Host a Tor Relay

If you want to support the Tor project and you have some bandwidth to share (at least 10 Mbps in both directions, i.e. download and upload) you should consider hosting a Tor Relay. It can be done quite easily with a Raspberry Pi or any similar hardware as described below.

Our assumption is that we connect the Raspberry Pi with an ethernet cable to our ISP router and that we have a wireless LAN running for our private, local machines already. We will connect the Raspberry Pi to this WLAN for administration purposes (so this is not done over the internet connection on eth0).

Initial Setup as a Server

The very first step to start out is to get a new Raspberry Pi and do the basic setup as a headless server (described in a separate post). For the sake of increased security you shouldn't use it for anything else than just to run the Tor Relay, so the Raspberry Pi should be dedicated to this task.

Install Tor and configure it

First we add the official repository 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:

```
deb https://deb.torproject.org/torproject.org buster main
```

Note that you might need to replace the debian release name which is currently „buster“ with a newer one.

Then install the keys signing the tor packages (maybe good to check torproject.org/docs/debian.html for the latest updates):

```
sudo gpg --recv A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89
sudo gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 |
sudo apt-key add -
```

and finally install the tor software (including the recommended package to keep the key updated):

```
sudo apt update
sudo apt install -y tor deb.torproject.org-keyring
```

To configure the tor service as a relay edit the config file:

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

and enter something like (change the details to your setup as required):

```
NickName MyNewRelay
ContactInfo myemail@example.org

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

ExitRelay 0
ExitPolicy reject *:*

ORPort  6080
DirPort 6443

RelayBandwidthRate  10 Mbit
RelayBandwidthBurst 15 Mbit

TransPort 127.0.0.1:9040
DNSPort   127.0.0.1:5300

AutomapHostsOnResolve 1
AutomapHostsSuffixes .onion,.exit
VirtualAddrNetworkIPv4 10.10.0.0/16
```

Now let tor check if the above config file is correct:

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

The last line should say that everything looks fine.

The contact information is optional but might be quite handy for others if there should be something strange with the relay. It doesn’t have to be an email address but could be any

kind of text.

In case you are running more than just one Tor Relay you have to also include a „MyFamily“ option in the config above and list all your key-fingerprints of your Tor Relays in each of the torrc config files. You get the fingerprint with sudo -u debian-tor tor –list-fingerprint and remember that there must be a $ (Dollar-sign) at the beginning of each fingerprint.

Port Forwarding on ISP Router

In our example the Raspberry Pi (our Tor Relay) sits behind a router which is the gateway into the internet (often provided by the ISP). With the tor configuration above we need to establish port forwarding on this internet router, so TCP traffic coming from the internet (on ports 6080 and 6443) is forwarded to out Tor Relay (on the same ports).

If you would like to use other ports to the outside world (internet) than on the Tor Relay server itself, the Tor config file (torrc) needs to have something like:

```
ORPort    80 NoListen
ORPort  6080 NoAdvertise
DirPort  443 NoListen
DirPort 6443 NoAdvertise
```

The port forwarding on the ISP router then obviously has to forward ports 80 and 443 to ports 6080 and 6443 respectively on the Tor Relay.

The ports chosen are kind of arbitrary and we are free to take whatever we like. One advantage of advertising (i.e. using) ports 80 and 443 towards the internet is that they are very unlikely to be blocked as they are usually taken for http and https traffic. The drawback is that you can't use these ports for something else (like a web-presence). Also some routers seem to have issues with port-forwarding these ports (e.g. lost after a router-reboot).

The details on how port forwarding is configured on the internet router depends heavily on that device but usually each of these kind of routers offers this feature somehow (just search the internet in case this is not obvious).

Start and Test it!

