CYVA Research

Raspberry Pi MQTT Configuration

06/21/18 10:09:29 AM

Table of Contents

Illustration Index

CYVA Raspberry Pi MQTT Configuration
06/21/18 10:09:29 AM

Introduction

Raspbian OS Build

From Linux

The manual steps for device prep for MQTT Service from NOOBS v2.8.1:

- Using Ubuntu as a base
- Use Gparted and remove all existing partitions of the MicroSD card.
  - Create a 16 to 32GB Primary Partition, depending on your needs.
    - Do not create a partition less than 12GB. If you do, you may not have the space on the device for a system upgrade to the next release of the Raspbian OS.
  - Format the partition to FAT32
    - mkfs.fat -F32 -v -l '/dev/sdb1'
  - Under manage flags – select boot
- Extract the NOOBS_v2_8_1.zip to the MicroSD card formatted above.
- Once the files are copied to the root of the MicroSD you should see something like the following:

Name	Description	Size	Modified
defaults	folder		03/14/2018 01:37
os	folder		04/24/2018 01:17
overlays	folder		04/18/2018 01:18
bcm2708-rpi-0-w.dtb	STL 3D model (binary)	21.7 KiB	04/18/2018 01:19
bcm2708-rpi-b.dtb	unknown	21.3 KiB	04/18/2018 01:19
bcm2708-rpi-b-plus.dtb	unknown	21.5 KiB	04/18/2018 01:19
bcm2708-rpi-cm.dtb	unknown	21.0 KiB	04/18/2018 01:19
bcm2709-rpi-2-b.dtb	STL 3D model (binary)	22.5 KiB	04/18/2018 01:19
bcm2710-rpi-3-b.dtb	STL 3D model (binary)	23.7 KiB	04/18/2018 01:19
bcm2710-rpi-3-b-plus.dtb	STL 3D model (binary)	23.9 KiB	04/18/2018 01:19
bcm2710-rpi-cm3.dtb	unknown	22.4 KiB	04/18/2018 01:19
bootcode.bin	unknown	50.8 KiB	04/18/2018 01:19
BUILD-DATA	plain text document	303 bytes	04/18/2018 01:19
INSTRUCTIONS-README.txt	plain text document	2.3 KiB	04/18/2018 01:18
recovery.cmdline	plain text document	99 bytes	04/18/2018 01:19
recovery.elf	executable	657.7 KiB	04/18/2018 01:19
recovery.img	Raw disk image	2.8 MiB	04/18/2018 01:19
recovery.rfs	Squashfs filesystem	27.2 MiB	04/18/2018 01:19
recovery7.img	Raw disk image	2.9 MiB	04/18/2018 01:19
RECOVERY_FILES_DO_NOT_EDIT	plain text document	0 bytes	04/18/2018 01:19
riscos-boot.bin	unknown	9.5 KiB	04/18/2018 01:18

Illustration 1: Directory Listing NOOBS Software

NOOBS Installation Instructions

Starting with Raspberry Pi NOOBS Read Me file:

1. Insert an SD card that is 8GB or greater in size into your computer.
2. Format the SD card using the platform-specific instructions below:
   (a) Windows (32GB cards and under)
      i. Download the SD Association's Formatting Tool from
         https://www.sdcard.org/downloads/formatter_4/eula_windows/
      ii. Install and run the Formatting Tool on your machine
      iii. Check that the SD card you inserted matches the one selected by the Tool
      iv. Click the "Format" button
   (b) Mac (32GB cards and under)
      i. Download the SD Association's Formatting Tool from
         https://www.sdcard.org/downloads/formatter_4/eula_mac/
      ii. Install and run the Formatting Tool on your machine
      iii. Select "Overwrite Format"
      iv. Check that the SD card you inserted matches the one selected by the Tool
      v. Click the "Format" button
   (c) Linux
      i. We recommend using gparted (or the command line version parted)
      ii. Format the entire disk as FAT32
   (d) Cards over 32GB
      i. Follow the instructions on
         https://www.raspberrypi.org/documentation/installation/sdxc_formatting.md
3. Extract the files contained in this NOOBS zip file.
4. Copy the extracted files onto the SD card that you just formatted so that this file is at the root directory of the SD card. Please note that in some cases it may extract the files into a folder, if this is the case then please copy across the files from inside the folder rather than the folder itself.
5. Insert the SD card into your Pi and connect the power supply.
6. Your Pi will now boot into NOOBS and should display a list of operating systems that you can choose to install.
7. If your display remains blank, you should select the correct output mode for your display by pressing one of the following number keys on your keyboard:

8. HDMI mode - this is the default display mode.
9. HDMI safe mode - select this mode if you are using the HDMI connector and cannot see anything on screen when the Pi has booted.
   Composite PAL mode - select either this mode or composite NTSC mode if you are using the composite RCA video connector.
   Composite NTSC mode

If you are still having difficulties after following these instructions, then please visit the Raspberry Pi Forums (**http://www.raspberrypi.org/forums/**) for support.

Connect Power and Boot

Insert the SD card into your Pi and connect the power supply.

Your Pi will now boot into NOOBS and should display a list of operating systems that you can choose to install. From the NOOBS screen select “Raspbian” install

After a few minutes.

Once the Raspberry Pi has booted into the GUI, click on the Raspberry in the upper left corner, then select Preferences, then Raspberry Pi Configuration.

Illustration 2: Raspbian OS: Raspberry Pi Configuration Menu

Under the “System” tab, there are a few items that need to be set so you can use VNC or SSH to access the Raspberry Pi. Since the VNC and SSH facilitate remote access, Raspbian requires the password to be changed before continuing. Assign a host name and choose a default resolution. If you don’t set the default resolution and you are running without a monitor, then any connection with VNC will have a tiny screen.

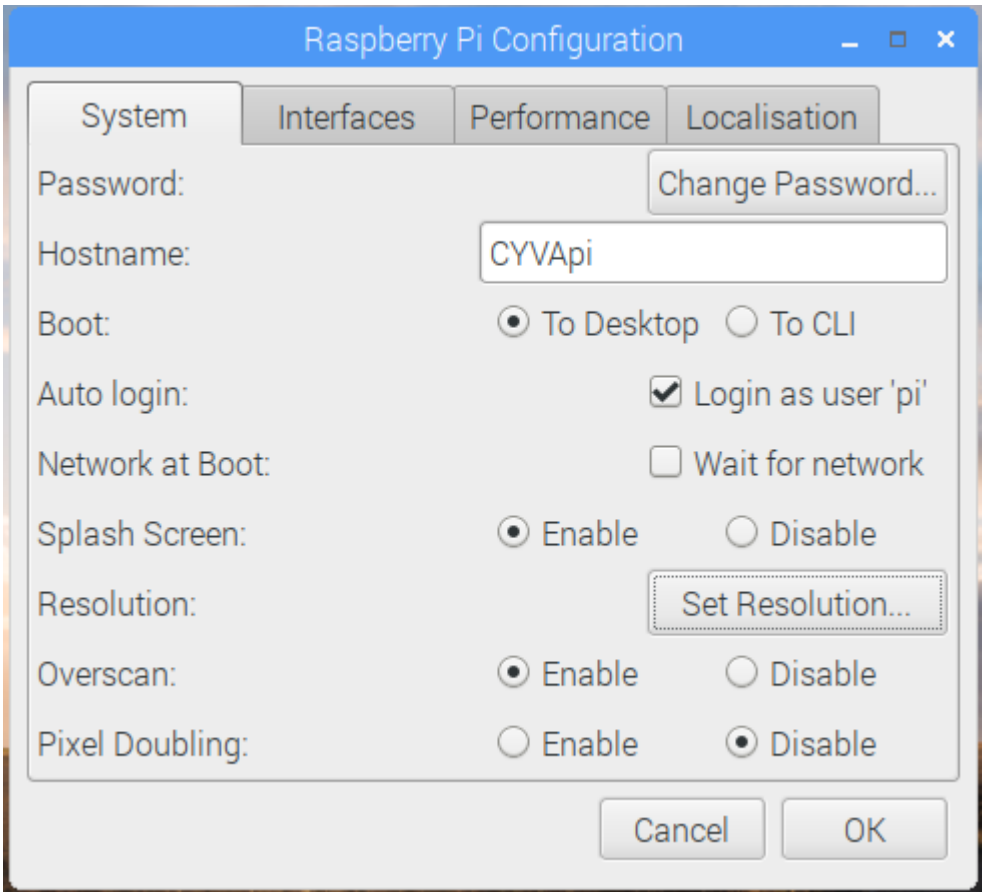


Illustration 3: Raspberry Pi System Tab

The next tab “Interfaces” is the screen where you can activate features that you desire. The CYVA MQTT does not require any settings other than SSH and VNC, however, if you plan on leaving a monitor attached, they remote access is not required.

