

# **SPR Guide v0.3.2**

# Table of contents:

- SPR : A Secure Programmable Router
  - Fundamentally Secure
  - The Network Design
    - Default deny
    - Zero-trust networking
    - Device isolation
    - Best of class WiFi features
    - Multicast Support
  - Design Ethos
    - Transparent and Modifiable
    - Containerized
    - Sound Software Selection
  - The Service Listing
  - API Extensions
- Raspberry Pi 4 Model B Hardware Guide
  - Hardware Requirements
    - Running SPR
    - Building from source
    - Using a different wireless card
  - Install the SPR Pi 4 Image
    - Running
    - Follow the SPR Setup Guide
- Setup Guide
  - Base System Setup
    - Download and install Ubuntu Server
    - Setup
    - Configuration
    - Run SPR
    - Run SPR dev release
    - Building the project
    - Additional Notes
    - Sign in to the Web UI
    - Add new devices
- Virtual Setup Guide
  - Quick Setup

- Host network configuration
- Manual configuration
  - Configuration
  - Start the SPR Services
  - Sign in to the Web UI
  - Setup and connect to VPN
- Overview
  - Updating SPR
  - Manage Devices
  - Mange DNS Rules
  - Link Setup
    - Uplink Configuration
    - Downlink Configuration (LAN)
- Firewall
  - Common rules
  - Advanced firewall configuration
- Logging and Events
- WiFi Settings
- Docker Tips & Tricks
  - Virtual mode
  - Updating from the command line
  - Listing running containers
  - Viewing logs
  - Entering a container
  - Release Channels
  - Starting plugins without the api/superd
  - Building super
    - Building with low system memory
- Making an API Extension
  - Get the sample code
  - Buiding & Running
  - API Calls
  - Configuring the plugin to auto-start
  - SPRBus notes
- Testing
  - Linux end to end tests
    - Setup

- Run the tests
  - UI Dev
    - Running the UI with the Mock API
    - Point the UI to your API
    - Run the tests
  - Manage Devices
    - Add a Device
      - Authentication
      - Groups
      - Advanced Settings
    - Edit Device
    - Copy or Duplicate A Device Entry
  - DNS Block Lists & Settings
    - Block lists
      - Manage Existing Blocklists:
      - Block Lists Tags
      - DNS Overrides
        - Permit Override
        - Block Override
      - Settings
      - Logs
  - Firewall
    - Firewall Use Cases
      - Advanced firewall configuration
    - Port Forwarding
      - Add a Port Forward
    - Inbound Traffic Block
      - Add a Inbound Block
    - Forwarding Traffic Block
      - Add a Forwarding Block
    - Custom Interface Access
    - Multicast Proxy
    - Endpoints
- Link Management
  - Uplink Configuration
  - Downlink Configuration (LAN)
  - Multiple Uplinks

- Set Uplink Interface IP
- Enable a WiFi Uplink
  - Enable Wired LAN Interface
  - Enable VLAN Trunk Port
- Installing Extensions
  - Community Plugins
  - Plugin Installation
  - Auto-starting a plugin
- Troubleshooting
- Updating SPR
  - How To Update?
    - Auto Updates
    - Release Channels
    - Updating from the command line
- WiFi Settings
  - SPR Compatibility
  - SSID
  - Channel Configuration
  - Enabling WiFi 6
  - Legacy WPA1 Support
  - Tuning Capabilities
  - Troubleshooting
- Overview
  - Site VPN
  - Mesh
  - Programmable Firewall (PFW)
  - Setup Guide
- WiFi Mesh Support
  - Install
- PFW Programmable Firewall
- Configuration
- Site VPN
  - Configuration
  - Routing Traffic
    - DNS Split Tunnel
    - Verifying connectivity
- Base

- Firewall Configuration
- Database
- DHCP
  - Tiny Subnets
- DHCP Client
  - IPv6
- DNS
  - The DNS Service runs off CoreDNS with custom plugins
  - DNS Configuration Files:
  - Local Mappings
  - DNS over HTTPS
- Dynamic DNS
- Extensions
- Multicast UDP Proxy
- Packet Logs
  - The Groups
    - Group 0
      - wan:in
      - lan:in
      - wan:out
      - lan:out
    - 
    - Group 1 - Dropped packets
      - drop:private
      - drop:forward
      - drop:input
      - drop:mac
  - Inspecting packets
    - Script to inspect traffic
- OUI & MAC Lookups
- PPP
- SPRbus
  - Usage
  - Available events
  - Example Events
  - Command line tools
- Development

- [Superd](#)
  - [About](#)
- [VPN](#)
  - [About](#)
- [Wifi Uplink](#)
- [Wifid](#)
- [API Overview](#)
  - [API Plugins](#)
    - [Query the http API](#)
    - [Query a plugin API](#)
  - [SPR Event Bus](#)
- [Authentication](#)
  - [Configuration](#)
  - [Basic Authentication](#)
  - [Bearer Tokens](#)
  - [WebAuthN](#)
- [SSL Support](#)
  - [Enable SSL web api](#)
- [Frequently Asked Questions](#)
  - [Troubleshooting](#)
- [Security](#)
  - [Contact Information](#)
- [Security Goals](#)
- [Key Security Features](#)
  - [Multi-PSK & VLANs](#)
    - [No MAC spoofing and other layer 2 network pivoting](#)
    - [Multicast Limitation](#)
  - [Upstream LAN Traffic Blocked By Default](#)
  - [WPA3 Support](#)
    - [Practical Limitations of WPA3](#)
      - [iOS Device QR-Code WPA2 Downgrade](#)
      - [Many devices don't support WPA3 yet, some still require WPA1 even](#)
    - [Network Visibility](#)
  - [Threat Actors](#)
    - [Remote Internet Attacker](#)
    - [Man In The Middle / Malicious ISP](#)
    - [Supply Chain Attacker](#)

- Physical Proximity Attacker (Evil Neighbor)
- Inside Perimeter Attacker (Evil Guest)
- Compromised Device Attacker (Implant)
- Threat Vectors
  - Network Flaws
    - Weak Passphrase / Password Reuse
    - ARP Spoofing
    - MAC Spoofing
    - DHCP MAC Spoofing
    - VLAN Hopping
    - Insecure Private Requests from Web Browsers
  - Software Implementation Flaws
    - Memory corruption
    - Command Injection
    - XSS, CSRF
    - DNS Cache Poisoning
    - Response Splitting Attacks
  - 802.11 Flaws
    - Cryptographic Vulnerabilities
    - Password Cracking
    - Frag Attacks
    - MITM
    - AP Isolation Bypass
    - Packet in Packet Attacks
- VPN Overview
  - VPN Client
    - Add a peer
  - Site-to-Site VPN
    - Setup site-to-site VPN
  - PFW support

# SPR : A Secure Programmable Router

**SPR** lets users run security-hardened networks without restrictive drawbacks by providing adaptive microsegmentation with flexibility.

## Fundamentally Secure

**SPR** is a new type of router, designed to be immune to network attacks that enable devices to breach laterally across networks which separates it from other networking technology. SPR addresses WiFi in particular, as it typically has weak security properties that SPR corrects through robust, defense-in-depth features. It also supports **VLAN Tagging** for wired devices and **Wireguard™ VPN** clients.

## The Network Design

### Default deny

The firewall drops forwarded and input packets by default. Devices are explicitly configured for communicating to the internet, accessing DNS, or intranet communications.

### Zero-trust networking

Device identities are established via unique wifi passphrases, VLAN tagging on wired devices, and VPN keys. This creates a high source of confidence about how traffic is routed.

### Device isolation

Each device on the network is isolated into its own private network, and devices send traffic through the router to communicate.

- ▼ How does this perform?

There is a negligible latency tradeoff when devices do not directly communicate on local networks. The router arbitrates communication between devices for better control. The verdict maps, similar to ipsets, provide scalable rules without an exponential number of firewall rules that make traditional firewall rules prohibitively slow.

## Best of class WiFi features

Each device is given a unique WiFi passphrase to block traffic interception, injection, and rogue AP attacks that are otherwise possible with WiFi. Devices have unique group encryption keys and each device connects with an individual VLAN and virtual interface. SPR is also the very first project to support multiple passphrases with WPA3 on the same SSID.

### ▼ What about Enterprise WiFi Security?

Among authentication suites, EAP-TLS is a recommended best practice and creates valid security properties. However, it requires installing certificates on devices which is not possible with many IOT or locked down devices, limiting the ability to harden their access. Rather than create MAC bypass holes, SPR uses the most commonly supported schemes that are universal to all wifi capable systems. Another common authentication mechanism is EAP-PEAP however this suffers from long standing MITM issues as [PEAPv2](#) with channel binding to stop attacks was never formally adopted or fully supported by the industry; the PEAP relay attack applies after secure authentication has happened.

## Multicast Support

Zeroconf and mDNS are both prevalent and a requisite for home networking. However, these require being on the same subnet to work or a router that supports IGMP forwarding or a proxy. SPR has a userland proxy to transparently relay multicast traffic between devices and across network interfaces. The firewall supports configuration for how to use multicast

# Design Ethos

## Transparent and Modifiable

SPR is built as an [open source](#) project. It is built to be developer friendly and allow iterative development without reflashing or reinstalling.

## Containerized

Containers make it simple to build the code from scratch or run prebuilt images. The containers split network services up by functionality and are orchestrated with APIs. SPR is built to run as a WiFi router, but it can also be launched as a standalone VPN & DNS Service within a virtualized network namespace.

## Sound Software Selection

SPR is built mostly with golang and a react frontend. Where possible, memory-safe languages with good security track records are preferred over native code such as C or code that is difficult to write securely, like PHP.

# The Service Listing

The following services are currently included with SPR:

- [api](#) ⇒ API for frontends and CLIs to manage connected devices, network groups, and firewall rules
- [base](#) ⇒ Networking initialization and configuration at startup
- [db](#) ⇒ Database key/value store
- [dhcp](#) ⇒ DHCP Server
- [dhcp\\_client](#) ⇒ DHCP Client for requesting IP addresses from uplink
- [dns](#) ⇒ DNS Service, running CoreDNS
- [multicast\\_udp\\_proxy](#) ⇒ A multicast proxy to support multicast networking for usability
- [packet\\_logs](#) ⇒ NFLog packet log emitter
- [ppp](#) ⇒ A Point-to-Point protocol service for authenticating and requesting IP addresses from uplink
- [superd](#) ⇒ Daemon to manage docker comms and restarting SPR
- [wifid](#) ⇒ Wifi Base Station Service, running hostapd

# API Extensions

- [DNS Block & Log](#) ⇒ Fine grained DNS Rules & Logs
- [Wireguard](#) ⇒ Configuration service for running a Wireguard VPN

- [Lookup](#) ⇒ OUI & ASN Lookup Service
- [Dyndns](#) ⇒ Dyndns Service

# Raspberry Pi 4 Model B Hardware Guide



## Hardware Requirements

The current setup assumes you'll be using a Raspberry Pi Model 4b with an mt76 based wireless adapter. The built-in ethernet port of the Raspberry Pi is connected to upstream/WAN/internet.

## Running SPR

- A WiFi Dongle for better performance and WPA3 support. The following WiFi6 dongles work great: [Netgear's A8000 6-E](#), and the bulkier [ALFA AWUS036AXML 6-E](#). For Wifi5: [Netgear's A620](#) and

Alfa's AWUS036ACM.

### ❗ WIFI DONGLES

The WiFi 5 dongles have been tested to perform well with a *500-550Mbps* top speed in a low noise environment and the WiFi 6 dongles can reach closer to *700mpbs* in a low noise environment.

- A [USB SSD](#) works best. A 16GB SD card can also be used but is not recommended
- Ethernet cable for setup (and if you run uplink over the wire)
- *Optional* For running multiple WiFi adapters, a powered USB 3.0 capable hub is recommended
- *Optional* An Ethernet USB Dongle for additional LAN devices since the built-in ethernet card (eth0) will be used for the WAN. The [UE 300](#) has been tested to run near line speed on the Pi (*950Mbps*). The [U3 330](#) also works and provides additional USB 3.0 ports.

### 💡 TIP

Grab the Complete Starter Kit for everything you need to get started [here](#)

## Building from source

- 8 GB of RAM are recommended for building SPR on the Pi. Any amount of RAM can work for building but ramdisks have to be disabled in the docker build. The prebuilt GHCR has prebuilt SPR images available also.

## Using a different wireless card

For using a different wireless adapter, the default hostapd configuration may need to be modified in the UI or in `configs/wifi/`. Note that the built-in Raspberry Pi Wireless card can not be used as an AP by default as it does not support AP/VLAN, which is a feature that SPR relies on.

## Install the SPR Pi 4 Image

For running SPR on a Raspberry Pi 4, a pre-built image is available on the [releases page](#).

You can download `spr.img.xz` and flash it with a tool such as [Balena Etcher](#), or if you prefer to write the image using the command line:

```
# On a mac, write the image to the external disk (rdiskX).  
# Be careful to make sure you identify your removable drive correctly.  
$ xzcat spr.img.xz | dd of=/dev/rdiskX bs=$[1024*1024]
```



### TIP

If you use Balena Etcher be sure to download the image first and choose *Flash from file*.

## Running

1. Plug in SPR to your local network using a ethernet cable
2. Boot the drive on the Pi - setup will complete on device and reboot
3. Visit <http://spr.local> to setup your device

Default credentials for ssh are: `ubuntu:ubuntu`

### UPSTREAM SERVICES

Under `/home/spr/super/config/base/config.sh`, consider disabling `UPSTREAM_SERVICES_ENABLE=1` if you do not want the API and SSH exposed to the WAN ethernet interface. This can also be toggled under the Firewall Services tab in the UI.

## Follow the SPR Setup Guide

If you prefer to run on an existing image, or run from a fresh Ubuntu server install, you can [Follow the SPR setup guide](#)

# Setup Guide

SPR is built to run on systems capable of running Ubuntu and Docker. For also developing on-device, systems should have 32-64GB of writable storage and 8GB of RAM. With less RAM, building from ram disks should be disabled (see the README).

## Base System Setup

**NOTE** This guide shows how to setup and run SPR using Docker. See the [Raspberry Pi 4 Setup Guide](#) for running SPR using a prebuilt image.

### Download and install Ubuntu Server

*Skip this step if you already have a base system setup.*

For Running on a Raspberry Pi you can download the 23.04 Ubuntu image [here](#). For other hardware, [download](#) an Ubuntu server installer for your architecutre (arm64 or amd64), then boot the device into the installer.

SPR has been tested mainly on ubuntu release however many Linux distributions should work.

#### Write image to SD-card or USB

You can flash the image with a tool such as [Balena Etcher @ https://www.balena.io/etcher/](https://www.balena.io/etcher/)

If you prefer to write the image using the command line:

```
# On a mac, write the image to the external disk (rdiskX).
# Be careful to make sure you identify your removable drive correctly.
$ xzcat ubuntu-22.0-4-preinstalled-server-arm64+raspi.img.xz | dd
of=/dev/rdiskX bs=$[1024*1024]
```

## Setup

Boot the system and login, the default credentials are `ubuntu/ubuntu`

## Fetch the SPR repository

```
# On the booted system  
git clone https://github.com/spr-networks/super  
cd super
```

## Prepare the Ubuntu Install for SPR [base/setup.sh](#)

```
#if using a flash drive  
sudo base/setup.sh  
#or if using an sdcard the following script reduces disk usage  
# base/setup-sdcard.sh #this setup will reduce writes from log files  
reboot
```

## Configuration

Inside the super directory, copy `base/template_configs/` to `configs/`

```
cd super  
cp -R base/template_configs configs  
.configs/scripts/gen_coredhcp_yaml.sh > configs/dhcp/coredhcp.yml  
.configs/scripts/gen_watchdog.sh > configs/watchdog/watchdog.conf
```

Verify the main config `configs/base/config.sh` is correct. Ensure that `WANIF` matches the name of the outbound interface if not using the builtin ethernet port.

Also, consider disabling `UPSTREAM_SERVICES_ENABLE=1` if you do not want the API and SSH exposed to the WAN port.

## Run SPR

Start the services (this script runs `docker-compose up -d`):

```
./run_docker_compose.sh
```

Testing containers are built and published to GitHub's Container Registry, and can be run without building on the device. If you would instead like to build everything from source, proceed to the [next section](#).

To see logs:

```
docker-compose logs -f
```

## Run SPR dev release

If you want to get the latest features and updates you can switch to the dev release of SPR by setting the `RELEASE_VERSION` and `RELEASE_CHANNEL` environment variables:

```
export RELEASE_VERSION="latest" RELEASE_CHANNEL="-dev"  
./run_docker_compose.sh
```

Setting the `RELEASE_VERSION` variable can also be used to run specific versions:

```
export RELEASE_VERSION="v0.1.28"  
./run_docker_compose.sh
```

## Building the project

SPR can be built on device, however there is a small gotcha.

Because SPR uses `nftables` exclusively, and docker is made to work with `iptables` which the SPR install disables for docker, a fixup script is needed.