First restart Tor (so it picks up the latest configuration):

```
sudo systemctl restart tor
```

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
```

You are perfectly fine if you see something like „*Self-testing*

indicates your ORPort / DirPort is reachable from the outside„. If there are no issues your new Tor Relay will also become visible on the torproject metrics-webpage at metrics.torproject.org/rs.html (this might take a few hours though, so be patient).

One could also increase the level of logging information written by tor. Just change the option in the /etc/tor/torrc configuration file – after the „log“ statement one could place either debug, info, notice, warn, or err. Additionally, one could (temporarily, for debugging) turn off the scrubbing of sensitive information in the log-files as well. So for debugging include something like the following in the torrc

```
SafeLogging 0
Log info file /var/log/tor/tor.log
```

Once running fine one should probably keep the logging at the ’notice‘ level though.

Also note that it takes up to 2 months until a new Tor Relay gets fully used – and since there is not always traffic available it will mostly never really run at the full possible bandwidth. See this article for some background on it: blog.torproject.org/lifecycle-new-relay.

Enable the Firewall

(nftables)

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

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

and paste in (ensure the port numbers of tor defined above match the firewall rules here!):

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

flush ruleset

table ip filter {
    chain input {
        type filter hook input priority 0; policy drop;

        iifname lo accept

        ct state established,related accept
        ct state invalid drop

        tcp dport ssh ct state new limit rate 10/minute accept

        iifname eth0 tcp dport {6080, 6443} accept
```

```
        icmp type echo-request limit rate 1/second accept
    }

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

    chain output {
        type filter hook output priority 0; policy drop;
        oifname lo accept

        ct state established,related accept
        ct state invalid drop

        skuid "debian-tor" accept

        oifname eth0 udp dport ntp counter accept
         oifname wlan0 tcp dport 8086 accept     # sending
measurements to influxdb
        oifname wlan0 udp dport domain counter accept   # not
needed ???
         ip daddr 127.0.0.1 counter accept       # not needed
???
          ip daddr { 192.168.178.0/24, 192.168.200.0/24,
255.255.255.255 } accept
    }
}

table ip nat {
    chain input {
        type nat hook input priority 100; policy accept;
    }

    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 counter
redirect to :9040
```

```
#        udp dport domain counter redirect to :5300
    }
}
```

and then activate these firewall rules with

```
sudo nft -f nftables.conf
```

One can check the applied firewall rules (and also see the counters) with

```
sudo nft list ruleset
```

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

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):

```
 sudo cp /etc/nftables.conf /etc/nftables.conf.default
 sudo cp nftables.conf /etc/nftables.conf
```

Setup dnsmasq

We use dnsmasq to handle all DNS requests initially since there we can configure which upstream server to use for which domains. In our case we use the (local, private) DNS server 192.168.200.1 on wlan0 to resolve our private domain and send everything else into Tor.

Install dnsmasq, save its default configuration for later reference and edit the config:

```
sudo apt install -y dnsmasq
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.default
sudo nano /etc/dnsmasq.d/dnsmasq.conf
```

and paste in:

```
# Configuration file for dnsmasq (DNS part, DHCP is disabled by default)

#Optional logging can be enabled to support problem determination
#log-dhcp
#log-queries
#log-facility=/var/log/dnsmasq.log

# Listen on these IP addresses for DNS requests (always include localhost):
listen-address=127.0.0.1

# 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
server=/alpha/192.168.200.1

# 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
```

and save the file. To test if the config for dnsmasq is free of errors check the syntax with:

```
sudo dnsmasq --test
```

We also need to disable that dnsmasq takes nameservers from resolvconf – edit the file

```
sudo nano /etc/default/dnsmasq
```

and uncomment the last line so it reads:

```
...
IGNORE_RESOLVCONF=yes
```

Now start the dnsmasq service:

```
sudo systemctl start dnsmasq.service
sudo systemctl status dnsmasq.service
sudo systemctl enable dnsmasq.service
```

Finally do a few tests (abc is fine, server1 must be a valid hostname on your local network):

```
host -v cnn.com
host -v abc.onion
```

```
host -v server1.alpha
```