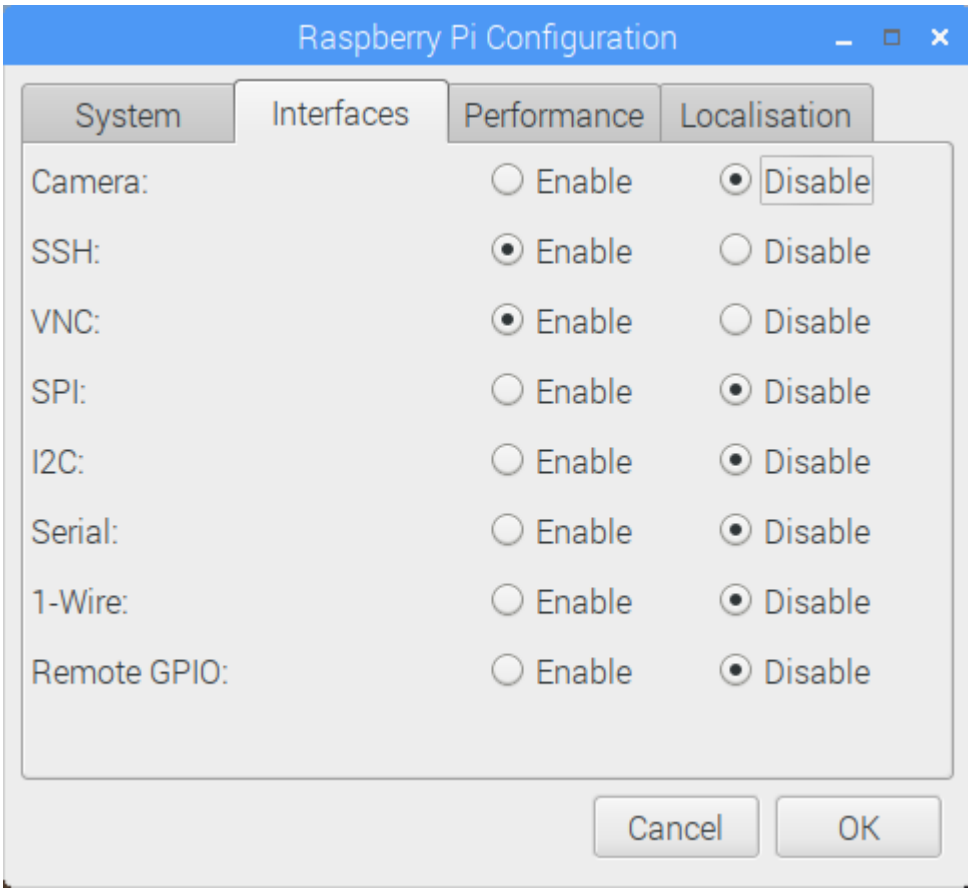


Illustration 4: Raspberry Pi Interfaces Tab

The last tab of "Localization" is where you can set the OS to use "en" for English, Country, and Character Set. CYVA suggests you use "UTF-8" for default. It is recommended that you define the Timezone, Keyboard and the WiFi Country code. Once these base settings are complete the Raspberry Pi will reboot.