The script installs minimal docker forwarding rules. Once SPR is running this is no longer needed. However, if SPR fails to run and you need to build with docker, you may need to run this script again.

```
#fix forwarding rules for Docker  
sudo ./base/docker_nftables_setup.sh
```

And build.

```
sudo ./build_docker_compose.sh
```

If something went wrong with with a cached stage, it is possible to specify `--no-cache`. For example:

```
sudo ./build_docker_compose.sh --no-cache
```

A specific service can also be built by passing it as an argument (`./build_docker_compose.sh api`)

To run the local build:

```
./run_docker_compose.sh
```

## Additional Notes

Check dns-Corefile to tweak DNS server configuration as well as the hostapd settings in `configs/dns/`

It is possible to use SPR's DNS server for the host system's DNS requests too, by updating the resolv.conf file.

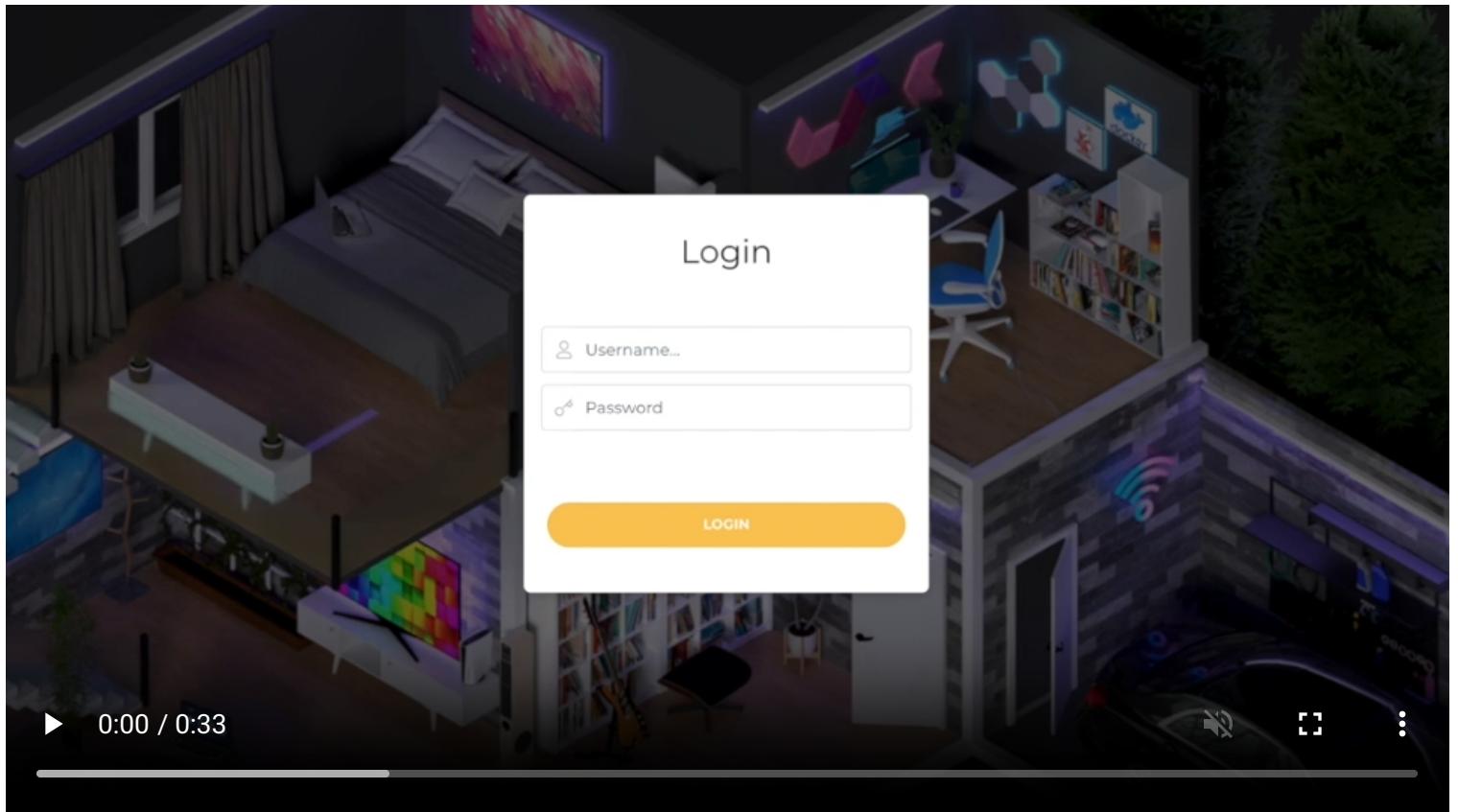
```
sudo echo -e "nameserver 127.0.0.1" > /etc/resolv.conf
```

## Sign in to the Web UI

Connect to the IP of the device from the wired WAN interface in a browser, SPR is using mDNS and you should be able to reach it @ <http://spr.local>.

A setup screen should show up asking you to set up interfaces and assign a username and password.

## Add new devices



# Virtual Setup Guide

SPR can run its services in a network namespace and expose Wireguard or DNS as services. This is helpful for running SPR in the cloud as a personal VPN, for example.

## Quick Setup

```
sudo bash -c "$(curl -fsSL https://raw.githubusercontent.com/spr-networks/super/master/virtual_install.sh)"
```

This will download the [SPR repository](#) and run *virtual\_install.sh*. The script will install dependencies, start all the containers and add one peer that can connect to the VPN. See the next section for a walkthrough of the script.

If you want to add another device, just run the setup script again:

```
cd super
sudo ./virtual_install.sh
```

Next [setup](#) a client to connect to SPR over VPN.

## Host network configuration

Your source ip address need to be able to access the host running spr on port 51280/udp. For setup in the cloud, see the blog specific vendor guides for network configuration:

- [Google Cloud](#)
- [Amazon AWS](#)
- [Digital Ocean](#)

## Manual configuration

[Clone the super repository](#)

```
git clone https://github.com/spr-networks/super.git
```

## Install dependencies

```
apt install -y curl git docker-compose docker.io jq qrencode iproute2  
wireguard-tools
```

## Configuration

### 1. Populate the configuration directory

```
cd super  
cp -R base/template_configs/ configs/
```

### 2. Enable virtual config in `configs/base/config.sh`

```
cp configs/base/virtual-config.sh configs/base/config.sh
```

Example configuration:

```
#!/bin/sh  
VIRTUAL_SPR=1  
UPSTREAM_SERVICES_ENABLE=1  
WANIF=eth0  
RUN_WAN_DHCP=true  
RUN_WAN_DHCP_IPV=4  
LANIP=192.168.2.1  
DNSIP=$LANIP  
TINYNETSTART=192.168.2.4  
TINYNETSTOP=192.168.2.255  
TINYNETMASK=255.255.255.252  
TINYSLASHMASK=30  
WIREGUARD_PORT=51280
```

### 3. Set an api username and password for the web UI

```
echo "{\"admin\" : \"your password goes here\"}" >
configs/auth/auth_users.json
```

## Start the SPR Services

```
docker-compose -f docker-compose-virt.yml up -d
```

Note: *On MacOS you may need to modify docker-compose-virt.yml to disable the `journal` driver and enable the `json-file` driver instead.*

**Optional** Copy the virtual config for easier control of the docker containers:

```
cp docker-compose-virt.yml docker-compose.yml
```

Now you can run `docker-compose restart` in the super directory to restart all the containers.

## Sign in to the Web UI

The API is setup to listen on localhost:8000 on the host. To access the service from your local machine, forward port 8000 using ssh:

```
ssh $HOST -N -L 8000:127.0.0.1:8000
```

Navigate to <http://localhost:8000>

Sign in with the username and password assigned in step 3 of the configuration.

## Setup and connect to VPN

In the SPR web UI, navigate to VPN and click **Add peer**, enable the *lan* group if you want to be able to connect to the admin interface over VPN.

Click **Save** and download the client config or scan the QR Code on your phone with the official [WireGuard \(© Jason A. Donenfeld\)](#) client.

Check [this guide](#) on how to setup the VPN client for your device.

**Note** SPR exposes port 51280 over UDP (see `$WIREGUARD_PORT` in config.sh) on the host. If a firewall is present you will need to allow access to this port. Check the documentation for your hosting provider on how to do this.

# Overview

Need a guide that's missing? [File a ticket](#)

## Updating SPR

- [How to update](#)

## Manage Devices

- [Add a Device](#)
- [Edit Devices](#)
- [Copy a device](#)

## Manage DNS Rules

- [DNS Block Lists](#) Block Ads, Tracking & More
- [DNS & Tags](#) Make block lists select devices by Tag
- [DNS Overrides](#) Make custom DNS Rules
- [DNS Settings](#) Set DNS Privacy settings, DNS Provider
- [DNS Logs](#) Inspect DNS Queries

## Link Setup

### Uplink Configuration

- [Set DHCP settings or assign a Static IP](#)
- [Enable a Wireless Uplink](#)

### Downlink Configuration (LAN)

- [Enable Wired LAN Interface](#)

- [Enable VLAN Trunk Port](#)

# Firewall

## Common rules

- [Port Forwarding](#) is if you want to open a port *from WAN* and forward the traffic to a local device
- [Inbound Traffic Block](#) block access from a specific address and/or destination
- [Forwarding Traffic Block](#) allow/block devices from your local network

## Advanced firewall configuration

- [Custom Interface Access](#) - Join a docker bridge or custom interface to SPR networking
- [Multicast Proxy](#) -- Configure mDNS, SSDP, and other UDP broadcast services
- [Endpoints](#) - Define one-way connectivity rules

# Logging and Events

- Events view
- Enable notifications
- Using the cli tools

# WiFi Settings

- [wifi-spr-compability](#)
- [Updating SSID Name](#)
- [Channel configuration](#)

# Docker Tips & Tricks

SPR runs inside of containers. It can run a network inside of a network namespace, or with the host network namespace. Here are some useful things to know

## Virtual mode

Super ships with 3 docker compose files

- `docker-compose.yml` the default for running SPR on the host network
- `docker-compose-virt.yml` for running SPR in virtual bridge, for example for a cloud VPN
- `docker-compose-test.yml` for running the end to end tests

If using SPR in virtual mode, make sure to use the virtual compose file.

```
docker compose -f docker-compose-virt.yml up -d
```

## Updating from the command line

```
sudo -s  
cd /home/spr/super/  
docker compose pull  
docker compose up -d
```

## Listing running containers

```
docker ps
```

```
root@haxlab0:/home/spr/super# docker ps  
CONTAINER ID IMAGE COMMAND  
CREATED STATUS PORTS NAMES  
ad9f3a4aa353 ghcr.io/spr-networks/super_wifid:latest  
"/bin/sh -c /scripts..." 17 seconds ago Up Less than a second  
superwifid  
e509fed11404 ghcr.io/spr-networks/super_wireguard:latest  
"/scripts/startup.sh" 28 seconds ago Up 1 second
```

```
superwireguard
5b0cec25f671    ghcr.io/spr-networks/super_multicast_udp_proxy:latest
"/scripts/startup.sh"    28 seconds ago    Up 2 seconds
super_multicast_udp_proxy
746c2599eb42    ghcr.io/spr-networks/super_db:latest
"/scripts/startup.sh"    28 seconds ago    Up 2 seconds
superdb
63c30c8dbc83    ghcr.io/spr-networks/super_plugin-lookup:latest
"/scripts/startup.sh"    28 seconds ago    Up 2 seconds
superplugin-lookup
8c54ec9f1769    ghcr.io/spr-networks/super_api:latest
"/scripts/startup.sh"    39 seconds ago    Up 4 seconds
superapi
a1679ce36ca4    ghcr.io/spr-networks/super_dhcp:latest
"/scripts/startup.sh"    39 seconds ago    Up 4 seconds
superdhcp
a69a08569bd8    ghcr.io/spr-networks/super_dhcp_client:latest
"/scripts/client.sh"    39 seconds ago    Up 4 seconds
superdhcp_client
4ca730f9cf84    ghcr.io/spr-networks/super_dns:latest
"/bin/sh -c /scripts..."    39 seconds ago    Up 4 seconds
superdns
d5ff0bd8d40d    ghcr.io/spr-networks/super_base:latest
"/bin/sh -c /scripts..."    50 seconds ago    Up 5 seconds
superbase
fc8006db1370    ghcr.io/spr-networks/super_superd:latest
"/scripts/startup.sh"    50 seconds ago    Up 5 seconds
superd
e58a351190b1    ghcr.io/spr-networks/super_packet_logs:latest
"/scripts/startup.sh"    50 seconds ago    Up 5 seconds
superpacket_logs
8a94e9476931    spr-nexmon-nexmon
"/scripts/startup.sh"    4 weeks ago        Up 3 days
supernexmon
```

## Viewing logs

```
docker compose logs -f wifid
```

```
root@haxlab0:/home/spr/super# docker compose logs -f wifid
superwifid | % Total    % Received % Xferd  Average Speed   Time     Time
```

```
Time Current  
superwifid | Dload Upload Total Spent  
Left Speed  
100 6 100 6 0 0 65 0 --:--:-- --:--:-- --:--:-- 65  
superwifid | wlan1: interface state UNINITIALIZED->COUNTRY_UPDATE  
superwifid | wlan1: interface state COUNTRY_UPDATE->HT_SCAN
```

## Entering a container

```
docker exec -it superapi bash
```

```
root@haxlab0:/# ip -br addr  
lo UNKNOWN 127.0.0.1/8 ::1/128  
eth0 UP 192.168.2.218/30 fe80::e65f:1ff:fefd:a175/64  
wlan1 UP fe80::9618:65ff:fe55:54ae/64  
tailscale0 UNKNOWN fe80::31cf:a08b:d7e0:47e4/64  
docker0 DOWN 172.17.0.1/16 fe80::42:1ff:fea3:7057/64  
tailscale DOWN 172.18.0.1/16  
mon0 DOWN  
wlan0 UNKNOWN  
sprloop UNKNOWN 192.168.200.1/32 fe80::6416:d1ff:fe70:bf2a/64  
wg0 UNKNOWN  
root@haxlab0:/#
```

## Release Channels

Want to test the dev branch? Don't want to build it? You can use the `-dev` RELEASE\_CHANNEL

```
export RELEASE_CHANNEL=-dev  
docker compose pull
```

## Starting plugins without the `api/superd`

The `SUPDERDIR` path is required to let docker compose know where the state and config directories are.

```
export SUPERDIR=/home/spr/super/  
docker compose -f plugins/plugin/docker-compose.yml up -d
```

## Building super

For convenience, we provide a script, `build_docker_compose.sh` that will build the containers.

### Building with low system memory

To speed up the build a ramdisk is used from docker's buildkit. If that's not possible, consider disabling that capability

```
./build_docker_compose.sh --set "* .args.USE_TMPFS=false"
```

# Making an API Extension

## Get the sample code

Get the [sample code](#)

Read the [api docs](#) for information about how plugins work

## Buiding & Running

Typically, plugins are placed into the `plugins` directory. These will be included in backups.

Copy the API sample

```
cp -R ..../api_sample_plugin my_plugin
cd my_plugin
docker compose build
export SUPERDIR=/home/spr/super/ #path where super is
docker compose up
```

## API Calls

The plugin can export API extensions over a unix socket to the API. Configuration can be set by updating `configs/base/api.json` and adding a new entry to the `Plugins` list

```
{
  "Name": "my_plugin",
  "URI": "my_plugin",
  "UnixPath": "/state/plugins/my_plugin/socket",
  "Enabled": true,
  "Plus": false,
  "GitURL": "",
  "ComposeFilePath": ""
},
```

or by updating the UI

The screenshot shows the SPR (Software Defined Router) web interface. On the left, there's a sidebar with various sections like PFW, Supernetworks, DNS, Blocklists/Ad-Block, DNS Overrides, DNS Log, DNS Log Settings, Dynamic DNS, DNS Settings, TRAFFIC, Bandwidth Summary, Bandwidth Timeseries, Signal Strength, Traffic, EVENTS, Events, Logs, Notifications, SYSTEM, System Info, Plugins, Auth, DHCP Table, and ARP Table. The Plugins section is expanded, showing dns-block-extension (0.3.2), dns-log-extension (0.3.2), plugin-lookup (0.3.2), wireguard (0.3.2), dyndns (NONE), db (0.3.2), PPP (NONE), WIFILINK (NONE), nexmon (LATEST), and PLUS Plugins. A modal window titled "Add a new Plugin" is open in the center. It has fields for "Name" (with placeholder "Use a unique name to identify your plugin") and "URI" (with placeholder "Plugin will be @ \"http://spr/plugins/URI\""). Below these are "UNIX Path" (placeholder "Plugin pathname for unix socket") and "ComposeFilePath" (placeholder "Plugin pathname for unix socket"). At the bottom is a large blue "Save" button.

# Configuring the plugin to auto-start

In an upcoming release, this mechanism will be streamlined for advanced users.

For now, SPR does not allow arbitrary containers to be configured for auto-start from the UI alone.

