master 3 Branches 3 Tags Go to file Go to file Code

thirtythreeforty README: Development has moved 186d580 · 5 months ago 473 Commits

.cargo	Swapping to newer config format	5 months ago
.github	Use aptitude over apt for resolving con…	5 months ago
crates/core	Cargo fmt	5 months ago
dissector	Add some copyright notices at the entr…	last year
docker	Implement Docker image based on Alp…	4 years ago
docs	use new command 'neolink rtsp' in docs	2 years ago
src	Add some copyright notices at the entr…	last year
.dockerignore	Implement Docker image based on Alp…	4 years ago
.gitattributes	Git: Do not attempt to diff/merge Carg…	4 years ago
.gitignore	Swapping to newer config format	5 months ago
Cargo.lock	Bump bumpalo from 3.11.0 to 3.16.0	5 months ago
Cargo.toml	Update Crossbeam	2 years ago
Dockerfile	Updates to docker	3 years ago
LICENSE	Add README and release under AGP…	4 years ago
README.md	README: Development has moved	5 months ago
build.rs	Update build.rs	5 months ago
sample_config.toml	Read me updates	3 years ago

README AGPL-3.0 license

Development has moved

Neolink is now maintained at https://github.com/QuantumEntangledAndy/neolink as I no longer have time to dedicate to development. Please direct your thanks and bug reports over there!

Neolink

CI passing

Neolink is a small program that acts as a proxy between Reolink IP cameras and normal RTSP clients. Certain cameras, such as the Reolink B800, do not implement ONVIF or RTSP, but instead use a proprietary "Baichuan" protocol only compatible with their apps and NVRs (any camera that uses "port 9000" will likely be using this protocol). Neolink allows you to use NVR software such as Shinobi or Blue Iris to receive video from these cameras instead. The Reolink NVR is not required, and the cameras are unmodified. Your NVR software connects to Neolink, which forwards the video stream from the camera.

The Neolink project is not affiliated with Reolink in any way; everything it does has been reverse engineered.

Supported cameras

Neolink intends to support all Reolink cameras that do not provide native RTSP. Currently it has been tested on the following cameras:

- B800/D800
- B400/D400
- E1
- Lumus
- 510A

Neolink does not support other cameras such as the RLC-420, since they already provide native RTSP.

About

An RTSP bridge to Reolink IP cameras

www.thirtythreeforty.net/posts/2020/05…

#rtsp #rtsp-server #reolink

Readme
AGPL-3.0 license
Activity
885 stars
52 watching
146 forks
Report repository

Releases 1

0.4.0 Pinned Latest
on Apr 11

Packages

No packages published

Contributors 15

Languages

Rust 91.9% Lua 7.7%
Other 0.4%

Usage

1. First, write a configuration yaml file describing your cameras. See the Configuration section below or the provided sample config.
2. Launch Neolink:

```
neolink rtsp --config=your_config.yaml
```

3. Then, connect your RTSP viewer to `rtsp://127.0.0.1:8554/your_camera_name`!

Additional commands

Neolink also has some additional command line tools for controlling the camera. They are all used through neolink subcommands like this:

```
neolink subcommand --config=...
```

The currently supported subcommands are

- **rtsp**: The standard neolink rtsp bridge
- **status-light**: Control the LED status light
- **reboot**: Reboot a camera
- **talk**: Enable talk back through either the microphone or by reading a sound file.

For a full list of commands use `neolink help`, or use `neolink help <subcommand>` for details on a subcommand of interest.

Download & Installation

Builds are provided for the following platforms:

- Windows x86_64 (download)
- macOS x86_64 (download)
- Ubuntu/Debian x86_64 (download)
- Ubuntu/Debian x86 (download)
- Debian aarch64 (Raspberry Pi 64-bit) (download)
- Debian armhf (Raspberry Pi 32-bit) (download)
- Arch Linux (AUR)
- Docker x86 (see below)

Windows/Linux

1. Install Gstreamer from the most recent MSI installer on Windows, or your package manager on Linux.

   On Ubuntu/Debian machines gstreamer can be installed with:

   ```
   sudo apt install \
     libgstrtspserver-1.0-0 \
     libgstreamer1.0-0 \
     libgstreamer-plugins-bad1.0-0 \
     gstreamer1.0-plugins-good \
     gstreamer1.0-plugins-bad
   ```

2. If you are using Windows, add the following to your `PATH` environment variable:

   ```
   %GSTREAMER_1_0_ROOT_X86_64%\bin
   ```

   Note: If you use Chocolatey to install Gstreamer, it does this automatically.

3. Download and unpack Neolink from the links above.

   i. Note: you can also click on this link to see all historical builds. You will need to be logged in to GitHub to download directly from the builds page (wget doesn't work)

   Ubuntu/Debian/Raspberry Pi OS example:

   ```
   unzip release-arm64-buster.zip
   sudo cp neolink /usr/local/bin/
   sudo chmod +x /usr/local/bin/neolink
   ```

4. Write a configuration file for your cameras. See the section below.

5. Launch Neolink from a shell, passing your configuration file:

   ```
   neolink rtsp --config my_config.toml
   ```

6. Connect your NVR software to Neolink's RTSP server.

The default URL is `rtsp://127.0.0.1:8554/your_camera_name` if you're running it on the same computer. If you run it on a different server, you may need to open firewall ports. See the "Viewing" section below for more troubleshooting.

Docker

A Docker image is also available containing Neolink and all its dependencies. The image is `thirtythreeforty/neolink`. Port 8554 is exposed, which is the default listen port. You must mount a configuration file (see below) into the container at `/etc/neolink.toml`.