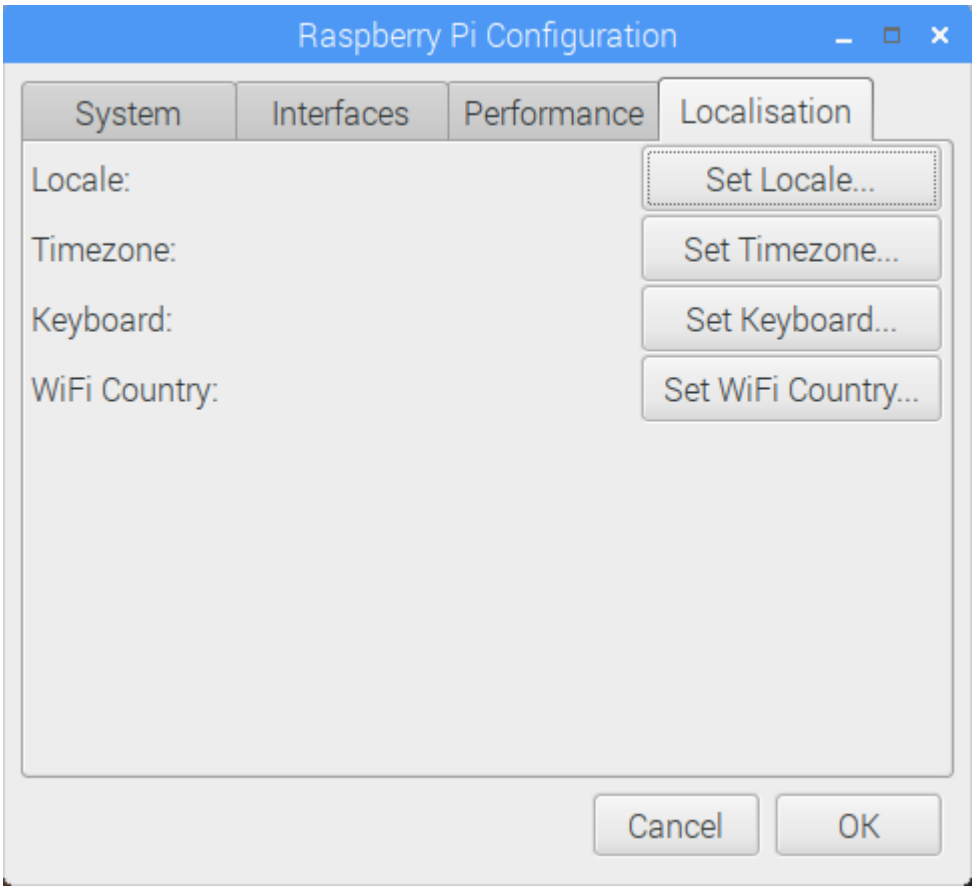


Illustration 5: Raspberry Pi Localisation Tab

Once the reboot is completed, your Raspberry Pi is ready to run.

The MQTT Service is available from Raspbian distribution, however, there are some software requirements. CYVA Research has included software to monitor the performance of the MQTT software on the Raspberry Pi.

The following are the recommended first steps before activating MQTT Service. Starting with a fresh install.