Update the `configs/base/custom_compose_paths.json` to add the plugin. It is expected to be relative from the `super/` directory, for example `plugins/my_plugin/docker-compose.yml`

# SPRBus notes

SPRBus is our event bus where the API can send events. The sample includes commented code for how to use it.

# Testing

## Linux end to end tests

SPR has end to end tests which can be executed on a linux machine with system privilege. The tests will spin up mac80211\_hwsim based wireless networks and exercise the API.

See [tests/README.md](#)

### Setup

```
apt install -y linux-image-extra-virtual net-tools iw  
  
cd tests/runner/  
./build.sh  
cd ../../
```

### Run the tests

```
./tests/tests_start.sh  
./tests/run-tests.sh  
./tests/tests_stop.sh
```

### Expected output

```
~ running tests @ http://localhost:8000  
~ ID= 8c60362f977d  
~ waiting for runner to be done ...  
+ test run passed
```

# UI Dev

In the super directory, go to the `frontend` directory.

## Running the UI with the Mock API

```
super/frontend $ REACT_APP_API=mock yarn start
```

The UI will start on port `localhost:3000`. React scripts will auto open a browser. The UI will update in real time as code changes are made.

## Point the UI to your API

```
super/frontend $ REACT_APP_API=http://192.168.2.1 yarn start
```

## Run the tests

```
super/frontend $ yarn test
```

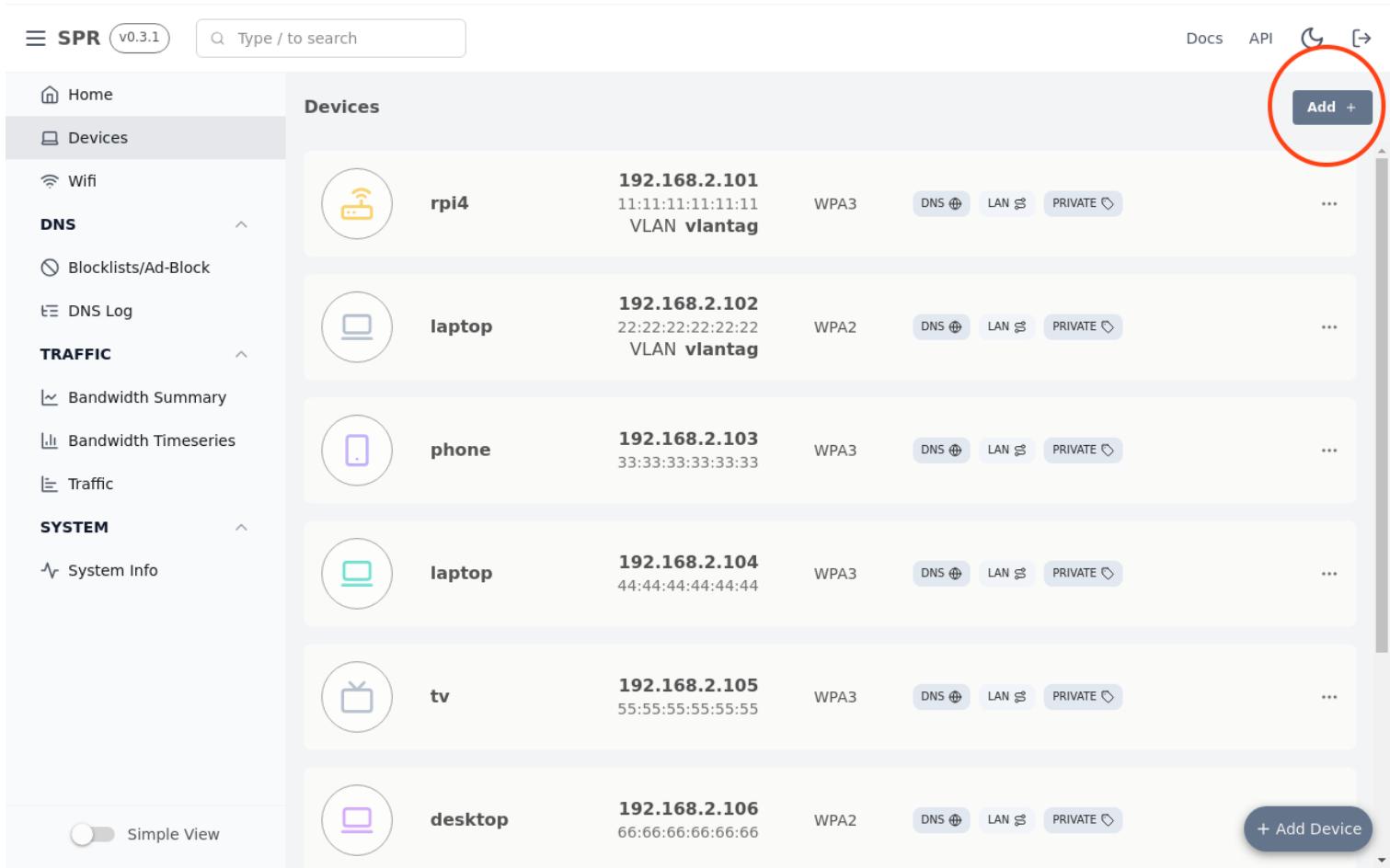
# Manage Devices

[Add a Device](#) Learn how to add new devices to your network. Since there is no global wifi password, a new entry is made for each new device.

[Edit Devices](#) Set tags, edit styles & update device info

[Copy a device](#) Lets you share wireless password between devices

## Add a Device



The screenshot shows the SPR web interface with the following details:

- Header:** SPR v0.3.1, search bar, Docs, API, and navigation icons.
- Left Sidebar:** Home, Devices (selected), WiFi, DNS, Blocklists/Ad-Block, DNS Log, Traffic (Bandwidth Summary, Bandwidth Timeseries, Traffic), System (System Info).
- Devices List:** A table listing six devices:
  - rpi4 (Icon: WiFi Dongle) - IP: 192.168.2.101, MAC: 11:11:11:11:11:11, VLAN: v1antag, WiFi: WPA3, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
  - laptop (Icon: Laptop) - IP: 192.168.2.102, MAC: 22:22:22:22:22:22, VLAN: v1antag, WiFi: WPA2, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
  - phone (Icon: Phone) - IP: 192.168.2.103, MAC: 33:33:33:33:33:33, WiFi: WPA3, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
  - laptop (Icon: Laptop) - IP: 192.168.2.104, MAC: 44:44:44:44:44:44, WiFi: WPA3, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
  - tv (Icon: TV) - IP: 192.168.2.105, MAC: 55:55:55:55:55:55, WiFi: WPA3, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
  - desktop (Icon: Desktop) - IP: 192.168.2.106, MAC: 66:66:66:66:66:66, WiFi: WPA2, DNS: Enabled, LAN: Enabled, PRIVATE: Enabled.
- Bottom Right:** A blue button labeled "+ Add Device".

Navigate to *Devices* in the menu & click *Add Device*.

▼ ⚠ iCloud Keychain Sync and WiFi Passwords

You need to set the same password on each of the iOS devices that are in the same keychain. Without this, they would sync the wrong password to each other and lock each other out of the network. You can copy an existing iOS device instead of adding a new one, to keep the same password.

The screenshot shows the SPR web interface with a sidebar on the left containing navigation links: Home, Devices, WiFi, DNS (selected), Blocklists/Ad-Block, DNS Log, TRAFFIC, Bandwidth Summary, Bandwidth Timeseries, Traffic, SYSTEM, and System Info. A 'Simple View' toggle is at the bottom of the sidebar. The main content area is titled 'Add a new WiFi Device'. It has two input fields: 'Device Name\*' (with placeholder 'A unique name for the device') and 'Passphrase' (with placeholder 'Optional. If empty a random password will be generated'). Below these are sections for 'Authentication' (radio buttons for WPA3, WPA2, and Wired, with WPA3 selected) and 'Groups' (checkboxes for wan, dns, and lan). A large blue 'Save' button is at the bottom. At the top right are links for Docs, API, and a refresh icon.

You are required to enter a *Device Name*. Pick a name for your device & enter a password (or leave empty for random), click **Save**.

## ▼ Authentication and Groups

### Authentication

WPA3 works for newer devices and is recommended. Choose *WPA2* if *WPA3* does not work or *Wired* if you are connecting the device over ethernet.

### Groups

- **wan** internet access
- **dns** for domain lookups, block/filter dns
- **lan** access to all other local devices

**wan** and **dns** is enabled by default. Add the device to **lan** sparingly, as this grants it access to all other devices. You can create groups and rules later for defining more specific access.

Connect to the Wifi with your device, either by scanning the QR code or with the password.

The screenshot shows the SPR web interface. On the left is a sidebar with navigation links: Home, Devices, Wifi, DNS (selected), Blocklists/Ad-Block, DNS Log, TRAFFIC (Bandwidth Summary, Bandwidth Timeseries, Traffic), and SYSTEM (System Info). At the bottom of the sidebar is a 'Simple View' toggle switch. The main content area displays a success message: "SSID TestAP" and "Password password". Below this is a large QR code. At the bottom is a "Back" button. The top right corner has links for Docs, API, and navigation icons.

A notification will popup when the device is connected & the device is available in the device view.

## Advanced Settings

Toggle "*Advanced View*" in the menu if you want to add Device expiry & other features.

[Home](#)  
[Devices](#)  
[Wifi](#)  
**NETWORK**  
[Uplink](#)  
[LAN](#)  
[Containers](#)  
[MESH](#)  
[VPN](#)  
**FIREWALL**  
[Firewall](#)  
[Services](#)  
[PFW](#)  
[Supernetworks](#)  
**DNS**  
[Blocklists/Ad-Block](#)  
[DNS Overrides](#)  
  
 Advanced View

### Add a new WiFi Device

Wired devices are added automatically & need WAN/DNS groups assigned to them for internet access  
If they need a VLAN Tag ID for a Managed Port do add the device here

<b>Device Name*</b>	Passphrase	
<input type="text" value="iPhone Test"/>	<input type="password"/>	
A unique name for the device	Optional. If empty a random password will be generated	
Authentication      Groups      Tags		
<input checked="" type="radio"/> WPA3 <input type="radio"/> WPA2 <input type="radio"/> Wired	<input checked="" type="checkbox"/> wan <input checked="" type="checkbox"/> dns <input type="checkbox"/> lan	<input type="checkbox"/> lan_upstream <input type="checkbox"/> guest
WPA3 is recommended		Assign device to groups for network access
MAC Address      VLAN Tag ID		Only needed for Wired devices on a managed port, set VLAN Tag ID
<input type="text"/> Optional. Will be assigned on connect if empty		<input type="text"/>
Expiration      Delete on expiry		
Never <input type="checkbox"/>		
If non zero has unix time for when the entry should disappear		
<b>Save</b>		

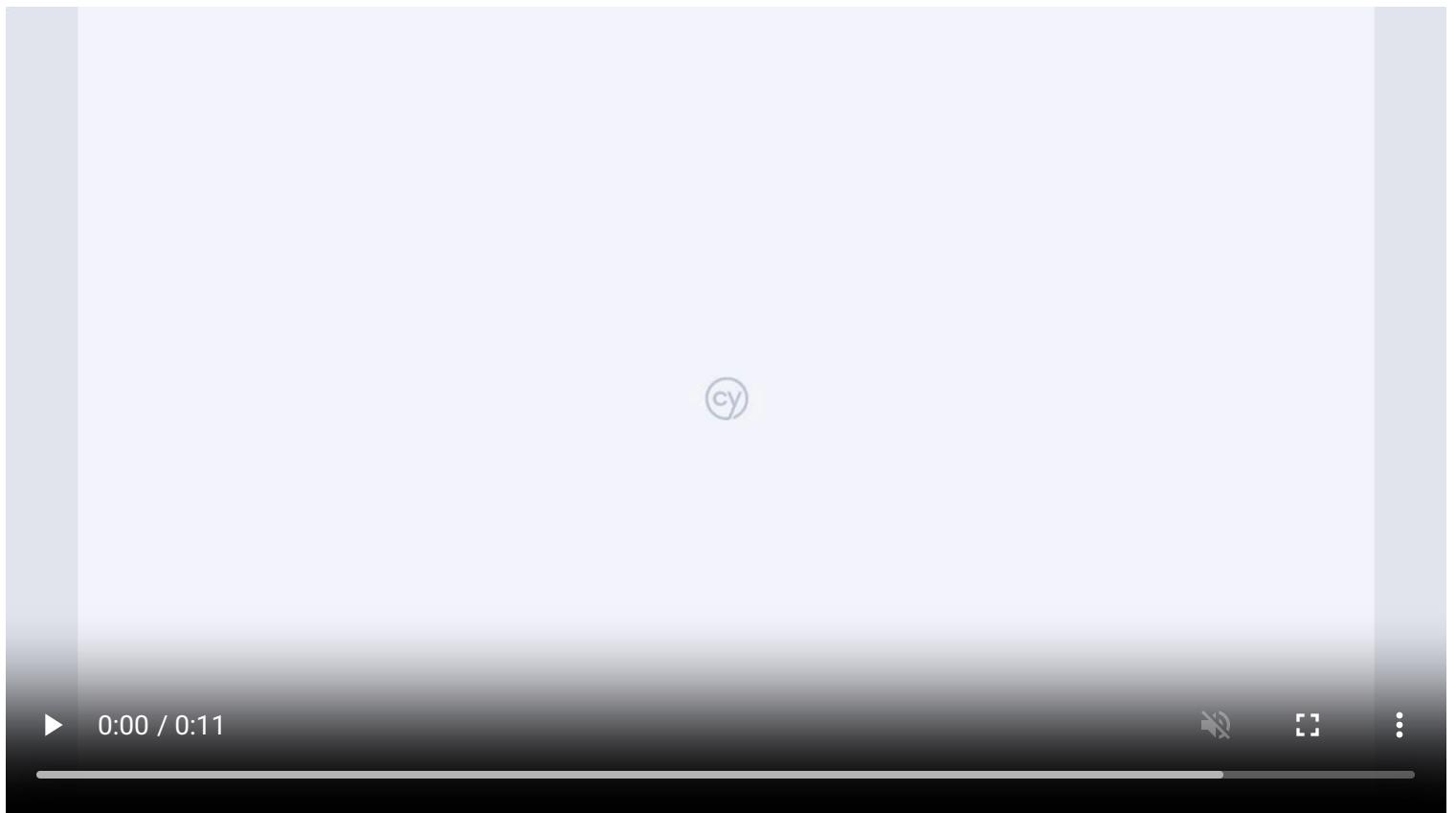
The advanced settings allow you to set MAC address, VLAN Tag ID and other settings for the device.

## Tags

- **lan\_upstream** allows the device to query LAN addresses from [rfc1918](#) upstream of SPR
- **guest** if this is a guest device

Set Expiration (example 1 day) if the device only needs temporary access, also if you want to automatically delete the device on expiry.

# Edit Device



See video on how to add a new tag for a device & use it to block ads for specific devices.

≡ SPR v0.3.1  Docs API ⌂ ↗

Home

Devices

Wifi

DNS

Blocklists/Ad-Block

DNS Log

TRAFFIC

Bandwidth Summary

Bandwidth Timeseries

Traffic

SYSTEM

System Info

**Edit Device**

Name: laptop

IP address: 192.168.2.102

VLAN Tag ID: vlantag

MAC address: 22:22:22:22:22:22 WiFi Auth: WPA2

Groups:

Tags:  PRIVATE

Icon:

Color:

Simple View

Set groups, tags, icon and other settings for your devices in the edit view.

# Copy or Duplicate A Device Entry

Click the ellipse on the end of the device listing, and select duplicate, then rename the device. This will copy the device to a new entry, and wait for a connection for a MAC address to be assigned.

The new entry will have the same WiFi password. This is especially useful for devices using iCloud Keychain Sync, which assumes the same WiFi password for all devices.

## ⚠️ iCloud Keychain Sync and WiFi Passwords

You need to set the same password on each of the iOS devices that are in the same keychain.

Without this, they would sync the wrong password to each other and lock each other out of the network.

Devices

Device	IP Address	MAC Address	Protocol	Encryption	Actions
rpi4	192.168.2.101	11:11:11:11:11:11	WPA3	DNS, LAN, PRIVATE	<ul style="list-style-type: none"><li>...</li><li>Edit</li><li>Duplicate</li><li>Delete</li></ul>
laptop	192.168.2.102	22:22:22:22:22:22	WPA2	DNS, LAN, PRIVATE	<ul style="list-style-type: none"><li>...</li><li>Edit</li><li>Duplicate</li><li>Delete</li></ul>
phone	192.168.2.103	33:33:33:33:33:33	WPA3	DNS, LAN, PRIVATE	<ul style="list-style-type: none"><li>...</li></ul>

# DNS Block Lists & Settings

- [DNS Block Lists](#) Block Ads, Tracking & More
- [DNS & Tags](#) Make block lists select devices by Tag
- [DNS Overrides](#) Make custom DNS Rules
- [DNS Settings](#) Set DNS Privacy settings, DNS Provider
- [DNS Logs](#) Inspect DNS Queries

## Block lists

### Manage Existing Blocklists:

Next to each blocklist, click the three dots open a menu for additional actions you can take, such as:

- Enable: Turn on a blocklist
- Disable: To turn off the blocklist without deleting it.
- Delete: To remove the blocklist entirely.