Some Improvement

Change Tor Repository

We can change the repository for the tor packages to their onion server:

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

and change it to:

```
# Repository for the Tor packages:
# deb https://deb.torproject.org/torproject.org buster main

deb http://sdscoq7snqtznauu.onion/torproject.org buster 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.

Monitoring

It’s good practice to regularly check the log files for errors for anything unusual:

```
sudo tail -25 /var/log/tor/tor.log
sudo tail -f /var/log/kern.log
```

Also check the torproject metrics-webpage at metrics.torproject.org/rs.html from time to time which should give a sufficient overview of the Tor Relay status. If you can browse onion websites you could even use the onion-metrics site at: rougmnvswfsmd4dq.onion/rs.html.

In case a more in-depth monitoring is required have a look at e.g. vnstat, theonionbox or (most advanced, usually not really required and with lots of shortcommings) Telegraf with InfluxDB and Grafana.

Tor-WLAN with Raspberry Pi 3

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=127.0.0.1,192.168.200.1,10.8.0.1

# 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
```

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
```

```
DNSPort   127.0.0.1:5300
TransPort 192.168.200.1:9040
DNSPort   192.168.200.1:5300

AutomapHostsOnResolve 1
AutomapHostsSuffixes .onion,.exit
VirtualAddrNetworkIPv4 10.192.0.0/10
```

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.

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

table ip filter {
        chain INPUT {
                  type filter hook input priority 0; policy
drop;

                iifname "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 22 accept
                 ip saddr 192.168.178.0/24 icmp type echo-
request counter packets 0 bytes 0 accept
                 ip saddr 192.168.200.0/24 icmp type echo-
request counter packets 0 bytes 0 accept
                 udp dport 1194 counter packets 0 bytes 0
accept
               iifname "wlan0" udp dport {123, 53, 67} accept
               iifname "tun*" udp dport {123, 53, 67} accept
               iifname "wlan0" tcp dport 9040 accept
               iifname "tun*" tcp dport 9040 accept
        }

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

        chain OUTPUT {
                 type filter hook output priority 0; policy
drop;
               oifname "lo" accept
               ct state invalid drop
               ct state related,established accept
               skuid "debian-tor" accept
               udp dport 123 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" 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 "tun*" udp dport 53 redirect to :53
                iifname "tun*" 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;
                iifname "lo" counter packets 5 bytes 245
accept
        }
```

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

        chain OUTPUT {
                type filter hook output priority 0; policy
drop;
                 oifname "lo" counter packets 5 bytes 245
accept
               counter packets 40 bytes 3241 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.

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):

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“, and „Privacy Badger“.

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=dnsm
asq&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).

Backup your Server

Once everything is running fine and stable it’s a good idea to create a full backup of the SD-card. Especially, as these cards are known to get broken quite easily; don’t be supprised if after one or two years the Raspberry Pi gives up, this is quite normal.

Create a Backup

To start out, shut down the Raspberry with sudo shutdown now and take out the SD-card. It’s easiest to store the backup on your linux laptop. Before plugging in the SD-card in your laptop run df -h on your laptop terminal then plug in the card and run df -h again. The difference tells you the device path of the SD-card; typically it will be something like /dev/sda1 or so. To address the whole SD-card you need to remove the last part (probably just a number, maybe p1 or so as well).

Now simply run the following to create a full one-by-one backup (image) of the card:

```
sudo dd bs=1M if=/dev/sda of=~/sd-card-backup.img
```

Note, that this will take some time (around 10 minutes for a 8 GB card) so be patient.

Restore a Backup

Unmount all the partitions of the SD-card (including the numbers)

```
sudo umount /dev/sda1
sudo umount /dev/sda2
```

and restore the backup with:

```
sudo dd bs=1M if=~/SD-card-backup.img of=/dev/sda
```

which again can take quite some time (just wait for the job to complete). To ensure everything is written out, type sudo sync .

x