Tor-WLAN with Raspberry Pi

Thoughts upfront

The following is not for a high-level of anonymity and privacy; if anonymity is really crucial please look for more secure solutions like Tails or Whonix instead. Using Tor will effectively only hide your IP address, i.e. no-one can see your traffic as it leaves your location, however, starting from the Tor Exit the IP traffic looks the same as without Tor – just coming from the Tor-Exit and not from your IP address directly. Any private data included in your traffic itself will still be visible (to everyone if not encrypted, i.e. using http and the website provider if using https encrypted traffic). Keep in mind that although many sites (seem) to use https they are in fact hosted by companies like Cloudflare which terminated the https tunnel and have full access to everything sent to the website. Also your specific browser can be identified easily (although not related to you in person) via so-called fingerprinting principles.

What you will need: Raspberry Pi 3 (comes with WLAN included), a LAN cable, a SD card and a computer to prepare the SD card and then configure the Raspberry Pi. How to setup the Raspberry Pi as a mini-server (i.e. headless, without a display) is described in a separate article and is a prerequisite for the following.

Enable WLAN (Access Point with DHCP)

First, we shutdown the WLAN interface while we configure it in the next steps:

```
sudo ifdown wlan0
```

Disable the DHCP Client on WLAN

Normally the dhcpcd daemon (DHCP client) will search the network for a DHCP server to assign a IP address to wlan0. This is disabled by editing the configuration file:

```
sudo nano /etc/dhcpcd.conf
```

We tell the DHCP client that on the wlan0 interface we have a static IP address and it should not configure anything else on it. This is achieved by adding at the very end of the config file:

```
interface wlan0
    static ip_address=192.168.200.1/24
    nohook wpa_supplicant
```

and then restart the service so the new config is loaded:

```
sudo systemctl restart dhcpcd
```

Enable the Access-Point Server

Install the WiFi Access-Point server:

```
sudo apt install hostapd
```

Create a new config file for the access point:

```
sudo nano /etc/hostapd/hostapd.conf
```

Paste the below – and change the country code, the WLAN name (SSID) and the WLAN password before saving the file:

```
interface=wlan0
driver=nl80211
country_code=FR
ssid=Your-WLAN-Name
hw_mode=g
channel=6
```

```
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Your-WLAN-Password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

The channel could be set to either 1, 6 or 11 depending on other wireless networks in your area. If you don’t know just take one by chance. Next, point the server to the right config file:

```
sudo nano /etc/default/hostapd
```

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

And then ensure the service is started:

```
sudo systemctl start hostapd
sudo systemctl status hostapd
sudo systemctl enable hostapd
```

Enable the DHCP Server

Start out by installing dnsmasq with the following command

```
sudo apt install -y dnsmasq
```

and then make a backup of the initial standard configuration for later reference before enabling the new config:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.backup
```

We will put the DHCP configuration in the file dnsmasq.dhcp.conf and edit it with:

```
sudo nano /etc/dnsmasq.d/dnsmasq.dhcp.conf
```

The new configuration should be something similar to the following, however with the ip address information and things like the MAC addresses changed to your local setup:

```
# Configuration file for dnsmasq (DHCP part)

# Extra logging for DHCP: log all the options sent to DHCP clients and the tags
# used to determine them.
#log-dhcp

# Suppress logging of the routine operation of these protocols. Errors and problems
# will still be logged. quiet-dhcp and quiet-dhcp6 are over-ridden by log-dhcp.
quiet-dhcp

# This is our local network to be served as DHCP on wlan0
dhcp-range=wlan0,192.168.200.20,192.168.200.199,255.255.255.0,24h

# This is the only DHCP-server for wlan0
dhcp-authoritative

# This is your local domain name. It will tell the DHCP server which
# host to give out IP addresses for.
domain=alpha

# Set the Gateway IP address
dhcp-option=option:router,192.168.200.1

# Set the DNS server
dhcp-option=option:dns-server,192.168.200.1

# Set the NTP time server address
dhcp-option=option:ntp-server,192.168.200.1

# adapted for a typical dnsmasq installation where the host running
# dnsmasq is also the host running samba.
```

```
# you may want to uncomment some or all of them if you use
# Windows clients and Samba.
dhcp-option=19,0            # option ip-forwarding off for
clients
dhcp-option=44,0.0.0.0       # set netbios-over-TCP/IP
nameserver(s) aka WINS server(s)
dhcp-option=45,0.0.0.0      # netbios datagram distribution
server
dhcp-option=46,8           # netbios node type


# Always allocate the same DHCP IP address based on the MAC
address
dhcp-host=00:25:a5:34:af:e7, Ethernet-Bridge,  192.168.200.20,
infinite
dhcp-host=84:cf:bf:89:9b:fd, Mobile,        192.168.200.120,
infinite
```