<b>BlockList Project Ads</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/ads.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/ads.txt</a>	<span>ENABLED</span>	
<b>BlockList Project Youtube domains</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/youtube.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/youtube.txt</a>	<span>ENABLED</span>	<span>FOCUS ↴</span>
<b>Steven Black's Adware &amp; Malware block list</b> <a href="https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts">https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts</a>		<span>Disable</span> <span>Delete</span> <span>New Tag...</span>
<b>BlockList Project Facebook and related services</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/facebook.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/facebook.txt</a>		
<b>BlockList Project Twitter and related services</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/twitter.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/twitter.txt</a>		
<b>BlockList Project Malware List</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/malware.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/malware.txt</a>		
<b>BlockList Project Pornography List</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/porn.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/porn.txt</a>		
<b>BlockList Project Redirect List, often used with spam</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/redirect.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/redirect.txt</a>		
<b>BlockList Project Tracker List for sites that track and gather visitor information</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/tracking.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/tracking.txt</a>		
<b>BlockList Project Everything list</b> <a href="https://raw.githubusercontent.com/blocklistproject/Lists/master/everything.txt">https://raw.githubusercontent.com/blocklistproject/Lists/master/everything.txt</a>		

## Block Lists Tags

It is also possible to apply a Tag to a blocklist. With tags set, the block list will only apply to devices that also have one of the tags assigned. Other devices are not affected by the block list

## DNS Overrides

### Permit Override

Permit overrides can be used to allow domains that are otherwise in a blocklist. The rule can be device specific, by assigning a Client IP, or it can be global.

The rule can also have an expiration assigned for the rule to expire.

Lastly, the permit rule can have an IP address to override a DNS response with an IP address

**FIREWALL**

- Firewall
- Services
- PFW
- Supernetworks

**DNS**

- Blocklists/Ad-Block
- DNS Overrides
- DNS Log
- DNS Log Settings
- Dynamic DNS
- DNS Settings

**TRAFFIC**

- Bandwidth Summary
- Bandwidth Timeseries
- Signal Strength
- Traffic

**EVENTS**

- Events
- Logs
- Notifications

**SYSTEM**

- System Info

**Block Custom Domain** Set rules for DNS queries Add Block +

Domain	IP	Client	Expiration
example.com.	1.2.3.4	192.168.2.101	Never
asdf.com.	1.2.3.4	*	Never

**Permit Domain Overrides** Add Permit +

Domain	IP	Client IP	Expiration
google.com.	8.8.8.8	192.168.2.102	53 years ago

**Add Permit Domain Override**

Type of override  
 Block  Permit  
 Override/Allow DNS lookup with a custom IP address

Domain\*  
 ads.google.com.  
Trailing dot for domain name is to avoid prefix matching

Result IP  
Optional. Set a custom IP address to return for domain name lookup

Client IP  
 192.168.2.102  
Optional. Set a Client IP this rule is applied to

Expiration  
 Never  
If non zero has unix time for when the entry should disappear

**Save**

## Block Override

A block override can also be used to create a rule to block a domain. The Result IP is ignored for a block rule.

## Settings

The DNS Settings can be used to set Client IPs or Domains to keep out of the DNS Logs

- Home
- Devices
- Wifi
- NETWORK**
  - Uplink
  - LAN
  - Containers
- MESH
- VPN
- FIREWALL**
  - Firewall
  - Services
  - PFW
  - Supernetworks
- DNS**
  - Blocklists/Ad-Block
  - DNS Overrides
  - DNS Log
  - DNS Log Settings

**Host Privacy IP List** Client IPs to exclude from logs Add IP +

192.168.1.1	192.168.1.1	X
192.168.1.101	192.168.1.101	X

**Domain Ignore List** Domains to exclude from logs Add Domain +

example.com	X
privatedomain.com	X

## Logs

The DNS Log View can be used to inspect dns requests. Use the buttons on the right to quickly add a permit or deny override rule.

Home

Devices

Wifi

**NETWORK**

- Uplink
- LAN
- Containers
- MESH
- VPN

**FIREWALL**

- Firewall
- Services
- PFW
- Supernetworks

**DNS**

- Blocklists/Ad-Block
- DNS Overrides
- DNS Log**
- DNS Log Settings
- Dynamic DNS

### 192.168.2.105 DNS Log 383 records

Client Date Search Block Type

tv 03/02/2023 Filter domain... Select option

Client	Date	Search	Block Type
501.2.861.291:evres:rnd.in-addr.arpa. rpi4.lan.	3/6/2022, 12:05:34 AM	NOERROR	✓ <input checked="" type="checkbox"/>
caldav.fe.apple-dns.net. 0.0.0.	3/6/2022, 12:05:34 AM	NODATA	✓ <input checked="" type="checkbox"/>
lb._dns-sd._udp.501.2.861.291:evres:rnd.in-addr.arpa. 0.0.0.	3/6/2022, 12:05:29 AM	OTHERERROR	✓ <input checked="" type="checkbox"/>

# Firewall

The screenshot shows the SPR Firewall interface. On the left, there's a sidebar with navigation links: Home, Devices, WiFi, DNS (with Blocklists/Ad-Block), DNS Log, TRAFFIC (Bandwidth Summary, Bandwidth Timeseries, Traffic), and SYSTEM (System Info). A 'Simple View' toggle is also present. The main area is titled 'Endpoints' and contains sections for Port Forwarding, Inbound Traffic Block, and Forwarding Traffic Block. Under Inbound Traffic Block, it says 'Block traffic coming into the network at the PREROUTING stage' with an 'Add IP Block +' button. A table lists a rule: TCP source 0.0.0.0/0 destination 192.168.1.102. Under Forwarding Traffic Block, it says 'Add rules to block traffic at the FORWARDING stage' with an 'Add Forwarding Block +' button. A table lists a rule: TCP source 1.2.3.4 destination 6.7.8.9/24 port 0-65535. There are also sections for Custom Interface Access and Multicast Proxy.

## Firewall Use Cases

Most common to use is *Port Forwarding* and *Inbound Traffic Block*.

- [Port Forwarding](#) is if you want to open a port *from WAN* and forward the traffic to a local device
- [Inbound Traffic Block](#) block access from a specific address and/or destination
- [Forwarding Traffic Block](#) allow/block devices from your local network

## Advanced firewall configuration

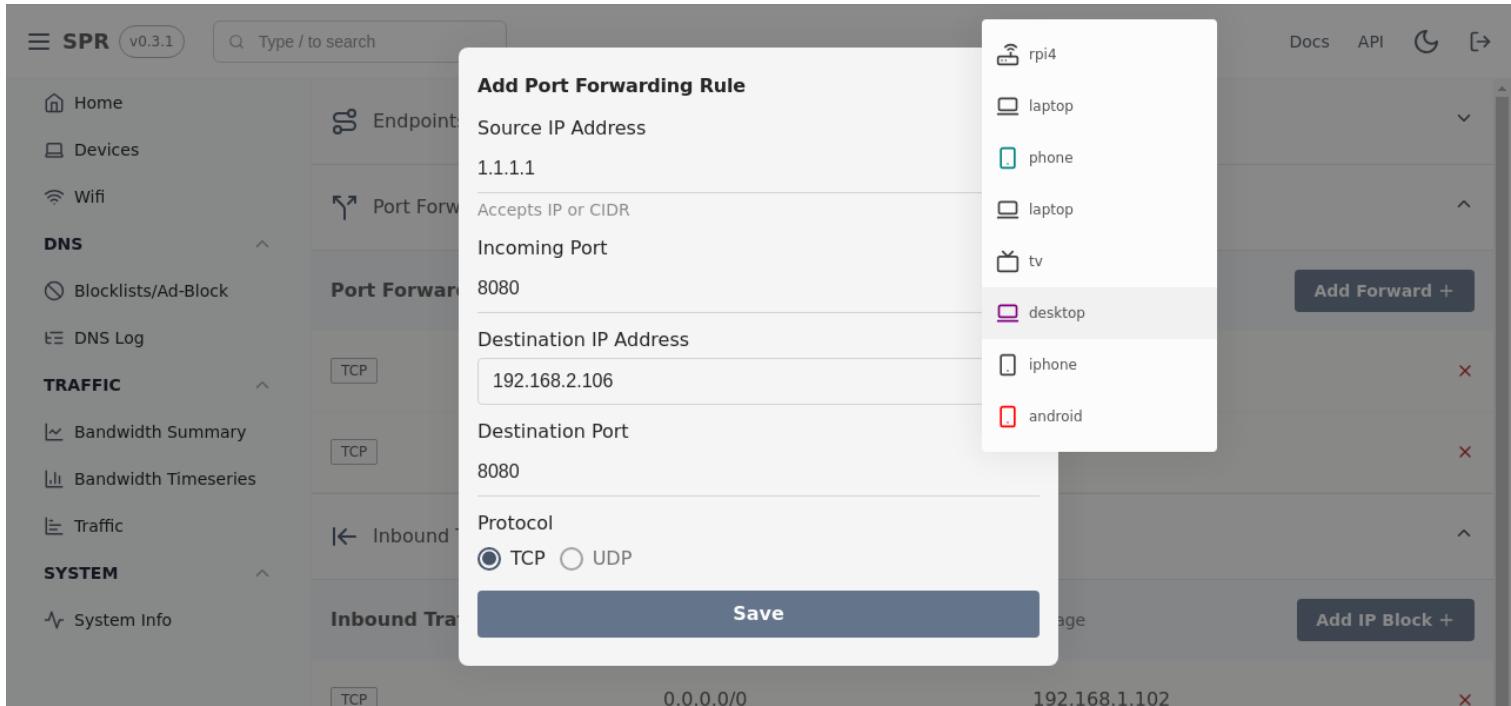
- [Custom Interface Access](#) - Join a docker bridge or custom interface to SPR networking
- [Multicast Proxy](#) -- Configure mDNS, SSDP, and other UDP broadcast services
- [Endpoints](#) - Define one-way connectivity rules

# Port Forwarding

Set rules to forward incoming traffic (DNAT forwarding).

## Add a Port Forward

On the *Firewall* page expand *Port Forwarding* and click *Add Forward*.



In the screenshot all traffic from source address 1.1.1.1 with destination port 8080 is redirected to 192.168.2.106 on port 8080.

## Inbound Traffic Block

Block traffic coming into the network at the PREROUTING stage.

## Add a Inbound Block

On the *Firewall* page expand *Inbound Traffic Block* and click *Add IP Block*.

The screenshot shows the SPR (Software-Defined Firewall) interface. On the left, there's a sidebar with navigation links like Home, Devices, WiFi, DNS, Blocklists/Ad-Block, DNS Log, Traffic (Bandwidth Summary, Bandwidth Timeseries, Traffic), and System (System Info). The main area is titled 'Endpoints' and shows 'Inbound Tra' (Inbound Traffic). A modal window titled 'Add IP Block' is open, prompting for 'Source IP Address\*' (1.1.1.1) and 'Destination IP Address\*' (192.168.2.106). Below these, there's a 'Protocol' section with 'TCP' selected. At the bottom of the modal is a large blue 'Save' button.

In the screenshot all traffic from source address **1.1.1.1** with destination address **192.168.2.106** will be blocked.

# Forwarding Traffic Block

Add rules to block traffic at the FORWARDING stage.

## Add a Forwarding Block

On the *Firewall* page expand *Forwarding Traffic Block* and click *Add Forwarding Block*.

The screenshot shows the SPR web interface with a modal dialog titled "Add Forwarding Block". The dialog contains the following fields:

- Source IP Address\*: 192.168.2.103
- Destination IP Address\*: 192.168.2.106
- Destination Port: 8080
- Protocol: TCP (selected)

At the bottom of the dialog is a "Save" button.

In the screenshot all traffic from source address 192.168.2.103 with destination address 192.168.2.106 on port 8080 will be blocked.

Example use is if you add a device to the lan group but want to block access on some ports.

## Custom Interface Access

Manage network access for custom interface names and control access for docker networks.

The screenshot shows the SPR web interface with a modal dialog titled "Add Custom Interface Rule". The dialog contains the following fields:

- Rule Name: dockerForward
- Friendly name for rule: dockerForward
- Interface\*: 172.17.0.0/24
- Container Address Range\*: 172.17.0.0/24
- Set Route Destination (optional):
- IP address:
- Network Groups & Tags:
  - Edit Groups +
  - Edit Tags +

At the bottom of the dialog is a "Save" button.

Use this to forward traffic to an interface running in a separate container.

See more in the repository for the [spr-mitmproxy plugin](#) for how this feature can be used.

# Multicast Proxy

Set multicast ip:port addresses to proxy between devices. This allows discovery services such as SSDP/Zeroconf and mDNS in SPR. By default only SSDP and mDNS are on. You may need to enable PTP or other services for streaming and device sync. If something is not working as expected, let us know, by [filing an issue](#)

The screenshot shows the Firewall configuration interface with the following details:

- Left Sidebar:** FIREWALL (Firewall, Services, PFW, Supernetworks), DNS (Blocklists/Ad-Block, DNS Overrides, DNS Log, DNS Log Settings, Dynamic DNS), TRAFFIC (Bandwidth Summary, Bandwidth Timeseries, Signal Strength, Traffic), EVENTS (Events, Logs, Notifications), SYSTEM.
- Top Bar:** Endpoints, Port Forwarding, Inbound Traffic Block.
- Inbound Traffic Block:** Block traffic coming into the network at the PREROUTING stage. A table shows a rule for TCP port 0.0.0.0/0 to 192.168.1.102.
- Forwarding Traffic Block:** I→ Forwarding Traffic Block.
- Forwarding Traffic Rule:** Add Multicast Service Rule for address 224.0.1.129 port 319. The "Multicast IP Address" dropdown is open, showing options: mDNS, SSDP, PTP events, and PTP general. The "Add Multicast Service +" button is highlighted with a red oval.
- Multicast Proxy:** Set ip:port addresses to proxy. A message states: "There are no multicast proxy rules configured yet".

A future release will enable tags to apply proxying to only a subset of devices.

# Endpoints

Service Endpoints serve as helpers for creating other firewall rules, as well as one-way connectivity from devices to the endpoint when they share a tag.

When devices are in the same group they have full access to each other. Device in the LAN group have one-way access to all devices via NAT.

Endpoints create a quick way to apply one-way connectivity to a specific destination, by applying tags.

## 1. Create a service definition

The screenshot shows the SPR (Service Provider) interface version v0.3.2-dev. The left sidebar includes sections for Home, Devices, Wifi, NETWORK (Uplink, LAN, Containers, MESH, VPN), FIREWALL (Firewall, Services, PFW, Supernetworks), DNS (Blocklists/Ad-Block, DNS Overrides, DNS Log, DNS Log Settings), and Multicast Proxy. The main area is titled "Endpoints" and describes them as helpers for building One-Way Firewall Rules and short names. A red circle highlights the "Add Service Endpoint +" button. A modal dialog titled "Add Service Endpoint" is open, prompting for a Name (nzyme-web), IP Address (192.168.2.58), Custom Interface (mitmweb0), Port (22900), and Protocol (TCP). The "Save" button is at the bottom of the dialog. Other buttons in the background include "Add Custom Interface Rule +", "Add Multicast Service +", and "Inbound Traffic Block".

## 2. Apply a tag

The screenshot shows the SPR (Service Protection Rule) web interface. The left sidebar has a 'FIREWALL' section selected, containing options like Firewall, Services, PFW, Supernetworks, DNS, and Dynamic DNS. The main area is titled 'Endpoints' and describes Service Endpoints for building One-Way Firewall Rules and short names. It lists an endpoint named 'nzyme-web' (TCP port 22900) with an IP of 192.168.2.58. A red circle highlights the three-dot menu icon next to the endpoint entry. A modal window titled 'Add Tag' is open over the list, showing a text input field with the value 'nzyme'. A 'Save' button is visible at the bottom right of the modal.