Here is a sample launch commmand:

```
docker run \
  -p 8554:8554 \
  --restart=on-failure \
  --volume=$PWD/config.toml:/etc/neolink.toml \
  thirtythreeforty/neolink
```

Here is an example docker-compose:

```
---
version: "2"
services:
  neolink:
    image: thirtythreeforty/neolink
    container_name: neolink
    ports:
      - 8554:8554
    volumes:
      - $PWD/neolink.toml:/etc/neolink.toml
    restart: unless-stopped
```

The Docker image is "best effort" and intended for advanced users; questions about running Docker are outside the scope of Neolink.

If you use a battery-powered camera (or other UDP-only camera) you will need to either use `--net=host` or setup a macvlan for the docker image that supports UDP broadcast. This is because UDP requires that udp broadcast messages are transmitted across the docker network interface, however this is not possible in the default bridging mode

Configuration

Note: for a more comprehensive setup tutorial, refer to the Blue Iris setup walkthrough in `docs/` (which is probably also helpful even with other NVR software).

Note: more comprehensive setup details for linux based devices is provided in docs/unix_setup.md

Note: instructions for also setting up a (systemd based) service for linux based devices is provided in docs/unix_service.md

Copy and modify the `sample_config.toml` to specify the address, username, and password for each camera (if there is no password, you can omit that line). The default credentials for some cameras is username `admin` password `123456`.

- For a non battery powered camera you need to provide the address field with the ip and port (default 9000).
- For a battery powered camera you need to provide the uid field with the camera's UID. In this case your network must support UDP. Battery cameras exclusively use this UDP mode so you must always use a UID.

Each `[[cameras]]` block creates a new camera; the `name` determines the RTSP path you should connect your client to.

By default, the HD stream is available at the RTSP path `/name` or `/name/mainStream`, and the SD stream is available at `/name/subStream`. You can use only the HD stream by adding `stream = "mainStream"` to the `[[cameras]]` config, or only the SD stream with `stream = "subStream"`.

Note: The B400/D400 models only support a single stream at a time, so you must add this line to sections for those cameras.

By default Neolink serves on all IP addresses on port 8554. You can modify this by changing the `bind` and the `bind_port` parameter. You only need one `bind` / `bind_port` setting at the top of the config file.

You can enable `rtsps` (TLS) by adding a `certificate = "/path/to/pem"` to the top section of the config file. This PEM should contain the certificate and the key used for the server. If TLS is enabled all connections must use `rtsps`. You can also use client side TLS with the config option `tls_client_auth = "none|request|require"`; in this case the client should present a certificate signed by the server's CA.

TLS is disabled by default.

You can password-protect the Neolink server by adding `[[users]]` sections to the configuration file, but this is not secure without also using TLS:

```
[[users]]
name: someone
```

```
pass: somepass
```

you also need to add the allowed users into each camera by adding the following to `[[cameras]]`.

```
permitted_users = ["someone", "someoneelse"]
```

Anywhere a username is accepted it can take any username or one of the following special values.

- `anyone` means any user with a valid user/pass
- `anonymous` means no user/pass required

The default `permitted_users` list is:

- `[ "anonymous"]` if no `[[users]]` were given in the config meaning no authentication required to connect.
- `[ "anyone" ]` if `[[users]]` were provided meaning any authourised users can connect.

You can change the Neolink log level by setting the `RUST_LOG` environment variable (not in the configuration file) to one of `error`, `warn`, `info`, `debug`, or `trace`:

- On sh:

```
set RUST_LOG=debug
```

- On Bash:

```
export RUST_LOG=debug
```

Viewing

Connect your RTSP client to the stream with the name you provided in the configuration file.

Again, the default URL is `rtsp://127.0.0.1:8554/your_camera_name` if you're running it on the same computer as the client. The smaller SD video is `rtsp://127.0.0.1:8554/your_camera_name/subStream`.

4K cameras send large video "key frames" once every few seconds and the client must have a receive buffer large enough to store the entire frame. If your client's buffer size is configurable (like Blue Iris), ensure it's set to 20MB, which should ensure plenty of headroom.

Stability

Neolink has had minimal testing, but it seems to be very reliable in multiple users' testing.

The formats of all configuration files and APIs is subject to change as required while it is pre-1.0.

Development

Neolink is written in Rust, and binds to Gstreamer to provide RTSP server functionality.

To compile, ensure you have the Rust compiler, Gstreamer, and gst-rtsp-server installed.

Then simply run:

```
cargo build
```

from this top directory.

Baichuan Protocol

The "port 9000" protocol used by Reolink and some Swann cameras is internally referred to as the Baichuan protocol; this is the company based in China that is known internationally as Reolink.

This protocol is a slightly convoluted header-data format, and appears to have been upgraded several times.

The modern variant uses obfuscated XML commands and sends ordinary H.265 or H.264 video streams encapsulated in a custom header.

More details of the on-the-wire protocol are provided in `dissector/`.

Baichuan dissector

A Wireshark dissector is available for the BC wire protocol in the `dissector` directory.

It dissects the BC header and also allows viewing the deobfuscated XML in command messages. (It cannot deobfuscate newer messages that use AES encryption.) To use it, copy or symlink it into your Wireshark plugin directory; typically this is `~/.local/lib/wireshark/plugins/` under Linux.

Currently the dissector does not attempt to decode the Baichuan "extension" messages except `binaryData`. This will change in the future as reverse engineering needs require.