To test if the config for dnsmasq is free of errors check the syntax with:

```
sudo dnsmasq --test
```

DNS Server and Cache

We put the DNS server configuration in a separate file (both DHCP and DNS is the same dnsmasq software and the config could well be in one file; but it’s a bit easier to read that way):

```
sudo nano /etc/dnsmasq.d/dnsmasq.dns.conf
```

and paste in the following

```
# Configuration file for dnsmasq (DNS part)

# For debugging, log each DNS query as it passes through
dnsmasq.
#log-queries

# Listen on these IP addresses for DNS requests (always
include localhost):
```

```
listen-address=127.0.0.1,192.168.200.1,10.80.0.1

# Where to send DNS request to which can't be resolved locally.
# Here we use the local Tor service listening on port 5300 (see /etc/tor/torrc)
server=127.0.0.1#5300

# The bind-interfaces directive instructs dnsmasq to bind only to
# the network interface specified in the listen-address directive.
bind-interfaces

# The no-hosts directive instructs dnsmasq not to read hostnames
# from /etc/hosts.
no-hosts

# Read hostname information from this file in addition
addn-hosts=/etc/dnsmasq.hosts

# Never forward plain names (without a dot or domain part)
upstream
domain-needed

# Never forward addresses in the non-routed address spaces.
bogus-priv

# If you don't want dnsmasq to read /etc/resolv.conf or any other
# file, getting its servers from this file instead, then uncomment this.
no-resolv

# If you don't want dnsmasq to poll /etc/resolv.conf or other resolv
# files for changes and re-read them then uncomment this.
no-poll

# Set this if you want to have a domain
```

```
# automatically added to simple names in a hosts-file.
expand-hosts

# Number of hostnames being cached in RAM for DNS lookups
cache-size=10000

# Do not cache negative responses
no-negcache

# Resolve these domains locally only, don't send upstream
local=/alpha/
local=/local/

# Some simple domain blocking (doesn't block the IP, just domain-name)
address=/doubleclick.net/0.0.0.0
address=/google-analytics.com/0.0.0.0
address=/analytics.google.com/0.0.0.0
address=/amazon-adsystem.com/0.0.0.0
address=/analytics.yahoo.com/0.0.0.0
address="cn"0.0.0.0
```

To resolve the hostname of this server we need to provide it in a separate host-file:

```
sudo nano /etc/dnsmasq.hosts
```

and just list the local hostname and its IP address on the WLAN:

```
192.168.200.1   MyTorGateway
```

so can use its name on other devices on the WLAN (instead of the IP address).