### 3. Apply the same tag to the device that should have access to the endpoint

SPR v0.3.2-dev  Docs API ⌂ ⌂

Home

Devices

Wifi

**NETWORK**

Uplink

LAN

Containers

MESH

VPN

**FIREWALL**

Firewall

Services

PFW

Supernetworks

**DNS**

Blocklists/Ad-Block

DNS Overrides

DNS Log

DNS Log Settings

Dynamic DNS

DNS Settings

**TRAFFIC**

Bandwidth Summary

Bandwidth Timeseries

Signal Strength

Traffic

**EVENTS**

Advanced View

**Edit Device**

**Name**  
nexmon-pi-haxlab

**IP address**  
192.168.2.218

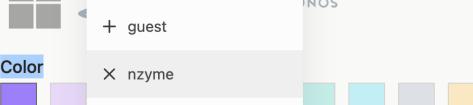
**VLAN Tag ID**

**MAC address** e4:5f:01:fd:a1:75 **WiFi Auth** N/A

**Groups**

**Tags** NZYME

**Icon**     

**Color** 

**Expiration** Never

If non zero has unix time for when the entry should disappear

**Delete on expiry**  Remove device

4. Example of device access before and after it has the tag

```
ubuntu@haxlab0:~$ curl -vv -m 3 192.168.2.58:22900
* Trying 192.168.2.58:22900...
* Connection timed out after 3001 milliseconds
* Closing connection 0
curl: (28) Connection timed out after 3001 milliseconds
ubuntu@haxlab0:~$ # now set the tag on the device
ubuntu@haxlab0:~$ curl -vv -m 3 192.168.2.58:22900
* Trying 192.168.2.58:22900...
* Connected to 192.168.2.58 (192.168.2.58) port 22900 (#0)
> GET / HTTP/1.1
> Host: 192.168.2.58:22900
> User-Agent: curl/7.88.1
> Accept: */*
>
* Empty reply from server
* Closing connection 0
curl: (52) Empty reply from server
ubuntu@haxlab0:~$
```

The other side, 192.168.2.58, does not gain access to the device on 192.168.2.218 beyond receiving TCP connections on :22900.

# Link Management

## Uplink Configuration

[Set DHCP settings or assign a Static IP](#)

[Enable a Wireless Uplink](#)

## Downlink Configuration (LAN)

[Enable Wired LAN Interface](#)

[Enable VLAN Trunk Port](#)

## Multiple Uplinks

SPR experimentally supports multiple uplink interfaces. It will load balance new connections across them based on addresses. No additional setup is required beyond enabling multiple uplinks.

## Set Uplink Interface IP

It is possible to configure DHCP settings for an uplink interface.

The screenshot shows the SPR (Simple Personal Router) web interface. The left sidebar contains navigation links for Home, Devices, WiFi, NETWORK (with Uplink selected), LAN, Containers, MESH, VPN, FIREWALL (with Firewall selected), Services, PFW, Supernetworks, DNS (with Blocklists/Ad-Block selected), DNS Overrides, DNS Log, DNS Log Settings, Dynamic DNS, and DNS Settings. The TRAFFIC section is also present.

The main content area displays the "Uplink Configuration" page. It lists two uplink interfaces: **eth0** (Uplink, Enabled) and **ppp0** (Uplink, 11.22.33.44, Enabled). Below these are "Other Interfaces": **eth0.123** (Other, 192.168.99.99) and **wlan0** (DHCP Settings, Manually Set IP). The **wlan1** interface is listed but not currently active.

A context menu is open over the **ppp0** row, with the "..." option circled in red. The menu options are: **Modify Interface**, **Modify IP Settings**, **Configure Wireless**, and **Configure PPP**.

A configuration dialog is open for the **Configure eth0** section. It includes fields for **DHCP Settings** (Manually Set IP selected) and **Assign IP** (192.168.1.1/24). Below that is the **Assign Router** field (192.168.1.1). At the bottom is a large blue **Save** button.

# Enable a WiFi Uplink

In the uplink view, click Configure Wireless

**Other Interfaces**

Interface	Type	IP Address	...
eth0.123	Other	192.168.99.99	...
wlan0	Other	192.168.22.22	...
wlan1	AP	192.168.2.1	<ul style="list-style-type: none"><li>↓↑ Modify Interface</li><li>↓↑ Modify IP Settings</li><li>Configure Wireless (circled)</li><li>Configure PPP</li></ul>

And then fill out your settings

**Uplink Configuration**

eth0	Uplink	✓ ENABLED	...	
ppp0	Uplink	11.22.33.44	✓ ENABLED	...

**Other Interfaces**

wlan0	<b>Configure wlan0</b>	...
wlan1	<b>Configure wlan1</b>	...

**Configure wlan0**

SSID: ssid\_AABBCC

BSSID: 33:33:33:33:33:33

Assign

Auth: WPA-PSK WPA-PSK-SHA256 SAE

Password:

Status: Enabled

**Save**

**Configure wlan1**

SSID: ssid\_AABBCC

BSSID: 33:33:33:33:33:33

Assign

Auth: WPA-PSK WPA-PSK-SHA256 SAE

Password:

Status: Enabled

**Save**

# Enable Wired LAN Interface

Currently SPR only supports one Wired LAN interface. An upcoming release will support multiple links.

## ▼ Adding a USB Ethernet on a Pi

The [UE 300](#) has been tested to run near line speed on the Pi (950Mbps). The [U3 330](#) also works and provides additional USB 3.0 ports.

To enable it, click the ellipse and save

The screenshot shows the SPR network configuration interface. On the left, there's a sidebar with sections like NETWORK, FIREWALL, and DNS. The main area lists network interfaces: eth0.123 (Uplink, Other, 192.168.99.99, ENABLED), eth0 (Uplink), ppp0 (Uplink, 11.22.33.44), wlan0 (Other, 192.168.22.22), and wlan1. A context menu is open over the eth0 row, with the '...' button circled in red. A modal window titled 'Configure eth0' is displayed, containing 'Update Interface' with a checked 'Enabled' checkbox, 'Set Interface Type' set to 'Downlink', and a large blue 'Save' button at the bottom.

# Enable VLAN Trunk Port

SPR has basic support for the downlink interface being a VLAN Trunk Port. This requires a managed switch. Refer to your switch manual for how to configure the VLAN assignment with 802.1Q.

**LAN Link Configuration**

Note: API support for multiple wired LAN interfaces is an upcoming feature.  
For now, ensure the wired LAN is synchronized with the config/base/config.sh LANIF entry.

Interface	Type	IP Address	Status	Actions
eth0.123	Other	192.168.99.99	ENABLED	...
eth0	Uplink			...
ppp0	Uplink	11.22.33.44		...
wlan0	Other	192.168.22.22		...
wlan1				<b>Configure eth0</b>

**Configure eth0**

Update Interface  
 Enabled

Select option  
Downlink  
 VLAN Trunk Port  
Other

**Save**

**NETWORK**

- Uplink
- LAN
  - Containers
- MESH
- VPN

**FIREWALL**

- Firewall
- Services
- PFW

**DNS**

- Blocklists/Ad-Block
- DNS Overrides
- DNS Log
- DNS Log Settings
- Dynamic DNS
- DNS Settings

Next, edit your device that is connected to the managed port and set the VLAN ID corresponding to that port.

**Edit Device**

**Name**  
rpi4

**IP address**  
192.168.2.102

**VLAN Tag ID**  
100

# Installing Extensions

Plugins are a mechanism to integrate additional containers running on the same device with SPR's API and UI. Many of the built-in containers are deployed as plugins, and third party plugins are fully supported.

Note: we plan to streamline plugin installation in a future release.

## Community Plugins

See the [curated plugin list](#)

## Plugin Installation

A plugin can be enabled by navigating to the Plugins page and selecting [Add Plugin](#).

The screenshot shows the SPR web interface with the 'Plugins' section selected. A modal window titled 'Add a new Plugin' is open in the foreground. The modal fields are as follows:

- Name:** (Input field)
- URI:** (Input field) - Description: 'Plugin will be @ "http://spr/plugins/URI"'
- UNIX Path:** (Input field) - Description: 'Plugin pathname for unix socket'
- ComposeFilePath:** (Input field) - Description: 'Plugin pathname for unix socket'
- Save** (Blue button)

The background shows a list of installed plugins:

- dns-block-extension** (Version 0.3.2) - Status: Enabled (green switch)
- dns-log-extension** (Version 0.3.2) - Status: Enabled (green switch)
- plugin-lookup** (Version 0.3.2) - Status: Enabled (green switch)
- wireguard** (Version 0.3.2) - Status: Enabled (green switch)
- dyndns** (Version NONE) - Status: Enabled (green switch)
- db** (Version 0.3.2) - Status: Enabled (green switch)
- PPP** (Version NONE) - Status: Enabled (green switch)
- WIFILINK** (Version NONE) - Status: Enabled (green switch)
- nexmon** (Version LATEST) - Status: Enabled (green switch)
- PLUS Plugins**
- PFW** (Version 0.3.0) - Status: Enabled (green switch)

**Name:** this is the plugin name

**URI:** This is used when the plugin extends the SPR API with additional API calls.

- Ex: specifying `nexmon` means that the API calls the plugin adds will be reachable under `http://spr/plugins/nexmon`

**UNIX Path:** This is the API's path to the unix socket the plugin provides

- Ex: `/state/plugins/nexmon/socket`

**ComposeFilePath:** This is the relative path to the docker compose path, so SPR auto-starts the plugin

- Ex: `plugins/spr-nexmon/docker-compose.yml`

- **Note** Currently `configs/base/custom_compose_paths.json` also needs to be updated to allow this path.

## Auto-starting a plugin

For now, a strict whitelist is used to determine which compose file paths to run. To add your plugin to the whitelist, update:

`configs/base/custom_compose_paths.json`

- Example

```
["plugins/spr-nexmon/docker-compose.yml", "plugins/test/docker-compose.yml.yml"]
```

# Troubleshooting

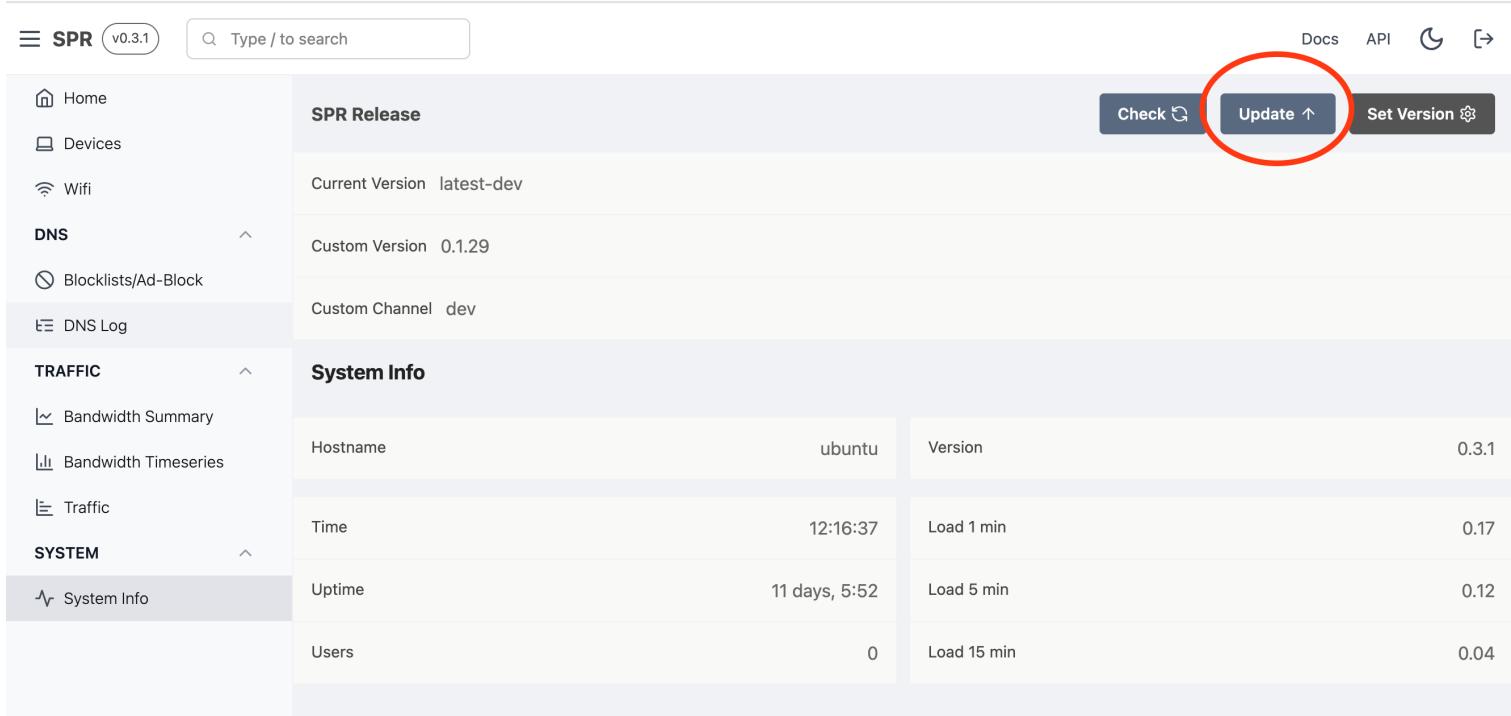
Something broken? [File a ticket](#)

See the [Troubleshooting FAQ](#)

# Updating SPR

## How To Update?

SPR can be updated from the `System Info` view



The screenshot shows the SPR web interface with a sidebar on the left containing links for Home, Devices, WiFi, DNS, Blocklists/Ad-Block, and DNS Log. The main content area is titled "SPR Release" and shows the current version as "latest-dev". Below that, it shows a custom version of "0.1.29" and a custom channel of "dev". At the top right of this section are three buttons: "Check ↗", "Update ↑" (which is circled in red), and "Set Version ⚙". The bottom section is titled "System Info" and contains a table with four rows. The columns are Hostname, Version, Time, Load 1 min, Uptime, Load 5 min, and Load 15 min. The data is as follows:

	Hostname	Version	Time	Load 1 min	Uptime	Load 5 min	Load 15 min
Bandwidth Summary	ubuntu	0.3.1					
Bandwidth Timeseries			12:16:37				
Traffic				Load 1 min			0.17
System Info					11 days, 5:52	Load 5 min	0.12
					0	Load 15 min	0.04

## Auto Updates

Auto updates are not currently enabled and may be added in a future release.

## Release Channels

The main release channel is `main`.

The `dev` branch publishes to the `dev` channel. This is not stable and not recommended outside of development.

It's also possible to roll back to a previous version by setting it.

## Updating from the command line

```
sudo -s  
cd /home/spr/super/  
docker compose pull  
docker compose up -d
```

# WiFi Settings

## SPR Compatibility

Device isolation today requires a card with AP/VLAN support. This is presented under the Interfaces tab under wireless settings.

The screenshot shows the SPR web interface version 0.3.1. The left sidebar has sections for Home, Devices, and WiFi. Under WiFi, there are sections for NETWORK (Uplink, LAN, Containers, MESH, VPN), FIREWALL (Firewall, Services, PFW), and DNS (Blocklists/Ad-Block, DNS Overrides, DNS Log). The main content area has tabs for Interfaces, Clients, Scan, and Radio Settings. The Interfaces tab is selected, showing a list of interfaces: phy0 (managed), phy1 (AP: TestAP), and phy2 (managed). The phy0 interface is expanded, showing the 'SPR compatibility' tab selected, along with other tabs for devices, interface modes, commands, ciphers, extended features, device supports, bands, and other. The 'SPR compatibility for phy0' section lists three items: 5GHz (Recommended for maximum speed), WPA3/SAE (Recommended for better security), and AP/VLAN (Required to create virtual interfaces). The phy1 and phy2 interfaces also have expandable sections.

## SSID

The SSID name can be set by navigating to WiFi ⇒ Radio Settings and typing in the new SSID. After hitting enter or leaving the field, it will update and restart with the new SSID.

Home Devices WiFi

NETWORK ▾

- Uplink
- LAN
- Containers
- MESH
- VPN

FIREWALL ▾

- Firewall
- Services
- PFW
- Supernetworks

DNS ▾

- Blocklists/Ad-Block

Interfaces Clients Scan Radio Settings

### Wifi Interface

WiFi Interface wlan1

#### Channel Selection

Use the Channel Selection to make sure the correct Frequency, Bandwidth & Channel is set. This will update your HostAP config.

Frequency Band	Bandwidth	Channel
5 GHz	80 MHz	36

Wifi 6 (AX)  Enable WPA1 SSID

**Advanced HostAP Config wlan1**

ssid YourSSIDHere  

country code US

Update All HT/VHT Capabilities Disable Radio Interface Reset Config

**Save ✓**



# Channel Configuration

Home Devices WiFi

NETWORK ▾

- Uplink
- LAN
- Containers
- MESH
- VPN

FIREWALL ▾

- Firewall
- Services
- PFW
- Supernetworks

Interfaces Clients Scan Radio Settings

### Wifi Interface

WiFi Interface wlan1

#### Channel Selection

Use the Channel Selection to make sure the correct Frequency, Bandwidth & Channel is set. This will update your HostAP config.

Frequency Band	Bandwidth	Channel
5 GHz	80 MHz	149

Wifi 6 (AX)  Enable WPA1 SSID

**Advanced HostAP Config wlan1**

ssid YourSSIDHere  

country code US

Update All HT/VHT Capabilities Disable Radio Interface Reset Config

**Save ✓**

You can also run a scan with an unused radio to try to determine available channels

The screenshot shows the SPR (Software Radio Project) web interface. The left sidebar has sections for Home, Devices, WiFi (selected), NETWORK (Uplink, LAN, Containers, MESH, VPN), FIREWALL (Firewall, Services), and a search bar. The main content area has tabs for Interfaces, Clients, Scan (selected), and Radio Settings. Under Scan, it shows an interface named wlan0 with SSID ssid\_AABBCC, Channel 1, Freq 2.41 GHz, Signal -67 dBm, Auth PSK, Model ABCDEF / 123456, and Device Name ABCdev. A red circle highlights the 'Scan' button in the top right corner of the main content area.

