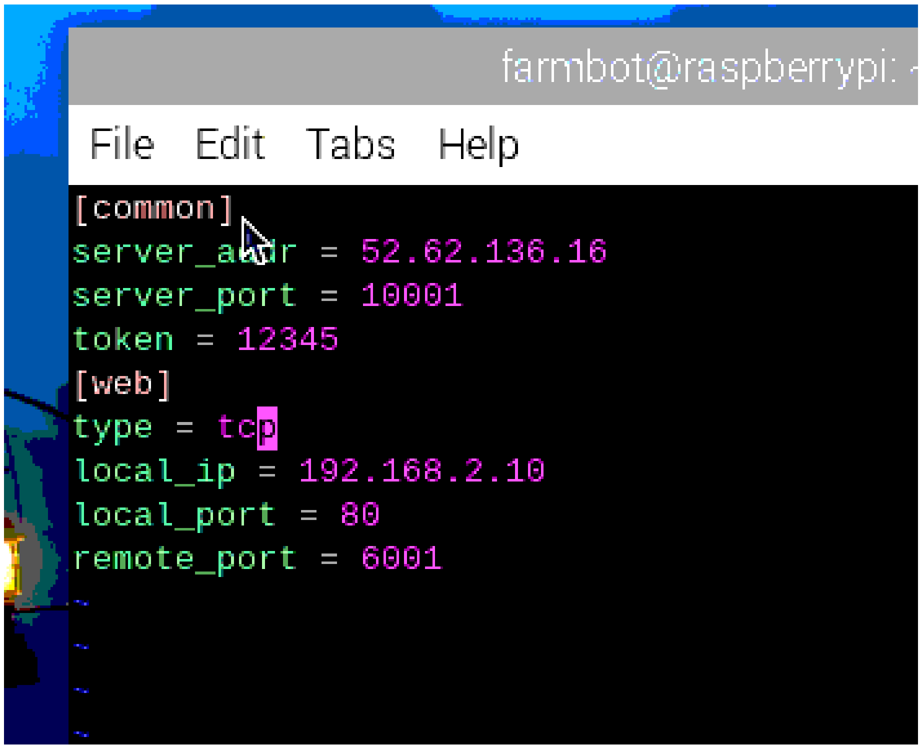
FRP Client Setup

Step 1: Download and Extract FRP on the Raspberry Pi

```
wget https://github.com/fatedier/frp/releases/download/v0.48.0/frp_0.48.0_linux_amd64.tar.gz
tar -zxvf frp_0.48.0_linux_amd64.tar.gz
```

Step 2: Configure the FRP Client (`frpc.ini`)

Within the extracted `frp` directory on the Raspberry Pi, locate the `frpc.ini` file and update it as follows:



```
[common]
server_addr = 52.62.136.16
server_port = 10001
token = 12345
[web]
type = tcp
local_ip = 192.168.2.10
local_port = 80
remote_port = 6001
~
~
~
~
```

`server_addr` and `server_port` should match the server’s IP address and port.

`token` should match the token set on the server for authentication.

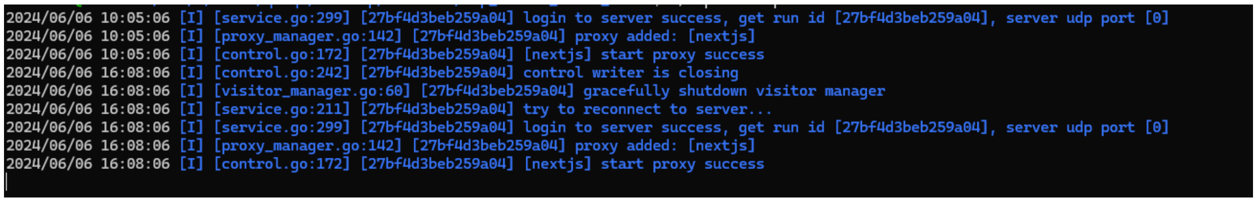
The `[web]` section specifies the service we want to expose on the public internet.

Step 3: Start the FRP Client

```
./frpc -c frpc.ini
```

Verifying the Setup

After starting both the server and client, verify the connection through the output logs. If the setup is successful, the output should indicate a successful connection between the client and server, enabling remote access to the camera service running on the Raspberry Pi.



```
2024/06/06 10:05:06 [I] [service.go:299] [27bf4d3beb259a04] login to server success, get run id [27bf4d3beb259a04], server udp port [0]
2024/06/06 10:05:06 [I] [proxy_manager.go:142] [27bf4d3beb259a04] proxy added: [nextjs]
2024/06/06 10:05:06 [I] [control.go:172] [27bf4d3beb259a04] [nextjs] start proxy success
2024/06/06 16:08:06 [I] [control.go:242] [27bf4d3beb259a04] control writer is closing
2024/06/06 16:08:06 [I] [visitor_manager.go:60] [27bf4d3beb259a04] gracefully shutdown visitor manager
2024/06/06 16:08:06 [I] [service.go:211] [27bf4d3beb259a04] try to reconnect to server...
2024/06/06 16:08:06 [I] [service.go:299] [27bf4d3beb259a04] login to server success, get run id [27bf4d3beb259a04], server udp port [0]
2024/06/06 16:08:06 [I] [proxy_manager.go:142] [27bf4d3beb259a04] proxy added: [nextjs]
2024/06/06 16:08:06 [I] [control.go:172] [27bf4d3beb259a04] [nextjs] start proxy success
```

Accessing the Camera

Upon completing these configurations, users can access the full range of Raspberry Pi camera services through the web application. This setup acts as the core middleware of our project, effectively bridging the gap between hardware and software components.