To test if the above config for dnsmasq is free of errors run:

```
sudo dnsmasq --test
```

If everything is fine, start the dnsmasq service and check its status with:

```
sudo service dnsmasq start
```

sudo service dnsmasq status

To see the listening ports, check with:

sudo netstat -tulpn | grep dnsmasq

Enable a Time Server (NTP via chrony)

Since the aim is to route all traffic from the wireless network through Tor we need to provide the time to the WLAN since this can’t be done over Tor. The best ntp server for our purpose seems to be chrony. It’s installed via:

sudo apt install -y chrony

and then we configure it (after saving the initial config for later reference):

```
sudo mv /etc/chrony/chrony.conf /etc/chrony/chrony.conf.default
sudo nano /etc/chrony/chrony.conf
```

and paste in the following:

```
# Welcome to the chrony configuration file. See chrony.conf(5) for more
# information about usuable directives.

# The time server where we get our time from; here the internet router
server 192.168.178.1 iburst
#pool pool.ntp.org iburst

# Time server for the following subnet
allow 192.168.200.0/24

# Listen on this interface for client ntp requests
bindaddress 192.168.200.1
```

```
# This directive specify the file into which chronyd will store the rate
# information.
driftfile /var/lib/chrony/chrony.drift

# Uncomment the following line to turn logging on.
#log tracking measurements statistics

# Log files location.
logdir /var/log/chrony

# The hardware clock (rtc) is always UTC
rtconutc

# This directive enables kernel synchronisation of the real-time clock.
rtcsync

# Step the system clock instead of slewing it if the adjustment is larger than
# one second, but only in the first three clock updates.
makestep 1 3
```

If the devices on the WLAN don’t pick up time after a few minutes it might be necessary to list the ntp server 192.168.200.1 in the /etc/ntp.conf (or similar) config files of those devices.

Data Forwarding (WLAN ↔ LAN)

Only in case you want to access servers on your LAN (the one directly attached to the internet) from the Raspberry WLAN you need to allow IP4 forwarding on the Raspberry:

```
sudo nano /etc/sysctl.conf
```

Set the forwarding option to 1:

```
net.ipv4.ip_forward=1
```

and then reload the configuration:

```
sudo sysctl -p
```

You can double-check that it worked with:

```
sudo sysctl net.ipv4.ip_forward
```

The WLAN is now setup and we only need to enable the interface:

```
sudo ifconfig wlan0 up
sudo reboot
```

Now you should test it – you should be able to connect to the new WLAN just defined and if IP4 forwarding was enabled you should have access to the Internet (no Tor yet though).

Install and configure Tor

Getting Tor up and running is really easy (including the recommended package to keep the key updated):

```
sudo apt install -y tor
```

Keep the initial config file for reference before changing it:

```
sudo mv /etc/tor/torrc /etc/tor/torrc.default
sudo nano /etc/tor/torrc
```

```
Log notice file /var/log/tor/tor.log
AvoidDiskWrites 1

ExitRelay 0
ExitPolicy reject *:*

TransPort      127.0.0.1:9040      IsolateClientAddr
IsolateClientProtocol IsolateDestAddr IsolateDestPort
DNSPort   127.0.0.1:5300
TransPort    192.168.200.1:9040    IsolateClientAddr
IsolateClientProtocol IsolateDestAddr IsolateDestPort
DNSPort   192.168.200.1:5300
TransPort      10.80.0.1:9040      IsolateClientAddr
IsolateClientProtocol IsolateDestAddr IsolateDestPort
```

```
DNSPort   10.80.0.1:5300

AutomapHostsOnResolve 1
AutomapHostsSuffixes .onion,.exit
VirtualAddrNetworkIPv4 10.12.0.0/16

ExitNodes     9EAD5B2D3DBD96DBC80DCE423B0C345E920A758D,
85D4088148B1A6954C9BFFFCA010E85E0AA88FF0
```

Check that Tor is happy with the config file and there are no issues with it:

```
sudo -u debian-tor tor --verify-config
```

The last line should say that everything looks fine. Then reload the new Tor config:

```
sudo systemctl reload tor.service
```

Check the logs for what Tor does and if it complains about anything – the following commands might be useful to check for any errors:

```
sudo systemctl status tor.service
sudo cat /var/log/tor/tor.log
journalctl | grep 'Tor'
```

In case of issues with tor one could increase the level of logging temporarily – in the tor config file /etc/tor/torrc replace the ‚log …‘ configuration with debug, info, notice, warn, or err. For example a ‚log info…‘ statement will provide more details. In the long run the log-level of ’notice‘ should be the best choice.