## Enabling WiFi 6

SPR will detect if WiFi 6 is supported. If so, select the WiFi 6 button and hit save. This will configure the AP settings to support WiFi 6 protocol settings.

## Legacy WPA1 Support

Unfortunately, some older devices do not support WPA2 or WPA3. If your card supports it, SPR can enable a secondary SSID with WPA1. It should have a different SSID name than the main SSID.

**Wifi Interface**

**WiFi Interface**: wlan1

**Channel Selection**

Use the Channel Selection to make sure the correct Frequency, Bandwidth & Channel is set. This will update your HostAP config.

Frequency Band	Bandwidth	Channel
5 GHz	80 MHz	149

Wifi 6 (AX)    Enable WPA1 SSID

Extra BSS Name: YourSSIDHere-WPA1-Legacy

**Advanced HostAP Config wlan1**

**Buttons:** Save ✓, Update All HT/VHT Capabilities, Disable Radio Interface, Reset Config

# Tuning Capabilities

There are various capabilities available for 802.11 HT , 802.11AC (VHT), and 802.11AX (HE). SPR will auto-detect and assign these capabilities with the "Update All" button. They can also be selectively modified.

If you need additional options, [please file an issue](#)

# Troubleshooting

If the wifi fails to start, try the following:

1. Hit Reset Config
2. Hit Enable HostAP
3. Re-configure your SSID name and channel settings and hit save

# Overview

[PLUS](#) is our add-on package that lets users support the SPR project.

## Site VPN

- [Configuration](#)
- [Routing traffic to the Site VPN](#)

## Mesh

- [Installing a Mesh Node](#)

## Programmable Firewall (PFW)

- [Configuration](#)
- scheduling dns block rules aka focus mode
- mitmproxy setup guides

## Setup Guide

After purchasing PLUS, an e-mail should arrive with the PLUS token.

Go to the plugins view, and paste the token to activate PLUS plugins.

The screenshot shows the SPR interface with the 'Plugins' section selected. Under 'PLUS Plugins', there is a 'PFW' plugin version 0.3.1 with its toggle switch turned on. Below it is a 'Reset PLUS Token' section with a text input field containing 'YOUR\_PLUS\_TOKEN' and two buttons: 'Update token' (circled in red) and 'Verify token'.

Next, enable the plugins you'd like to use. Today PLUS ships with two extensions -- Mesh and PFW

The screenshot shows the SPR interface with the 'Plugins' section selected. Under 'PLUS Plugins', there are two sections: 'PFW' version 0.3.0 with its toggle switch turned off (circled in red), and 'MESH' with 'NONE' selected and its toggle switch turned off.

# WiFi Mesh Support

Today Mesh support has been tested primarily with wired backhaul. SPR does not use 802.11s mesh networking for wireless backhaul but may do so in a future release

## Install

To get started, install SPR on your mesh device, and enable PLUS and the Mesh plugin.

Next, visit the Mesh page and generate a leaf token.

The screenshot shows the SPR web interface with the following details:

- Header:** SPR v0.3.1, search bar, Docs, API, and navigation icons.
- Left Sidebar:** Home, Devices, Wifi, NETWORK (with Uplink, LAN, Containers, MESH selected), VPN, FIREWALL (with Firewall, Services).
- Middle Content:**
  - Mesh Setup:** Configure downstream routers for mesh networking. Only wired backhaul is supported for now. Includes "Add Leaf Router +".
  - Device Token:** Generate an API token to use this device as a leaf router. Includes "Generate API Token +".
- Right Side:** There are no leaf routers configured yet. Includes "Sync SSID Across Devices: TestAP" button.

Copy the generated token, and then navigate back to the main SPR router.

Select the IP of the mesh router, and configure it with the key.

# PFW Programmable Firewall

PFW allows for advanced firewall rules using policy based and DNAT.

It can also be used to schedule actions on the router, like joining groups or receiving tags at a specified time.

The screenshot shows the PFW configuration interface. On the left, there's a sidebar with navigation links: Home, Devices, Wifi, NETWORK (with Uplink, LAN, Containers, MESH, VPN), FIREWALL (with Firewall, Services), PFW (selected), Supernetworks, DNS (with Blocklists/Ad-Block, DNS Overrides). The main area is titled "Flows" and contains two entries:

- Always block**: TCP, 0.0.0.0, 213.24.76.23, 0-65535
- Set focus mode, midnight - 6pm**: WEEKDAYS, 00:00, 18:00, 192.168.2.14, FOCUS

To the right, there's a configuration panel with fields for Name (NewFlow), When..., Then..., and buttons for Save (✓) and Reset (✗).

## Configuration

The screenshot shows the PFW configuration interface with the "Add trigger to flow" dialog open. The sidebar on the left includes options for VPN, FIREWALL (selected), Firewall, Services, PFW, Supernetworks, DNS, Blocklists/Ad-Block, DNS Overrides, DNS Log, and DNS Log Settings. The dialog has two tabs:

- Always**: Always run the selected trigger
- Date**: Trigger on selected date and time (MON,TUE,WED, 10:00, 11:00)

Containers   WEEKDAYS   00:00   18:00   192.168.2.14   FOCUS   Add card +

MESH   VPN   FIREWALL   Firewall   Services   PFW   Supernetworks   DNS   Blocklists/Allowlists   DNS Overrides   DNS Log   DNS Log Settings   Dynamic DNS   DNS Settings

**Add action to flow**

<b>Block</b> Block from source address or group to destination address  TCP   0.0.0.0   1.2.3.4   *	<b>Forward</b> Forward for specified source to destination address and port  TCP   0.0.0.0   0.0.0.0   *   0.0.0.0   *
<b>Forward all traffic to Interface, Site VPN or Uplink</b> Forward traffic over a Site VPN Gateway, an Uplink, or a Custom Interface  0.0.0.0   0.0.0.0   *   1.2.3.4	<b>Port Forward to Interface, Site VPN or Uplink</b> Forward traffic over a Site VPN Gateway, an Uplink, or a Custom Interface  TCP   0.0.0.0   0.0.0.0   *   *   1.2.3.4
<b>Set Device Groups</b> A device joins a group only when conditions 	<b>Set Device Tags</b> Assign device tags when conditions are met 

# Site VPN

The PFW plugin from PLUS adds support for Site VPN Destinations with Wireguard.

## Configuration

Once PFW is enabled, a new menu should appear under the VPN view.

The screenshot shows the SPR (v0.3.2) interface with the 'VPN' tab selected. On the left, there's a sidebar with sections for Home, Devices, Wifi, NETWORK (Uplink, LAN, Containers, MESH), and FIREWALL (Firewall, Services, PFW, Supernetworks). The main content area is titled 'Wireguard' and displays a public key: `RwgHaTBwMywZCYxqEfoU2Le5Tm4Y7r7kiBgRq9k7iE=`. It has sections for 'Default Endpoints' (with an 'Add Endpoint +' button) and 'Peers' (with an 'Add Peer +' button). Below these is a 'Site-To-Site VPNs' section with an 'Add a Site +' button, which is circled in red.

Next, fill out the wireguard details. The 'Interface Address' should be the Peer IP for SPR that the remote wireguard has assigned.

Wireguard

Wireguard is listening on port 51280 with PublicKey: `RwgHaTBwMywZCYxqEfoU2Le5Tm4Y7r7kiBgRq9k7iE=`

**Default Endpoints** Set default endpoint clients should connect to

**Add Endpoint +**

**Peers**

There are no peers

**Site-To-Site VPNs**

There are no site V

**Add Site VPN**

**Remote Endpoint**  
1.2.3.4:51820

The remote wireguard VPN address and port, "1.2.3.4:51820"

**Remote Peer's PublicKey**

RemotePeerPubKey

The Public Key for the Remote Endpoint

**Interface Address**  
192.168.10.10

The local interface address

**PresharedKey**

PSK

Leave empty if no PresharedKey is configured

**PrivateKey**

PrivateKey **Generate**

Local Site's Private Key

**Save**

Home  
Devices  
Wifi  
NETWORK  
Uplink  
LAN  
Containers  
MESH  
VPN  
FIREWALL  
Firewall  
Services  
PFW  
Supernetworks  
DNS  
Blocklists/Ad-Block  
DNS Overrides  
DNS Log  
DNS Log Settings  
Dynamic DNS  
DNS Settings  
TRAFFIC  
Bandwidth Summary

# Routing Traffic

Next, we can use the **PFW** extension to create Policy rules that redirect traffic over the site interface.

Select the **Forward all traffic to ...Site VPN or Uplink** action

The screenshot shows the SPR v0.3.2 web interface. On the left, there's a sidebar with various network-related sections like Home, Devices, WiFi, NETWORK (Uplink, LAN, Containers, MESH), FIREWALL (Firewall, Services, PFW), DNS (Supernetworks, Blocklists/ACLs, DNS Overrides, DNS Log, DNS Log Settings, Dynamic DNS), and TRAFFIC (Bandwidth Summary). The main area is titled 'Flows' and shows 'No flows configured'. A 'Name' field is set to 'NewFlow'. Below it, a 'When...' section has a 'Always' condition. An 'Add action to flow' dialog is open, listing several actions:

- Block**: Block from source address or group to destination address. Parameters: TCP, 0.0.0.0, 1.2.3.4, \*.
- Forward**: Forward for specified source to destination address and port. Parameters: TCP, 0.0.0.0, 0.0.0.0, \*, 0.0.0.0, \*.
- Forward all traffic to Interface, Site VPN or Uplink**: This option is highlighted with a large red circle. Parameters: 0.0.0.0, 0.0.0.0, \*, 1.2.3.4.
- Forward traffic over a Site VPN Gateway, an Uplink, or a Custom Interface**: Parameters: 0.0.0.0, 0.0.0.0, \*, 1.2.3.4.
- Port Forward to Interface, Site VPN or Uplink**: Parameters: TCP, 0.0.0.0, 0.0.0.0, \*, \*, 1.2.3.4.
- Set Device Groups**
- Set Device Tags**

Fill out the parameters, selecting the Client/Group/Tag to apply the rule to. Select the **site0** destination interface. The **destination** can be left blank for Site VPNs.

**Flows**

No flows configured

**Name**  
test  
Use a unique name to identify this flow

**When...**

Always ⓘ

**Then...**

Forward all traffic to Interface, Site VPN or Uplink ⓘ

192.168.200.6 | 0.0.0.0 | SITE0 | \*

**Save ✓** **Reset ✕**

Now outbound traffic from the selected device will go out over the Site VPN.

## DNS Split Tunnel

Note that DNS request still go through the main router DNS service. In a future release, a parameter will be available to also route DNS queries through the Site VPN.

## Verifying connectivity

In an upcoming release, status will be visible in the UI. For now, users can run the following command on SPR

```
docker exec -it superwireguard wg show
```

```
interface: wg0
  public key: PUBKEYPUBKEY=
  private key: (hidden)
  listening port: 51280
```

```
interface: site0
  public key: PUBKEYPUBKEY=
  private key: (hidden)
  listening port: 52538
```

```
peer: PEERPUBKEYPUBKEY=
```

```
presharded key: (hidden)
endpoint: 1.2.3.4:51280
allowed ips: 0.0.0.0/0
latest handshake: 11 seconds ago
transfer: 272.20 MiB received, 50.05 MiB sent
```

And check that the `site0` interface has the correct peer IP

```
docker exec -it superwireguard ip -br addr show site0
```

site0	UNKNOWN	192.168.241.2/32
-------	---------	------------------

# Base

The base service establishes the SPR system configuration and initializes the firewall. It also configures performance tuning parameters for managing irq balancing.

The main configuration variables are found in [config/base/config.sh](#)

## Firewall Configuration

The firewall uses NFTTable rulesets defined in [base/scripts/nft\\_rules.sh](#)

The forwarding and input policies are default drop.

The following ports can be exposed to WAN by enabling UPSTREAM\_SERVICES\_ENABLE in [configs/base/config.sh](#):

- sshd (tcp 22),
- api (port 80 or 443 with SSL),
- iperf3 (tcp 5201)
- wireguard (udp 51280)

OR by updating them in the UI under the Firewall settings.

On LAN the following services are available:

- DHCP tied to the authenticated MAC address over WiFi or all wired LAN devices
- DNS for devices in the dns\_access group
- 1900, 5353 multicast repeater to all devices for SSDP and MDNS
- The API (port 80, 443)
- SSH (tcp port 22)

Routing to devices on the LAN or to WAN only happens for authenticated, approved MAC addresses.

[⇒ View the code](#)

# Database

The database service is powered by [bolt](#).

Bolt is a pure Go key/value store inspired by Howard Chu's LMDB project. The goal of the project is to provide a simple, fast, and reliable database for projects that don't require a full database server such as Postgres or MySQL."

The API communicates with SPR over a unix socket, located at [state/plugins/db/socket](#) and is [documented here](#).

⇒ [View the code](#)

# DHCP

DHCP is powered by [CoreDHCP](#). CoreDHCP is an extensible DHCP client and server purely written in golang.

SPR maintains a fork with one additional plugin:

## Tiny Subnets

This plugin will call into the core API to get DHCP client addresses and options to assign.

After receiving a DHCP request, it will query the API with the following parameters

```
type DHCPRequest struct {
    MAC      string
    Identifier string
    Name     string
    Iface    string
}
```

And the API will respond with IP, Router, DNS, and Lease time to construct a packet

```
type DHCPResponse struct {
    IP        string
    RouterIP string
    DNSIP    string
    LeaseTime string
}
```

[⇒ View the code](#)

# DHCP Client

To remove attack surface, SPR uses CoreDHCP as a memory-safe DHCP client. If that fails, it will fall back to the Ubuntu dhclient implementation.

## IPv6

Note: IPv6 is not supported at this time by the CoreDHCP client.

 [View the code](#)

# DNS

CoreDNS is an extensible golang dns server. It powers kubernetes and many other services today. The server is configured to use DNS over HTTPS upstream for all requests and handles DNS for wireguard and wifi clients.

## The DNS Service runs off CoreDNS with custom plugins

- Ad Blocking
- DNS Rebinding Protection
- JSON Log

## DNS Configuration Files:

- Corefile

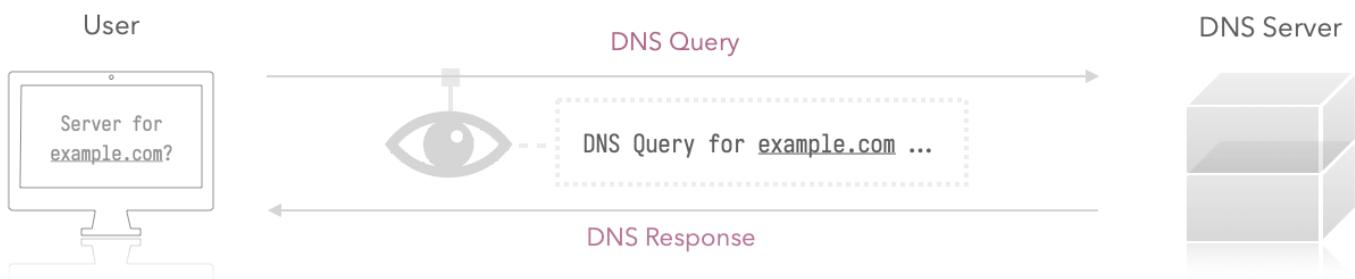
## Local Mappings

CoreDNS is configured to use Local Mappings of names from DHCP. If a device DHCPs with a client name of "test", "test.lan" resolves with the LAN IP address.

```
hosts /state/dns/local_mappings {  
    ttl 60  
    reload 30s  
    fallthrough  
}
```

## DNS over HTTPS

The config for SPR is using DNS over HTTPS by default for all upstream requests.



[View the code](#)

# Dynamic DNS

The DYNDNS plugin runs [godns](#) under a docker container, under the default docker bridge.

This plugin is dynamically started when enabled.

It can be configured under the [Dynamic DNS](#) view when enabled.

[View the code](#)

# Extensions

The API can be extended with plugins. Containers can expose unix sockets that the API will reverse proxy into and make available for clients. See the API docs for the [default API extensions](#) or check the [example plugin](#) in the super repository.

A list of curated plugins is [available here](#)

Read more about how plugins are configured in the [API overview](#).

# Multicast UDP Proxy

The multicast proxy forwards multicast packets across VLANs and network interfaces. The service is written in golang and uses transparent UDP sockets to forward packets.

By default it provides MDNS and SSDP. More multicast services can be configured under the multicast firewall settings.

[⇒ View the code](#)

# Packet Logs

This service receives packets over NFLog netlink messages.

The packet info is sent to the [sprbus](#) where clients subscribing to the *nft*-prefix gets notified. It's easier to talk to sprbus for packet inspection but also [available by packet\\_logs](#).

*Example:* The [db service](#) can enable storage of logs by adding the topic to `configs/db/config.json` under the `SaveEvents` key.