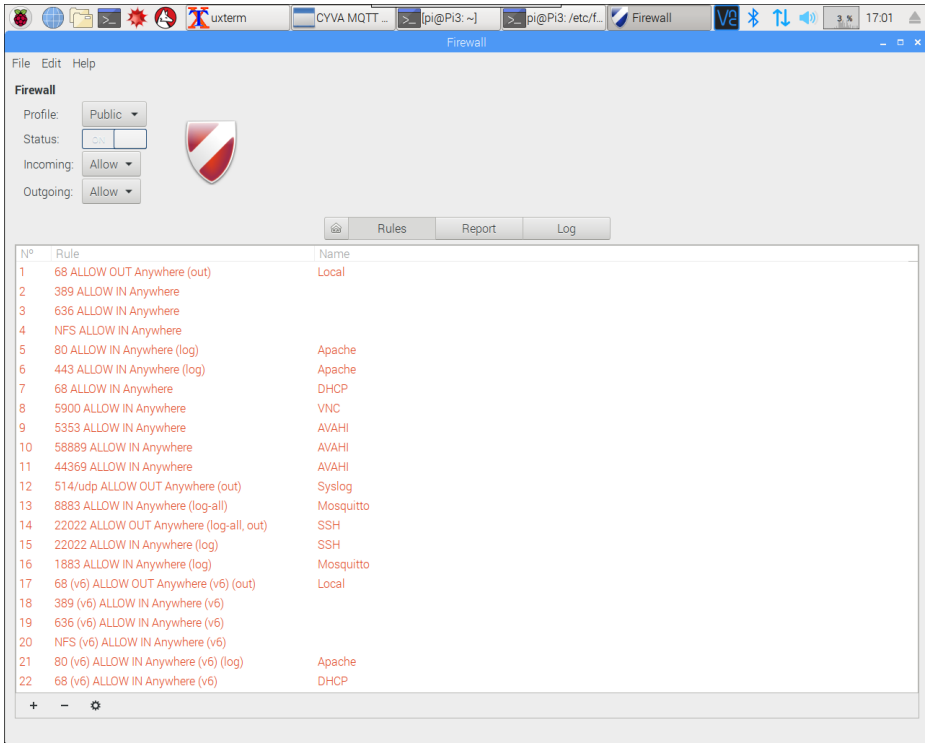
Software Install Steps

- Update and Upgrade Raspbian to the latest release and patches
  - **sudo apt update**
  - **sudo apt upgrade**
- Install and Configure SSH
  - SSH is installed by selecting “ssh” on the Raspberry Pi Interfaces Configuration screen
  - It is recommended that you change the “ssh” daemon port to something different from 22.

- Configure NTP
  - NTP should be installed and you can activate it by entering the following commands.
  - **`sudo timedatectl set-ntp True`**
  - **`timedatectl status`**
- Raspberry Pi Security reference
  - **https://www.raspberrypi.org/documentation/configuration/security.md**
- Install and Configure the following items. UFW installs a command line firewall. GUFW installs a GUI interface for the firewall. The GUI has the ability to export and import firewall profiles. See Appendix A for a example an active firewall profile. Fail2ban, written in Python, is a scanner that examines the log files produced by the Raspberry Pi, and checks them for suspicious activity. It catches things like multiple brute-force attempts to log in, and can inform any installed firewall to stop further login attempts from suspicious IP addresses. It saves you having to manually check log files for intrusion attempts and then update the firewall (via iptables) to prevent them. Xterm is used for autostart of CYVA MQTT Monitoring Service.
  - UFW – Firewall

```
pi@Pi3:~ $ sudo ufw --help
[sudo] password for pi:

Usage: ufw COMMAND

Commands:
 enable                          enables the firewall
 disable                         disables the firewall
 default ARG                     set default policy
 logging LEVEL                   set logging to LEVEL
 allow ARGS                      add allow rule
 deny ARGS                       add deny rule
 reject ARGS                     add reject rule
 limit ARGS                      add limit rule
 delete RULE|NUM                 delete RULE
 insert NUM RULE                 insert RULE at NUM
 route RULE                      add route RULE
 route delete RULE|NUM           delete route RULE
 route insert NUM RULE           insert route RULE at NUM
 reload                          reload firewall
 reset                           reset firewall
 status                          show firewall status
 status numbered                 show firewall status as numbered list of RULES
 status verbose                  show verbose firewall status
 show ARG                        show firewall report
 version                         display version information

Application profile commands:
 app list                        list application profiles
 app info PROFILE                show information on PROFILE
 app update PROFILE              update PROFILE
 app default ARG                 set default application policy

pi@Pi3:~ $
```

  - GUFW – GUI for Firewall



- Fail2Ban – Service Monitoring
- Python Development
  - build-essential
  - python-dev
  - python3-dev
  - xterm
- **sudo apt-get install ufw fail2ban gufw build-essential python-dev python3-dev xterm apache2**
- **sudo ufw enable**
- **sudo ufw status**
- **sudo ufw allow 'WWW Secure'**
- **sudo ufw allow WWW**
- **sudo ufw allow 8883**
- **sudo ufw allow 1883**
- **sudo ufw allow ssh**
- **sudo ufw allow 5900**
- **sudo ufw status**