If anonymity and privacy is not your major concern but rather useability you might want to specify one or more Exit nodes to be used for all your traffic. This will ensure for example that you can specify the country where you seem to be coming from. To do that, just add to torrc config file something like:

```
ExitNodes 85D4088148B1A6954C9BFFFCA010E85E0AA88FF0
```

Enable the Firewall

First install the firewall frontend and enable the firewall:

```
sudo apt install nftables -y
sudo systemctl enable nftables.service
```

Enable the following firewall rules, starting with a config file in your home directory

```
nano ~/nftables.conf
```

and paste in

```
#!/usr/sbin/nft -f

flush ruleset

define wireguard_port = 40042

table ip filter {
        chain INPUT {
                  type filter hook input priority 0; policy
drop;
                iif lo accept
                ct state related,established accept
                ct state invalid drop
                 ip saddr {192.168.178.0/24, 192.168.200.0/24}
tcp dport ssh ct state new accept
                 ip saddr {192.168.178.0/24, 192.168.200.0/24}
icmp type echo-request accept
                      udp dport $wireguard_port accept
# might not be needed
                 iifname {"wlan0", "wg0"} udp dport domain
accept
                 iifname {"wlan0", "wg0"} tcp dport domain
accept
                  iifname {"wlan0", "wg0"} udp dport {ntp,
bootps} accept
                iifname {"wlan0", "wg0"} tcp dport 9040 accept
        }
```

```
        chain FORWARD {
                 type filter hook forward priority 0; policy
drop;
                ct state related,established accept
                ct state invalid drop
                  iifname "wlan0" ip daddr 192.168.178.0/24
accept
        }

        chain OUTPUT {
                 type filter hook output priority 0; policy
drop;
                oif lo accept
                ct state related,established accept
                ct state invalid drop
                skuid "debian-tor" accept
                udp dport ntp accept
                ip daddr 127.0.0.1 accept
                 ip daddr {192.168.178.0/24, 192.168.200.0/24,
255.255.255.255} accept
        }
}

table ip nat {
        chain PREROUTING {
                type nat hook prerouting priority -100; policy
accept;
                 iifname "wlan0" udp dport domain redirect to
:53
                 iifname "wlan0" tcp dport domain redirect to
:53
                 iifname "wlan0" udp dport ntp    redirect to
:123
                  iifname "wlan0" ip daddr {192.168.178.0/24,
192.168.200.0/24} accept
                 iifname "wlan0" tcp flags & (fin|syn|rst|ack)
== syn redirect to :9040
                iifname "wg0" udp dport domain redirect to :53
                iifname "wg0" tcp flags & (fin|syn|rst|ack) ==
syn redirect to :9040
        }
```

```
        chain INPUT {
                  type nat hook input priority 100; policy
accept;
        }

        chain POSTROUTING {
                type nat hook postrouting priority 100; policy
accept;
                  oifname "eth0" ip saddr 192.168.200.0/24
masquerade
        }

        chain OUTPUT {
                  type nat hook output priority -100; policy
accept;
                skuid "debian-tor" accept
                 ip daddr {192.168.178.0/24, 192.168.200.0/24}
accept
                 tcp flags & (fin|syn|rst|ack) == syn redirect
to :9040
        }
}

table ip6 filter {
        chain INPUT {
                  type filter hook input priority 0; policy
drop;
                iif lo counter accept
        }

        chain FORWARD {
                  type filter hook forward priority 0; policy
drop;
        }

        chain OUTPUT {
                  type filter hook output priority 0; policy
drop;
                oif lo counter accept
                counter reject
        }
```

}

and activate these firewall rules with

sudo nft -f nftables.conf

In case something goes horribly wrong (e.g. you lock ssh sessions) you can hard reboot the server and will start without the firewall rules.

