# How to use Raspberry Pi as a Wireless Router with Firewall?

I wanted to build a router firewall on Raspberry Pi for a long time. I first tested Pfsense and OpenWRT with no success, and on a fresh Raspberry Pi OS I was missing information. But now it's ok, I finally found how to do it, and I'll share this with you

**The Raspberry Pi only have one Ethernet socket, so it's not possible to create a firewall with two RJ45 interfaces.**
**But there is a Wi-Fi interface that can be used for one side (LAN for example).**
**One way to build a firewall is to use the hostapd and iptables services.**

And I'll show you how. It's a big topic with many services and network notions to understand.
I'll start with the theory and then explain how to install each software.

If you are looking to quickly progress on Raspberry Pi, you can check out my e-book here. It's a 30-day challenge, where you learn one new thing every day until

you become a Raspberry Pi expert. The first third of the book teaches you the basics, but the following chapters include projects you can try on your own.

## Router

**A router is a network device that connects two networks together.**
If you have two Ethernet ports on a computer, with different networks on each, your computer can act as a router.

In this schema, we have two different networks, connected with a router: 1.0 and 2.0.
If the router is well configured, it allows A and B to see each other, while on a different network.

The Raspberry Pi have only one Ethernet card, but we can use the Wi-Fi card to create a second network.
**So, the router part in this tutorial will allow us to connect the Wi-Fi network to the Ethernet network**.

## Firewall

**A firewall is a software. It allows us to add security policies in the router.**
For instance, in the previous example, we can configure that A can ping B, but not

access the HTTP server on B.

I'll use a software called "iptables" for this, but you can use any other firewall software if you prefer.

## Our goal

If you use your Raspberry Pi at home, you probably don't need to connect two networks.

But **my goal is to create a new wireless access point with a firewall and other cool software to monitor the network and filter some kind of traffic**.

Here is my current network:

And I want to turn it like this:

So, here are the steps you need to follow to do the same:

- Install your Raspberry Pi on the network
- Enable Wi-Fi access point with a different network subnet
- Create a bridge between the two networks
- Create firewall rules
- Install other cool software

I'll explain you each step in detail.

Let's move to the first step of this process 🙂

## Install your Raspberry Pi

The first thing to do is to install your Raspberry Pi on the network:

- **Install Raspberry Pi OS** by following this tutorial
  You don't need the Desktop version, except if you want to use the Raspberry Pi for other things too
- **Plug the Raspberry Pi on the network** with an RJ45 cable
  We'll use the Wi-Fi later, so you need to let him available
- A static IP address is not mandatory but it can help
  You can check the end of this article to know how to configure it
- Once done, **update your system** by doing:

```
sudo apt update
sudo apt upgrade
sudo reboot
```

- **Enable SSH** in raspi-config > Interfacing options

```
sudo raspi-config
```

You'll use it in this tutorial, to copy/paste command from this page.
You can find more details on how to use SSH in this tutorial if needed.

That's it, you have done the preparation step, we can start with the router installation.

Are you a bit lost in the Linux command line? Check this article first, for the most important commands to remember, and a free downloadable cheat sheet so you can have the commands at your fingertips.

# Install a wireless router

On Stretch (Debian 9), a script was available to do everything automatically, but it hadn't been updated and doesn't work anymore.
So, we'll do it manually, it' not so complex.

## Configure the Wi-Fi country

If you are using a fresh new Raspberry Pi OS, you need to set a Wi-Fi country first. The Wi-Fi is disabled until that.

- Open raspi-config
  ```
  sudo raspi-config
  ```
- Go to Localisation Options > Change WLAN country
- Select your country in the list

- Confirm and exit

## Install the services

We'll mainly use two new services on our router:

- **Hostapd**: to create the wireless access point
- **DNSmasq**: to forward the DNS requests to another DNS server

Start by installing the required packages:

```
sudo apt install hostapd dnsmasq
```

That's it, we can now move to the configuration part.

## Configure Hostapd

In the Hostapd configuration file, we will add the settings for our new wireless network:

- Open the configuration file:
  ```
  sudo nano /etc/hostapd/hostapd.conf
  ```
- Paste these lines (the file is probably empty):
  ```
  interface=wlan0
  driver=nl80211
  ssid=RaspberryTips-Wifi
  ```

```
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=<YOURPASSWORD>
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Don’t forget to set a passphrase, and you can edit the line you want (for example to use another channel, security level or SSID)

- Save & exit (CTRL+O, CTRL+X)

Hostapd won’t start automatically on boot, there are two changes to do to enable this:

- Edit the default configuration file:
  `sudo nano /etc/default/hostapd`
- Add this line at the end:
  `DAEMON_CONF="/etc/hostapd/hostapd.conf"`
- Save & Exit
- Then, enable the service with:
  `sudo systemctl unmask hostapd`
  `sudo systemctl enable hostapd`

## Configure DNSmasq

The next step is to configure DNSmasq:

- Open the configuration file:
  sudo nano /etc/dnsmasq.conf
- Paste these lines at the end:
  interface=wlan0
  bind-dynamic
  domain-needed
  bogus-priv
  dhcp-range=192.168.42.100,192.168.42.200,255.255.255.0,12h
  Nothing to change here, except if you want to change the subnet.
- Save & exit (CTRL+O, CTRL+X)

## Configure the DHCP server

The last configuration file to change is the DHCP server, set it on the same subnet:

- Open the configuration file:
  sudo nano /etc/dhcpcd.conf
- Scroll to the end of the file and paste these lines:
  nohook wpa_supplicant
  interface wlan0
  static ip_address=192.168.42.10/24
  static routers=192.168.42.1
- Save & exit

We are almost ready for the first test.

## Enable IP forwarding

If you have several network cards, the default behavior on Linux is to isolate them.

In our case, we want to enable the communication between the LAN and the Wi-Fi.

So, we need to change this:

- Open this file:
  `sudo nano /etc/sysctl.conf`
- Find this line (first page):
  `#net.ipv4.ip_forward=1`
- And uncomment it:
  `net.ipv4.ip_forward=1`
- Save & exit

You can now reboot for a first try:

```
sudo reboot
```

*Note: I had to do this two times on my two tests because I was not getting an IP address on the first reboot. No idea why…*

Once this is complete you should be able to see your Raspberry Pi access point in the networks list

In your Wi-Fi networks list, you should see something like this:

**You can connect to it and check that everything is working as expected**.

You should get an IP in the 192.168.42.0/24 subnet, the script created this network for you.

You'll not get any Internet connection for now, as we need to configure the firewall to allow the Internet traffic.

# Understand the firewall concepts

I'll start with an introduction on the theory about firewall configuration.

If you are already fluent with this, you can move on to the next section.

## Introduction

**The role of a firewall is to block or allow access from a specific IP to another.**

**And often we also use a port to set the exact permission**.

Ex: We deny port 22 to everyone, except computer A that can access computer B with port 22.

## Black or White

In a firewall configuration, you have the choice between two default rules:

- **Blacklist**: Allow all except …
- **Whitelist**: Deny all except …

Depending on what you want to do with your Raspberry Pi router, it's your choice to take the one you want.
The first option is probably ok if you are using it at home. You may want to block only certain things like the torrent protocol or specific IPs address.
But at work it's rather the second. The good practice is to block everything except what is allowed.

## In, Out and Forward

This one is easier.
In your firewall, you can create rules in three directions:

- **Input**: Network packets coming into the firewall
- **Output**: Packets going out from the firewall
- **Forward**: Packets going through the firewall

On a hosted web server, you can block anything in input except HTTP and HTTPS.
But in output it's not a big deal what your server is doing on the Internet.

**In our case, we'll mainly use FORWARD rules only as there is nothing on the Raspberry Pi.**

There is no need to protect it more than that.

## Configure the firewall service

If you want, you can add a firewall in your router to filter the traffic.

Maybe at home it's not mandatory, but for a company or a public place you need this.

## IPTables

There are several firewall packages available on Raspberry Pi OS: iptables or ufw for example.

There is also OpenWRT, a Raspberry Pi compatible distribution, to create a router firewall.

In this post, I'll use iptables, the most used.

It's already installed on your Raspberry Pi, so there's nothing else to do.

## See the current configuration

Before adding rules, you need to check the current configuration.

To do this, use the command:

```
sudo iptables -L
```

You should get something like that:

We find the input, output and forward parts I talked about previously.

**For each we see that the default policy is ACCEPT, so everything allowed except what we add (blacklist mode)**.

You can use this command later to check if the new rules you add correspond to what you want.

## Enable Internet forwarding

Before going into more details, we'll just **add some basic rules to allow the Internet traffic:**

- **Type these 3 commands:**
  sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
  sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
  sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
- You don't need to understand them right now, I will explain everything in the next parts.
  But once done, you should have an Internet access working on your access point.
- **Here are the rules you have now, with iptables -L**

## Add a FORWARD rule

**We'll use the iptables command to add new rules in the firewall.**

Every network is different, so every firewall rules table is different.

I'll start by an example, and then I'll give you the whole syntax to add specific rules in your environment.

- **Start by resetting the iptables configuration**
  sudo iptables -F

The order is important in the rules list, and as the router already accept everything in the forward section, we can't block specific connections.

- **Add a DROP rule**

```
sudo iptables -A FORWARD -p tcp --dport 80 -j DROP
```

This command specifies that:

– we add a new rule (**-A**)

– in the forward section (**FORWARD**)

– for the tcp protocol (**-p tcp**)

– for the HTTP port (**–dport 80**)

– and the action is to DROP everything (timeout connection)

- **Use the 3 commands I gave you in the previous part to allow the other Internet traffic**
- After that, you should not be able to access a website like this one for example (but sites in HTTPS still work)

It's ok, your first rule is operational.

You can use iptables -F to remove all rules and start again.

Or you can **use the same command with the -D operator instead of -A**.