See also: [sprbus](#)

## The Groups

### Group 0

```
chain INPUT { ...
$(if [ "$WANIF" ]; then echo "iifname $WANIF log prefix \"wan:in \" group 0";
fi)
$(if [ "$WANIF" ]; then echo "iifname ne $WANIF log prefix \"lan:in \" group
0"; else echo "log prefix \"lan:in \" group 0"; fi)

}

chain FORWARD { ...

$(if [ "$WANIF" ]; then echo "oifname $WANIF log prefix \"wan:out \" group 0";
fi)
$(if [ "$WANIF" ]; then echo "oifname ne $WANIF log prefix \"lan:out \" group
0"; else echo "log prefix \"lan:out \" group 0"; fi)

}
```

#### wan:in

Input from the upstream interface (\$WANIF)

## **lan:in**

Input from all other devices (wireguard, wireless clients, wired devices)

## **wan:out**

Packets forwarded upstream to the internet over the \$WANIF interface

## **lan:out**

Packets forwarded to any other non-upstream interface

## **Group 1 - Dropped packets**

```
log prefix "drop:private" group 1
counter log prefix "drop:forward" group 1
counter log prefix "drop:input" group 1
log prefix "drop:mac" group 1
```

### **drop:private**

This prefix marks packets that were dropped because they were headed upstream to a private network address, but blocked from doing so because they were not in a permitted group

### **drop:forward**

Packets that were dropped during forward

### **drop:input**

Input packets into SPR that were dropped

### **drop:mac**

Packets that were dropped because of strict MAC filtering in the INPUT chain in the FORWARD chain.

[⇒ View the code](#)

# **Inspecting packets**

Retrieve JSON packet logs on SPR with:

```
docker exec -it superpacket_logs /stream-json-logs > log.json
```

will log packets for 20 seconds, can specify with `-timeout` flag.

Packets can be inspected with `jq`, see following section for fields. Raw data is available and base64 encoded.

## Script to inspect traffic

```
> .
```

Example script with `gum` – a tool for glamorous shell scripts.

Install dependencies:

```
apt install jq
go install github.com/charmbracelet/gum@latest
```

```
#!/bin/sh
# packet_logs json stream build if not running on spr:
# cd packet_logs/stream-json-logs && go build -o stream
```

```

# ./stream > log.json
F="$PWD/log.json"
F_CSV="${F}.csv"
BIN_STREAM="docker exec -it superpacket_logs /stream-json-logs"

log_stream() {
    gum spin -s moon --title "Logging traffic... 20s" -- sh -c "$BIN_STREAM >
$F"
}

table() {
F_UDP='[.Timestamp,"UDP",.Prefix,
([.IP.SrcIP,.UDP.SrcPort|tostring]|join(":")),
([.IP.DstIP,.UDP.DstPort|tostring]|join(":"))]'

F_TCP='[.Timestamp,"TCP",.Prefix,
([.IP.SrcIP,.TCP.SrcPort|tostring]|join(":")),
([.IP.DstIP,.TCP.DstPort|tostring]|join(":"))]'

cat "$F" \
| jq -r -c "if .UDP!=null then $F_UDP elif .TCP!=null then $F_TCP else
empty end | @csv" \
| sed 's//"/g' > "$F_CSV"
}

IS_TABLE=0

if [ $IS_TABLE -eq 1 ]; then
    SEL_ROW=$(gum table -c "ts, proto, prefix, src, dst" -w 30,8,10,20,20 --
height 20 -f "$F_CSV")
else
    SEL_ROW=$(cat "$F_CSV" \
| sed 's/,/ /g' \
| gum filter --no-fuzzy --indicator.foreground="99" --
match.foreground="99" --height 20)
fi

SEL_TS=$(echo $SEL_ROW|sed 's/,/\t/g'|awk '{print $1}')

if [ -z "$SEL_TS" ]; then
    exit
fi

SEL_JSON=$(cat "$F" | jq "select(.Timestamp == \"$SEL_TS\")")

echo "$SEL_JSON" | gum pager
}

```

```
gum style --foreground 99 --border double --border-foreground 99 --padding "0  
4" --margin 1 "SPR packet logs cli"  
gum input --placeholder="Press ENTER to start"  
  
if [ ! -f "$F" ]; then  
    log_stream  
fi  
  
while true; do table; done
```

save the script in packetlogs.sh and run it:

```
./packetlogs.sh
```

# OUI & MAC Lookups

This plugin provides api endpoints to query for IP address ASN & MAC address OUI information

For ASN lookups its using this: <https://iptoasn.com/data/ip2asn-v4.tsv.gz>

And the OUI vendor list from Wireshark: <https://gitlab.com/wireshark/wireshark/-/raw/master/manuf>

Files are downloaded on startup of the container [⇒ View the code](#)

# PPP

SPR can be run as the point of entry for home internet.

PPPoE is supported with the standard ubuntu ppp daemon. To configure it, see the Uplink view, or the [uplink/ppp API endpoint documentation](#)

This service is off by default and dynamically started when enabled.

 [View the code](#)

# SPRbus

version v0.1.7 go report A+

SPRbus is the event bus used in SPR to send and subscribe to events.

[View the code](#)

## Usage

- [db](#) is listening for events and store these in a database
- [packet logs](#) send events when netfilter rules are triggered
- [api](#) for notifications
- [api](#) publish event for access log
- [dns](#) send events on domain lookup or block

## Available events

- dns:block:event
- dns:override:event
- dns:serve:192.168.2.2
- log:api
- log:www:access
- nft:lan:in
- nft:lan:out
- nft:wan:in
- nft:wan:out
- nft:drop:private
- nft:drop:forward
- nft:drop:input
- nft:drop:mac
- www:auth:user:success

# Example Events

- *wifi:auth:success* is published when a device connects
- *device:save* is published when a device is updated

[View the API documentation for sprbus here](#)

By default *log:api*, *dns:block:event*, *dns:override:event*, *dns:serve:* events are stored in the database. The log items can be viewed under *System -> Logs*. If you want to store more events you can add them in the web ui under *System -> System Info -> Database*.

## Command line tools

```
> .
```

See [cmd/main.go](#)

```
cd cmd/; make  
./sprbus --help
```

## **remote**

```
export TOKEN="SPR-API-TOKEN"  
./sprbus --addr 192.168.2.1
```

## **local**

```
./sprbus
```

## **example topics**

```
#www and api logs  
./sprbus -t log  
# device and wifi events  
./sprbus -t device,wifi  
#network traffic  
./sprbus -t nft
```

## **network traffic in json, no timeout and pipe to jq**

```
./sprbus -t nft -j --timeout 0 | jq .
```

## **publish test event**

```
./sprbus -t test:event -p '{"msg": "testevent1234"}'
```

```
> c
```

example showing how to publish events.

## Development

See [example/main.go](#)

using default sprbus:

```
//publish json string
sprbus.PublishString("wifi:station:event", "{\"json\": \"data\"}")

//publish object
sprbus.Publish("www:auth:user:fail", map[string]string{"username": username})

//subscribe
go sprbus.HandleEvent("wifi", func (topic string, json string) {
    fmt.Println("wifi event", topic, json)
})
```

### Custom unix socket server and client

See [example/main.go](#) for code to setup a custom unix socket server and client

# Superd

## About

The superd service can restart SPR services and download PLUS extesions.

The API communicates with SPR over a unix socket, located at `state/plugins/superd/socket` and documented [here](#).

⇒ [View the code](#)

# VPN

## About

VPN is powered by Wireguard (© Jason A. Donenfeld).

The service is listening on udp port 51280.

# Wifi Uplink

This is a built-in extension that is dynamically started when a wireless uplink is configured under the [Uplink](#) page.

The plugin will run `wpa_supplicant` to connect to the configured access point.

The management of the `/configs/wifi_uplink/wpa.json` configuration for this plugin is handled under the core API service.

[View the code](#)

# Wifid

The wifi service runs [Hostap](#) with some patches for better WPA3 support that will be submitted upstream at some point in the future.

The service supports multiple radio interfaces, and will run parallel hostap daemons to manage them.

When a station successfully authenticates, fails to, or disconnects: an [action script](#) is ran to inform the API as well as helper program. This communication happens over a dedicated unix socket [documented here](#).

Once a station authenticates successfully, a [helper program](#) is kicked off to grant permission to DHCP from the client's approved MAC address. An [XDP filter](#) is employed to block stations from making DHCP requests for arbitrary MAC addresses.

⇒ [View the code](#)

# API Overview

[View the OpenAPI Documentation here](#)

The API service exists for communication between service containers and to support external requests from the web front end or CLI tools.

The internal APIs run over unix sockets and rely on filesystem namespaces for authentication. Currently there are a few internal APIs exposed to containers over unix sockets:

- Support for [wifid service accepting stations](#)
- Setting up dynamic firewall rules for [DHCP clients](#)
- Support for [wireguard peers](#)

⇒ [View the code](#)

## API Plugins

A list of curated plugins is [available here](#)

API plugins let other docker services expose web APIs with unix sockets that are relayed with a reverse proxy.

To add a new plugin update `configs/base/api.json` or use the UI to enable it.

If the plugin should be started by SPR, then make sure that the compose path is added to the whitelist.

To do so, update `configs/base/custom_compose_paths.json`

Example dyndns plugin

```
{  
  "Plugins": [  
    {  
      "Name": "dyndns extension",  
      "URI": "dyndns",  
      "UnixPath": "/state/plugins/dyndns/dyndns_plugin",  
      "Enabled": true,  
    }  
  ]  
}
```

```
        "ComposeFilePath": "dyndns/docker-compose.yml"
    }
]
}
```

You can also talk to the spr api directly to list/add/delete plugins - see the [api documentation for plugins](#).

Or in the UI navigate to *System -> Plugins*. Note, SPR for now will not accept arbitrary compose file paths. Instead, update `configs/base/custom_compose_paths.json` with the relative path to the plugin.

## Query the http API

It is recommended to create a token to talk to the API (more info about authentication [here](#)). In the Admin UI, click **Add Auth Token** under *System -> Auth*. Use this token for authorization:

```
export TOKEN="paste-token-here"
curl -s -H "Authorization: Bearer $TOKEN" 192.168.2.1/plugins
```

## Query a plugin API

If developing a plugin on the same host as spr you can talk to a specific plugins unix socket.

**Example** - talk to db service to get stats:

```
pwd
/home/spr/super
sudo curl -s --unix-socket ./state/plugins/db/socket 0/stats
```

## SPR Event Bus

[sprbus](#) is an event bus for publishing and subscribing to events. The server routing messages between services is setup in the [api service](#).

Clients connect to `/state/api/eventbus.sock`, either subscribing to a topic or publishing events for other clients to handle.

The sprbus command line tool can be used to inspect live events:

```
> c
```

See the [github repository](#) for more examples.

Read more about [SPRbus](#)

# Authentication

## Configuration

On the device, the following files contain credentials for access.

- `config/auth/auth_users.json` is a JSON file with a dictionary for username, password pairs
- `config/auth/auth_tokens.json` is a JSON file with a list of authentication tokens

## Basic Authentication

The API supports Basic Authentication based on the `auth_users` file, using the Basic Schema.

To implement this, a client should include an "Authorization: Basic" header with the credentials formatted as `base64(username:password)`. For a javascript example, see the frontend's [Api.js](#).

## Bearer Tokens

The API also supports Bearer Tokens, based on the `auth_tokens` file.

To implement this, a client should include an "Authorization: Bearer" token. See [Swagger's page](#) for more examples.

## WebAuthN

WebAuthN will support passwordless authentication. Support is a work in progress and is currently disabled. The following endpoints will be available.

GET /register/?username=username

POST /register/

GET /login/?username=username

POST /login/

# SSL Support

The API service looks for a certificate at `configs/base/www-api.crt` and the corresponding key `configs/base/www-api.key`. If they exist the api will listen on port 443 for TLS-encrypted requests.

Use the provided script to generate a self-signed certificate.

## Enable SSL web api

### 1. Generate a certificate

```
D= ./ ./api/scripts/generate-certificate.sh
```

### 2. Restart api container

```
docker-compose restart api # (or restart api-virt if using the virtualized network variant)
```

### 3. Verify the SSL service is running

```
curl -k https://127.0.0.1
```

### 4. Open port 443 for WAN access (optional)

**NOTE** This is not recommended if running SPR on the internet.

```
echo "UPSTREAM_SERVICES_ENABLE=1" >> configs/base/config.sh
docker-compose restart api # (or restart api-virt if using the virtualized network variant)
```

# Frequently Asked Questions

## ▼ What does SPR stand for?

SPR stands for Secure Programmable Router

## ▼ What is SPR?

SPR is Linux software for running a hardened wifi network that also makes it easy to add and manage devices. Devices are isolated into individual VLANs and then connected securely into the rest of the network.

## ▼ How is SPR different from OpenWRT?

- SPR is built to support the leading edge of Linux's networking features, while applying best in class security practices and modern coding standards. The focus is on making usable micro-segmentation with zero-trust networking possible, which is not simple to configure with OpenWRT.
- OpenWRT powers many of the proprietary consumer wifi routers on the market today and was originally intended as a replacement for proprietary embedded firmware. Some routers allow replacing the vendor's OpenWRT with open source builds of the OpenWRT firmware.
- SPR is designed to run in docker containers on systems with a bit more memory and storage, opening up many possibilities like hosting voice assistants, network fuzzers, and media servers. However, it likely won't work as a replacement for embedded router firmware.
- OpenWRT tries to optimize builds for embedded devices without much storage space or memory. It supports a tremendous range of devices and hardware, although older devices are losing support due to storage and memory limitations.
- SPR is built mostly with golang and a react frontend. Where possible, memory-safe languages with good security track records are preferred over native code such as C or code that is difficult to write securely, like PHP.

- OpenWRT is built mostly with C, Shell Scripts, and Lua/LUCI for the web front end.

▼ How do I update SPR?

You do not need to reflash your system. From the UI: go to *System Info* -> *SPR Release*. You can run **Check** to see if there's an update and hit **Update**.

From the CLI: Go to the SPR directory (`/home/spr/super/`), Run:

```
cd /home/spr/super  
git pull  
docker-compose pull  
docker-compose up -d
```

▼ How is SPR different from an ordinary network and what is a supernetwork?

SPR places each WiFi device into its own VLAN and subnet, using per-device passphrases. When combined together, the devices form a supernetwork. The VLAN isolation creates strong hardening features. Devices can not sniff each other's traffic, they can not perform ARP/MAC spoofing, and they can not communicate with each other without being granted access while they may still be able to access the internet.

▼ Is there any telemetry or statistics sent from SPR to Supernetworks or Third Parties?

No. This can be confirmed by viewing the code in the github repo or if you prefer, look at the network traffic from SPR

▼ What are groups vs tags?

Groups in SPR refer to network access and which devices should be able to communicate with one another. There are three special, built-in groups:

- **wan** : the device can communicate with the internet
- **dns** : the device can make DNS queries
- **lan** : the device can communicate with all other devices

When custom groups are created, all of the devices in that group can communicate with each other. For example, a custom **gaming** group could be created for devices that need to access one another, or a **media** group for streaming to TVs.

Tags are an additional mechanism for organizing devices and applying special features to them. Today there is one built-in tag, ***lan\_upstream***.

Custom tags can be used with DNS block lists, to apply blocklists to only select devices. Place the tag with both the list and the devices it should apply to. For example, a user can set a **focus** tag on specific devices, and create a dns block list under the **focus** tag.

Custom tags can be used with PLUS's Programmable Firewall plugin. An advanced feature is to use PFW to apply tags on a schedule. The **focus** tag could be applied from 9AM-5pm for example.

▼ How does the "lan\_upstream" tag work?

By default, SPR prevents devices from accessing LAN addresses upstream of SPR. This is useful to restrict access to other subnets that SPR may be on. If a user does want a device to be able to access LAN networks upstream of SPR, the device should have the ***lan\_upstream*** tag applied to grant access.

▼ Does SPR work with iCloud Keychain Sync?

Yes - however you need to set the same password on each of the iOS devices that are in the same keychain. Without this, they would sync the wrong password to each other and lock each other out of the network.

▼ Can I try SPR without the hardware?

Yes. You can run SPR in docker, either locally or on a cloud instance. NOTE: wifi-features will not be available on MacOS & Windows

▼ Can I run SPR without a wifi-card?

Yes. SPR runs great as a VPN service as well. See: [Virtual Setup Guide](#) to setup SPR with VPN and DNS support

▼ Why does SPR need AP/VLAN mode?

AP/VLAN is used to place each wifi station into a VLAN with its own virtual interface. Not all drivers may support this feature.

It is possible to use SPR for wifi without this but it is not recommended. To do so run hostapd with VLANs disabled (set `per_sta_vif=0`), and under `base/config/config.sh` set LANIF to your wifi interface (i.e. `wlan1`). With this configuration, MAC spoofing and packet injection across devices is possible, as with an ordinary wifi network.

▼ Does SPR run on x86 / x86-64?

Yes. We publish docker containers for both arm64 and x86-64. See: [Building the project](#) to build from source.