To list the active rules applied by nft currently just check with:

sudo nft list ruleset

Note that nft uses its own matching of service names to port numbers – to see the list simply type in:

nft describe tcp dport

Once you're happy with them working make them permanent with copying them to the standard place (enabled on reboot):

mv ./nftables.conf /etc/nftables.conf

Note on Firefox configuration

If you're using firefox as browser you also need to change its config as by standard firefox is blocking access to services residing on tor (onion services). To do so, type

about:config

in the field at the top (where you would type in e.g. a web-address usually) then search for „onion" at the top and change the value for network.dns.blockDotOnion to „false" by a double-click on it. Btw, Chrome and Edge aren't blocking tor by default and they will work right away with the standard configuration. Test access to onion-services by browsing to e.g. „https://protonirockerxow.onion".

If you don’t use WebRTC services then it’s strongly recommended to disable media.peerconnection.enabled in about:config. Otherwise your local IP address will be visible to everyone on the internet (also while connected via Tor).

It might be good to also minimize the tracking via fingerprinting. One example is to enable privacy.resistfingerprinting by setting to true. Also one should disable plugin.expose_full_path and dom.battery.enabled by setting them both to false.

Optional but highly recommended is to install the „Privacy Pass“ Firefox add-on; it helps to avoid many of the annoying challenges one has to complete when using Tor. Additionally, and also optional it might be wise to install the „uBlock Origin“ add-on as well to avoid advertisements, malware, etc. which would otherwise cause additional traffic and would increases tracking capabilities.

Additional recommended add-ons for Firefox are: „Cloud Firewall“, „HTTPS Everywhere“, „Privacy Badger“, and „ClearURLs“.

Some Improvements and Hardening

IP Forwarding WLAN ↔ LAN

If no connection between WLAN and LAN is required one should remove IPv4 forwarding – this can be done in two ways (one is sufficient, two are safer):

sudo nano /etc/sysctl.conf

net.ipv4.ip_forward=0

sudo sysctl -p

or remove the rule for the forwarding filter rule of ferm.

Add the official Tor Repository

We add the official repository on the Tor network of the tor-project for Debian (as outlined at www.torproject.org/docs/debian.html.en):

sudo nano /etc/apt/sources.list.d/torproject.org.list

and put in the following:

```
# Tor onion-site replacement for:
# deb http://deb.torproject.org/torproject.org stretch main

deb http://sdscoq7snqtznauu.onion/torproject.org stretch main
```

However, Debian is not supporting anonymous and safe software repos by standard, so we need to enable it first – create a new apt config file:

sudo nano /etc/apt/apt.conf.d/80onion-repos

and put in just one line:

```
Acquire::BlockDotOnion "false";
```

Now finally update and upgrade the system with:

```
sudo apt update
sudo apt upgrade -y
```

Automatic updates for Tor

It’s also recommended to get the tor software upgraded automatically. Assuming that the package ‚unattended-upgrades‘ is installed already, we simply need to add the updates from the torproject to the config:

sudo nano /etc/apt/apt.conf.d/50unattended-upgrades

and add the line with the TorProject repository:

```
"origin=Debian,codename=${distro_codename},label=Debian-Security";
        "origin=TorProject,codename=${distro_codename}";
};
```

Sidenote: information about the ‚origin‘ and other parameters can be found in the files /var/lib/apt/lists/*_InRelease.

Block many sites (adservers)

If you want to have a much more extensive ad-blocking done by dnsmasq one could just download the latest quite complete adserver list from pgl.yoyo.org/adservers/ (formated to be used with dnsmasq as plain text file) and save it as e.g. dnsmasq.adservers.conf in the folder /etc/dnsmasq.d/.