```
sudo iptables -D FORWARD -p tcp --dport 80 -j DROP
```

This command allows you to delete a specific rule and not all like with the -F.

## Iptables command syntax

As you should already understand, you can now use the same command template to create the firewall rules you need.

The command template is:

```
iptables -<operation> <direction> -p <protocol> --dport <port> -j <action>
```

- **operation**:
  - -F: flush, remove all rules, it requires no other parameters
  - -A: append, add a new rule
  - -D: delete, remove an existing rule
- **direction**: INPUT, OUTPUT or FORWARD (see the previous section)
- **protocol**: mainly TCP or UDP
- **port**: the port number you want to create your rule for
  You can find a list of common ports here
- **action**: Define the choice to make for the corresponding traffic
  - ACCEPT: Allow access (while in whitelist mode)
  - REJECT: Deny access and tell the sender it's not allowed
  - DROP: Deny access but don't tell the sender

This is the short introduction to what you'll mainly use.

If you need further information, use "man iptables" or check this page for all parameters.

## Switch to whitelist

As you can see with “iptables -L”, we are in blacklist mode: ACCEPT all except the rules we add.

If you are in a stricter environment, switch to whitelist mode.
For example, if you are creating a free Wi-Fi in a hotel or other business, you probably want to allow only a few ports (like web and mails).

To do this, you need to create a list of all ports you want to allow.

If you do all the commands manually, you’ll lose access after the first one 🙂
So, the easiest way is to create a script that run all commands at once.

## Create the firewall script

- **Create a new file** with nano
  sudo nano /usr/local/bin/firewall.sh

- **Paste these lines** inside

```
#!/bin/sh
#Clear all rules
iptables -F

#Whitelist mode
iptables -P INPUT ACCEPT
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

#Allow PING for everyone
iptables -A FORWARD -p icmp -j ACCEPT

#Allow HTTP/HTTPS for WiFi clients
```

```
iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -j ACCEPT

#Allow POP/IMAP/SMTP for WiFi clients
iptables -A FORWARD -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -p tcp --dport 110 -j ACCEPT
iptables -A FORWARD -p tcp --dport 993 -j ACCEPT

#Allow PING for WiFi clients
iptables -A FORWARD -p icmp -j ACCEPT

#Allow NAT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

- You can **adapt these lines to your needs**
  This is just an example
  I don't drop on INPUT and OUTPUT in this sample script, but you can do it if you want a better control on your network usage
- **Save and exit** (CTRL+O, CTRL+X)
- **Add execution right to this script**
  `sudo chmod +x firewall.sh`
- **Run it**
  `sudo /usr/local/bin/firewall.sh`
- **Check if everything works fine**
  If something goes wrong, reboot the Raspberry Pi to recover all access
  Then find what you miss in your script

When it's ok, you can add it in the init tab to start it on boot

## Make your configuration persistent

The GitHub script we installed before uses the file /etc/iptables.ipv4.nat to save the configuration

So once it's working, you can save your current configuration inside :

```
sudo iptables-save > /etc/iptables.ipv4.nat
```

This will load the configuration file on load and apply directly the changes

# Monitor the network

Now that the router is working (with a firewall or not), we can add other packages to improve the Raspberry Pi capabilities

In this part, I suggest adding a web interface to monitor what happens on the Raspberry Pi (and on the network)

## What's this tool?

The tool I chose is Webmin

It's a web interface, easy to install, that shows you all the current configuration, and several statistics and graphs about the system usage

You can even change the configuration from this interface

If you know other ones tell me in the community

It's a tool that exists for a long time, but I don't know a more recent one to do this

## Webmin Installation

- **Check the latest Webmin version** from this page (Source file)
- **Download the file** like this:
  wget https://prdownloads.sourceforge.net/webadmin/webmin-1.941.tar.gz
- **Extract the archive** and move to the new folder
  tar -zxvf webmin-1.941.tar.gz cd webmin-1.941
- **Run the setup**
  sudo ./setup.sh /usr/local/webmin
  Keep all the default values
  Set a password and choose if you want to use SSL or not
  Wait a few seconds for the installation to finish
- **The tool is now available at http://<IP>:10000**
  Log in with the user name and password you just created

## Webmin interface

**I let you discover the web interface and browse through the menu**

There are A LOT of tools inside, we don't need all of this

We'll mainly use those in the "Networking" part

For example, you can enable bandwidth monitoring or manage your firewall configuration from here