▼ Does SPR support extensions?

SPR supports plugins running as docker containers. See: [Extensions Documentation](#)

▼ Which wifi drivers work with SPR?

The MT76 family of drivers has been best tested. Mt76x2u, mt7915, mt7915e have seen the most use.

Note that iwlwifi, from intel, disables AP mode, and so the popular ax210/ax200 cards are not today compatible with hostapd. It is possible to run some AP with wpa\_supplicant but important features are missing. Contributions to get these working with SPR are welcome but it may not be feasible.

Recommended features for SPR:

- WPA3 Support
- AP/VLAN mode

If you have SPR installed you can see if your card is supported under *Wifi -> Interfaces -> SPR compatibility*

▼ Does SPR support Wifi 6?

Yes.

▼ Does SPR support Mesh Networking?

Mesh is currently a [PLUS](#) plugin. It supports mesh nodes with wired backhaul. Mesh networking (802.11s) is not currently configurable for a wireless backhaul from the API, but can be applied by configuring wpa\_supplicant manually.

▼ What is PLUS?

Plus offers extended features & lets you back the development of SPR

[Read more about SPR PLUS](#)

▼ What is PFW?

PFW is a **Programmable Firewall** available to [PLUS](#) users. It allows scheduling and event based firewall rules. It supports advanced features such as per-device rerouting to site-to-site VPNs and docker containers, and scheduled focus mode.

▼ Think I've found a bug, where can I report it?

---

The best place to file an issue is our [Github Issue Tracker](#)

▼ Do you have a mobile app for the admin interface?

---

The iOS app is available on [App Store](#).

▼ Can I use the mobile app for Virtual SPR?

---

Yes!

If you have setup SPR using the [Virtual Setup Guide](#) you can login to SPR (192.168.2.1) when connected to the VPN.

▼ Can I run a dev branch of this project?

---

Yes, See *System -> Releases*

▼ What Operating System is SPR based on?

---

Ubuntu Linux 23.04. It can run on a variety of Linux hosts though.

▼ Do you have a custom Linux kernel?

---

The Clearfog release is running a [custom build](#). The Raspberry Pi image is using the default kernel for ubuntu.

▼ Where can I learn more about Wifi security?

Check out our turtles [ctf challenges](#) & [write-ups](#)

# Troubleshooting

## ▼ Device can't connect using WPA3

Some IOT devices may have problems with WPA3 support, or even incorrectly detect WPA3 as an Open Network. Try adding the device with WPA2 and connect again. Some IOT may not even support WPA2, and require WPA1.

## ▼ How can I set a custom subnet for my SPR network?

*System -> Supernetworks*

## ▼ I'm running other containers on my SPR & they have no internet connection

If SPR is not running, make sure to setup the firewall rules. Make sure WANIF is configured with your upstream interface in config/config.sh and try running the base/docker\_nftables\_setup.sh script. SPR disabled docker's reliance on IPtables and so custom NFTable scripts are required for docker containers to reach the internet.

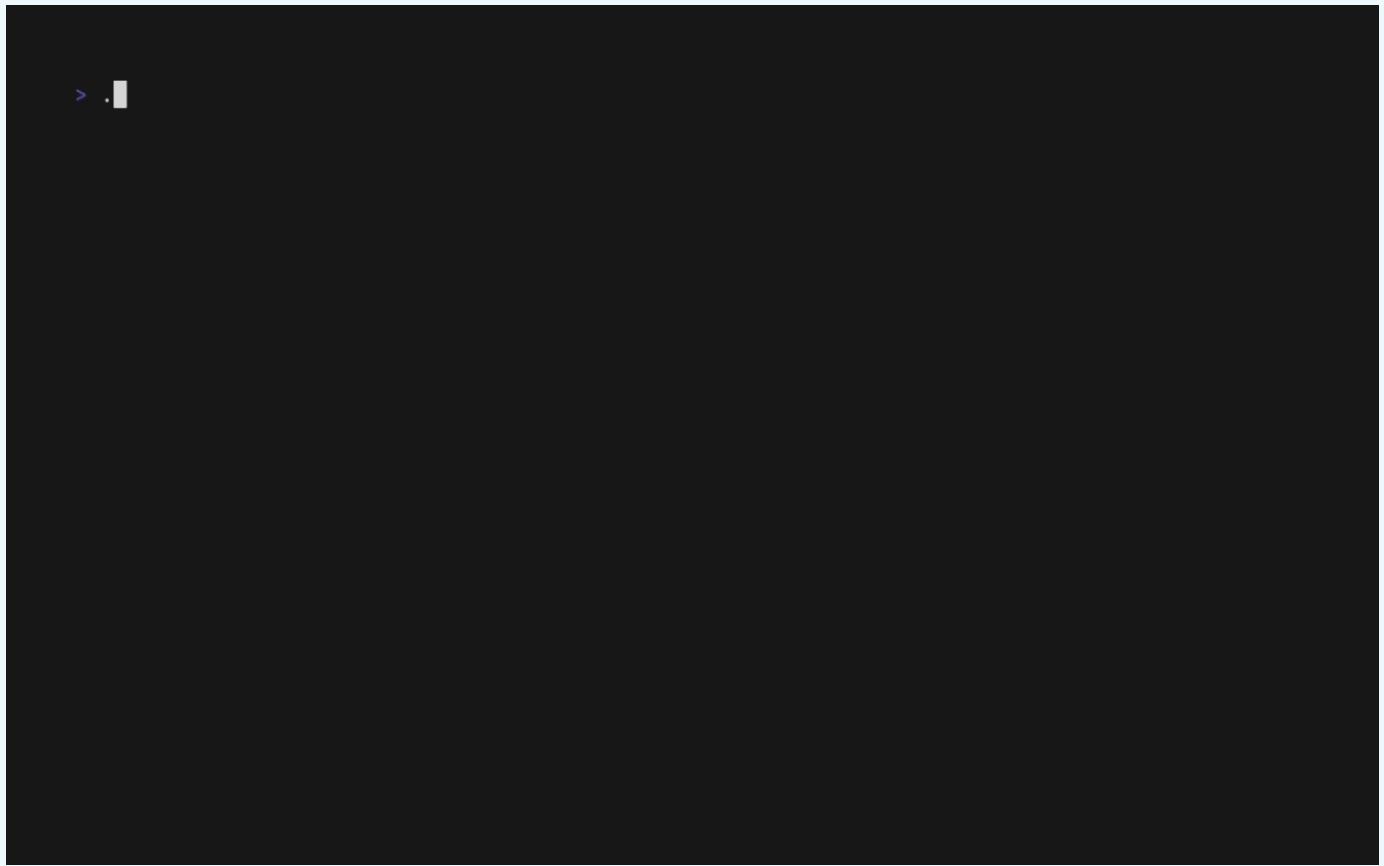
## ▼ How can I debug dropped packets?

1. Check the events page for `nft:drop:` events. If they are not recorded, enable them first under event settings. Dropped packets may be under `nft:drop:mac`, `nft:drop:forward`, `nft:drop:input`, or `nft:drop:private`

2. Retrieve JSON packet logs with

```
docker exec -it superpacket_logs /stream-json-logs
```

3. Or build the [sprbus tool](#) and connect with an API token. You can generate an API token in the UI under "Auth-> Add Auth Token"



# Security

## Contact Information

Email spr-security [at] supernetworks.org or reach out on the [discord chat](#)

## Security Goals

**Router is secure against compromises via the web services, remote uplink, or local network attack surfaces**

**Practical to use strong passwords for wifi devices**

**One compromised device should not be able to impersonate other devices on the network or intercept their network traffic**

**Devices can only communicate to systems they are explicitly allowed to. No spoofing.**

---

## Key Security Features

### Multi-PSK & VLANs

SPR places each WiFi device into its own VLAN. The device password and MAC address combination is used to authenticate into the DHCP assigned VLAN. The device does not need to be aware it is in a VLAN, except that DHCP has provided a /30 "tiny" network, and communications to other LAN devices should be routed over the AP.

There is no limit to the number of VLANs and a user does not have to assign devices to VLANs.

### No MAC spoofing and other layer 2 network pivoting

Firewall rules enforce the MAC address for the authenticated device to block MAC spoofing. Further OS configuration blocks ARP spoofing from interfaces on a VLAN. The packet forwarding to other devices

is default deny. If a device is part of the all devices LAN group, or is in a group with other devices then traffic will be allowed.

GTK are unique per VLAN so devices can't bypass the router to communicate. TDLS is disabled.

These rules are also enforced with mesh networking, supported today in SPR PLUS over wired backhaul.

## Multicast Limitation

Currently the multicast proxy will relay multicast traffic to all devices. A further hardening step is to enforce routing policy for multicast traffic as well.

## Upstream LAN Traffic Blocked By Default

Typically, users of SPR will plug the SPR into their existing network which will be called an upstream LAN. By default, SPR will block traffic to upstream Private LANs [rfc1918](#) for devices, unless the `lan_upstream` tag is enabled. This prevents SPR devices from accessing upstream private addresses.

## WPA3 Support

WPA3 uses the Simultaneous Authentication of Equals (SAE) protocol for authentication. The Key Exchange can not be sniffed and cracked as with WPA2 (PBKDF2 based) because it's a zero knowledge-proof of the password.

WPA3 Also provides for Management Frame Protection (MFP) 802.1w which is optional for WPA2 but mandatory in WPA3.

## Practical Limitations of WPA3

### iOS Device QR-Code WPA2 Downgrade

iOS has a long standing flaw where networks with WPA3 that are scanned with a QR Code are later saved as WPA2. As a result, SPR supports both WPA2 and WPA3 for devices.

**Many devices don't support WPA3 yet, some still require WPA1 even**

Since not all devices support WPA3, a bssid, SPR runs MFP with mixed mode (ieee80211w=1).

## Network Visibility

SPR provides for DNS, traffic monitoring capabilities as well as authentication logs for the APIs.

---

# Threat Actors

### **Remote Internet Attacker**

Anyone on the internet that can send packets to the WAN/Uplink interface

### **Man In The Middle / Malicious ISP**

An attacker with a man in the middle position on the uplink

### **Supply Chain Attacker**

An attacker looking to insert code into the SPR project to compromise routers

### **Physical Proximity Attacker (Evil Neighbor)**

An attacker with physical proximity to WiFi

### **Inside Perimeter Attacker (Evil Guest)**

An attacker with physical access

### **Compromised Device Attacker (Implant)**

An attacker operating from a compromised device, authenticated on the network

# Threat Vectors

## Network Flaws

### **Weak Passphrase / Password Reuse**

**ARP Spoofing**

**MAC Spoofing**

**DHCP MAC Spoofing**

**VLAN Hopping**

**Insecure Private Requests from Web Browsers**

## **Software Implementation Flaws**

**Memory corruption**

**Command Injection**

**XSS, CSRF**

**DNS Cache Poisoning**

**Response Splitting Attacks**

## **802.11 Flaws**

**Cryptographic Vulnerabilities**

**Password Cracking**

**Frag Attacks**

**MITM**

**AP Isolation Bypass**

**Packet in Packet Attacks**

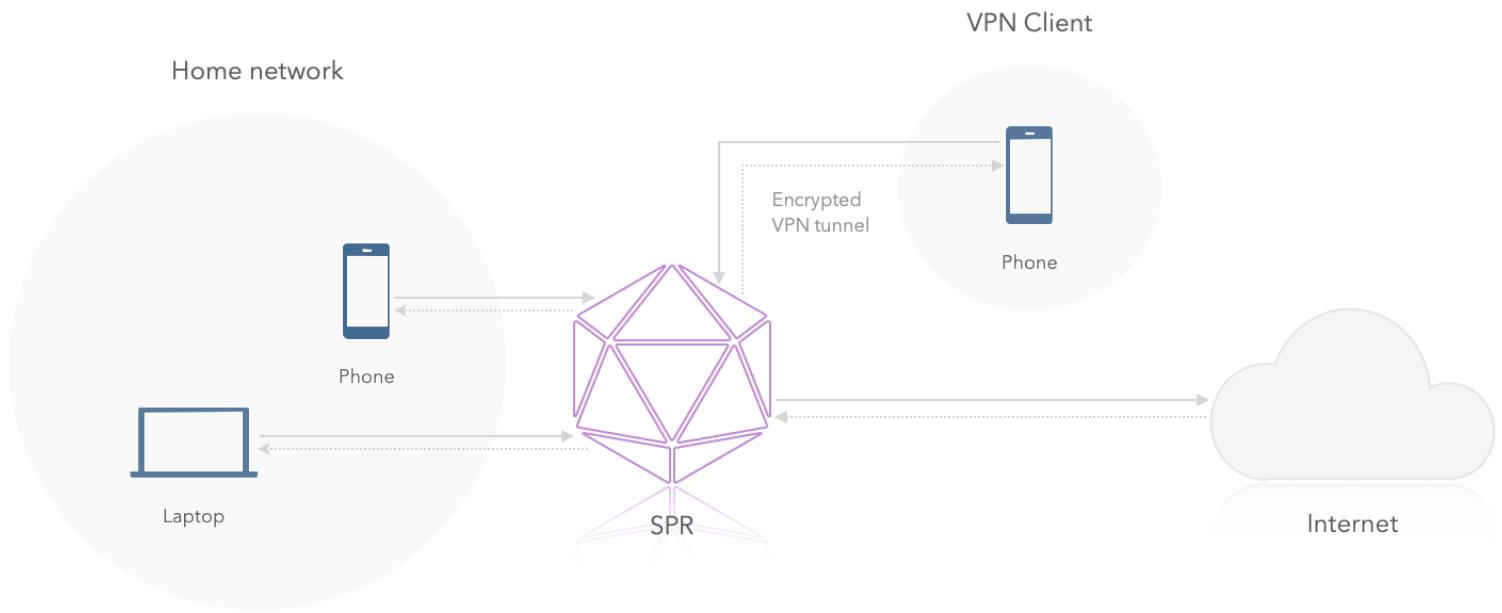
# VPN Overview

SPR support multiple modes of VPN connections:

- connect *from* the internet to your home network
- connect *to* another network using a vpn tunnel

## VPN Client

Example: Connect to your home network when traveling.



This is useful if you always want to have the same setup (adblock, firewall rules) from your phone or laptop, or want to be able to access your home network when traveling.

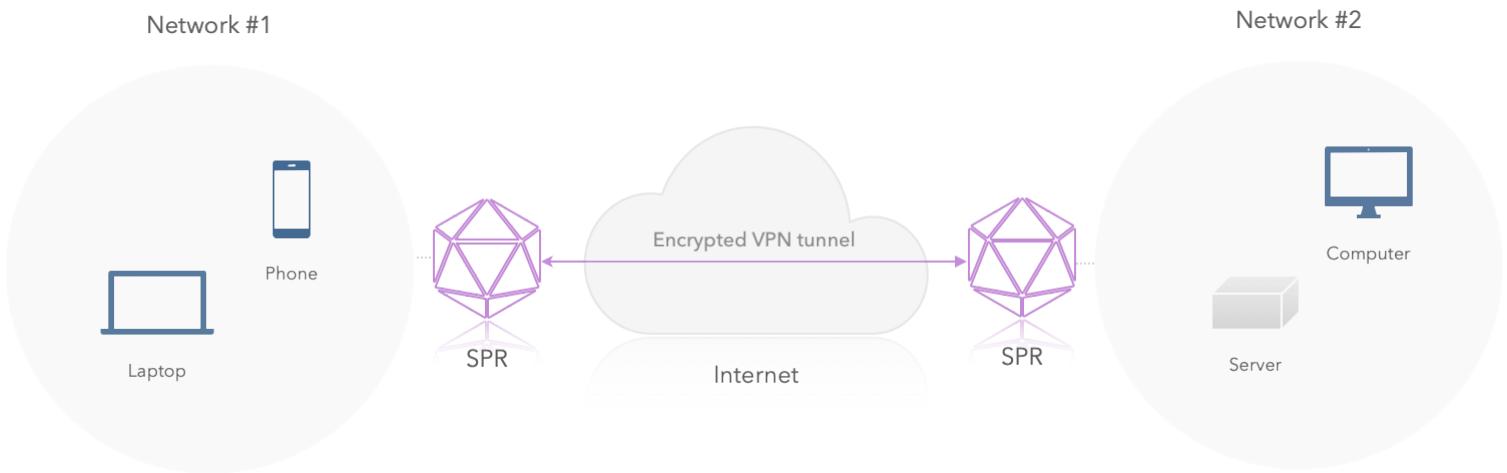
## Add a peer

See the [Virtual Setup Guide](#) on how to add a vpn peer.

**Note:** If you link the created pubkey to an already configured device it will use the same IP address.

## Site-to-Site VPN

A site-to-site VPN can be used to link two networks together. Example: another spr instance, cloud network.



## Setup site-to-site VPN

See the [PLUS Site VPN Configuration](#)

## PFW support

More advanced rules can be setup using the "*Forward to Site VPN*" action in PFW. Example:

- Only for specific devices
- Run at a specific time during the day, only on weekdays