This can be even automated with a little script like the following – it must be run with root priviliges (e.g. with sudo) as it needs to install the new adblocking file in /etc:

```
#!/bin/bash

if [[ $EUID -ne 0 ]]; then
   echo "This script must be run as root (use sudo)"
   exit 1
fi

curl -o adservers 'https://pgl.yoyo.org/adservers/serverlist.php?hostformat=dnsmasq&mimetype=plaintext'

# Do some work on the file:
# Now remove MS-DOS carriage returns, replace 127.0.0.1 with 0.0.0.0 (much faster),
# clean up trailing blanks,
# Pass all this through sort with the unique flag to remove duplicates and save the result

sed -e 's/\r//' -e 's/127\.0\.0\.1/0\.0\.0\.0/' -e 's/[\t]*$//' < adservers | sort -u > /etc/dnsmasq.d/dnsmasq.new-
```

```
adservers.conf

rm adservers

# Seems like the dnsmasq configtest doesn't work (always returns success)
if ! /usr/sbin/dnsmasq --test; then
    printf '%s\n' 'dnsmasq configtest failed!' >&2
    rm /etc/dnsmasq.d/dnsmasq.new-adservers.conf
    exit 1
fi

mv   --force   /etc/dnsmasq.d/dnsmasq.new-adservers.conf /etc/dnsmasq.d/dnsmasq.adservers.conf
/bin/systemctl reload dnsmasq.service
```

Save this file in the /root folder (home folder for root) and add it as a cron-job for root:

```
sudo crontab -e
```

adding at the end something similar to:

```
15 04 * * * /root/adservers.sh > /root/adservers.log 2>&1
```

to run it each night at 4:15. The only thing which doesn’t seem to work is to check the dnsmasq config to be valid (the check always returns a successful check).

Keep in mind though that is only blocking DNS lookups, not IP addresses. Any direct connection to an IP address is not blocked with the above (that requires a blocking within the netfilter part; ipset is the command to look for to achieve something on the IP level).

Remote Access (VPN) via

WireGuard

```
sudo mkdir /etc/wireguard
cd /etc/wirguard
```

Configure the server

First create the keys:

```
umask 077
wg genkey > private.key
wg pubkey < private.key > public.key
```

Add a new interface for WireGuard and assign an IP address to it:

```
sudo ip link add dev wg0 type wireguard
sudo ip address add dev wg0 10.80.0.1/24
```

Configure the new interface:

```
sudo nano /etc/wireguard/wg0.conf
```

and add something along the lines of:

```
[Interface]
Address = 10.80.0.1/24
ListenPort = 40042
PrivateKey = xxxxx

[Peer]
# Smartphone
PublicKey = PEER_PUBLIC_KEY
AllowedIPs = 10.80.0.20/32

[Peer]
# Laptop
PublicKey = PEER_PUBLIC_KEY
AllowedIPs = 10.80.0.24/32
```

and finally enable it:

```
sudo ip link set up dev wg0
```

To check the status you can use the following commands:

```
sudo ip addr show dev wg0
sudo wg showconf wg0
sudo networkctl status wg0
```

To permanently enable the wireguard server (e.g. on reboot) run the following command on the shell:

```
sudo systemctl enable wg-quick@wg0.service
sudo systemctl start wg-quick@wg0.service
sudo systemctl status wg-quick@wg0.service
```

Configure the client (e.g. smartphone)

Probably the easiest way to configure your wireguard app on the smartphone is to use a QR code – just generate it on the server command line with:

```
qrencode -t ansiutf8 < client.conf
```

To manually install the WireGuard VPN (e.g. on a laptop) create a config file:

```
nano /etc/wireguard/wg0.conf
```

with something along the lines of

```
[Interface]
PrivateKey = <LOCAL-PRIVATE-KEY>
Address = 10.80.0.24/32
DNS = 10.80.0.1

[Peer]
PublicKey = <SERVER-PUBLIC-KEY>
Endpoint = <HOST-NAME>:40042
AllowedIPs = 0.0.0.0/0, ::/0
```

To start and stop the VPN use:

```
sudo wg-quick down wg0
sudo wg-quick up wg0
```

If there are issues also don’t forget to check that traffic forwarding is allowed by linux and the firewall and that the wireguard traffic is allowed by the firewall (cf. above).