We'll come back later to this interface, for the proxy configuration for example

It's the next part

# Add a proxy and a web filter

## What's a proxy?

A proxy has three main roles:

- Creating a cache of all Internet pages, to increase web browsing speed
- Log all pages, to generate reports (top domain, top traffic, ...)
- And we can add a website blocker, to deny access to some kinds of content

To do this, we'll install Squid (the proxy) and SquidGuard (the filter) on the Raspberry Pi

## Squid installation

- Squid is available in the repository. **Install it with**:
  sudo apt install squid
- **Backup the configuration file**:
  cd /etc/squid
  sudo mv squid.conf squid.conf.old
- **Switch to the root user** a few seconds and **remove all the comments quickly**
  sudo su
  cat squid.conf.old | egrep -v -e '^[[:blank:]]*#' | grep "\S" > squid.conf
  exit
  Then **edit the configuration file**:
  sudo nano squid.conf
- And **add these lines at the beginning**
  acl LocalNet src 192.168.42.0/24
  http_access allow LocalNet
  Squid works only on HTTP for the Wi-Fi network (42.X)
- **Restart Squid** to apply changes
  sudo systemctl restart squid

Once Squid is configured, you have two choices to use it:

- **Configure your web browser to use the Raspberry Pi as the HTTP Proxy**
  It depends on your browser, but you should find this option in Options > Network settings
- **Configure iptables** to redirect automatically all HTTP traffic to squid
  It should be something like this:
  ```
  iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j DNAT --to 192.168.42.1:3128
  iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
  ```

Then check that HTTP websites are working fine

You can see all the websites visited in the Squid log file :

```
sudo tail -f /var/log/squid/access.log
```

## Webmin add-on

There is a Webmin add-on called “calamaris” you can install to monitor the proxy efficiency

Install it with apt:

```
sudo apt install calamaris
```

And then go back to webmin, in the unused modules to find new tools for squid monitoring

# SquidGuard installation

We can now move to the SquidGuard installation:

- **Install the squidguard package**

  sudo apt install squidguard

- **Download a list of websites by category**

  wget http://squidguard.mesd.k12.or.us/blacklists.tgz

- **Extract files from the archive**

  tar -zxvf blacklists.tgz

  While extracting, you'll see some blacklist categories appear on your screen

  You can choose one and use it for the SquidGuard configuration later

  You can open the files to get some websites examples

- **Move files to the SquidGuard folder**

  sudo mv blacklists /var/lib/squidguard/db

- **Archive the original SquidGuard configuration file**

  cd /etc/squiguard

  mv squidGuard.conf squidGuard.conf.old

- **Create a new configuration file**

  sudo nano squidGuard.conf

- **Paste these lines**

  dbhome /var/lib/squidguard/db logdir /var/log/squidguard dest violence {

  domainlist blacklists/violence/domains

  urllist blacklists/violence/urls

  log violenceaccess

  }

  acl {

  default

  {

  pass !violence

  redirect http://localhost/block.html

  } }

This is a sample file, you can block whatever you want
You can also add more categories by creating more dest sections

- **Save and exit** (CTRL+O, CTRL+X)
- **Build the SquidGuard database**

  ```
  sudo squidGuard -C all -d
  ```

  This can take a long time. If it's too long, remove files you don't need from the blacklists folder
  Use a screen if you don't stay in front of your computer (screen -S build)
  So if the SSH connection with your computer is lost, this will not stop the build process

- **Restart Squid to apply changes**

  ```
  sudo service squid restart
  ```

It should be ok now, try to access a URL from the domain list and check that you are blocked by squidGuard

## Related questions

**Is it possible to add an Ad Blocker brick in this router?** Yes, you can use Pi-Hole to do this, it's easy to install and you just need to set the Raspberry Pi as your DNS server (manually or in the DHCP configuration file, see Firewall > DNS issues for more information)

**Is it possible to use a Raspberry to build a "full Ethernet" router?** Yes, you can add an Ethernet hat to your Raspberry Pi like this expansion Hat (more details on Amazon). It's perfect for a firewall.



**Want to chat with other Raspberry Pi enthusiasts?** Join the community, share your current projects and ask for help directly in the forums.

**You may also like:**

- 25 awesome Raspberry Pi project ideas at home
- 15 best operating systems for Raspberry Pi (with pictures)
- My book: Master your Raspberry Pi in 30 days

# Conclusion

That's it, you should now have better knowledge on how to build a complete firewall router with proxy on a Raspberry Pi

I hope it's working for you.

It took me a lot of time to write this post with many tests I didn't include here, but

you have the most important things, with the best tools

If you have any issues, ask your question in the community, we'll try to help you

Also, these tools are basically Linux stuff and you can find a lot of help on the Internet to go further

I give you all the documentation links here if needed:

- RPI Wifi router
- Iptables
- Squid
- SquidGuard
- Pi Hole