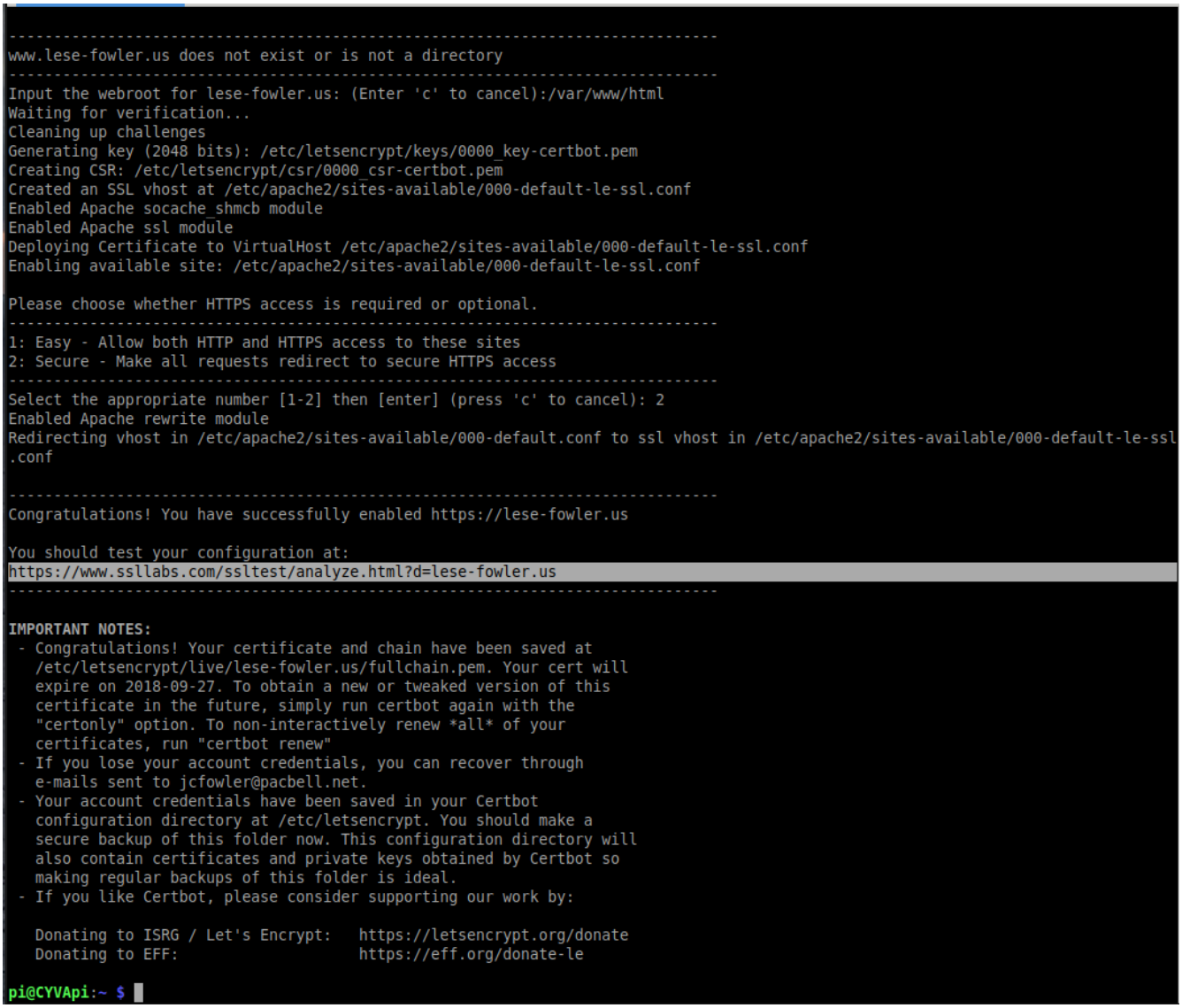
Running the above commands will configure the firewall to allow for HTTP/HTTPS, SSH, VNC, MQTT Clear Text, MQTT Secure with SSL. The resulting firewall status is shown below.

To support SSL communications with the MQTT server you need to install SSL Certificates. There is a free service from Let's Encrypt that will give you a SSL certificate for the fully qualified URI. Details instructions can be found at the following link: https://certbot.eff.org/lets-encrypt/debianstretch-apache

Let's Encrypt leverages a web server every few months to validate the communication and install new SSL certificates. Therefore we need to install Apache2 on the Raspberry Pi. Let's Encrypt requires agreement with their terms and conditions. You can find them at:
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf

- Install and Configure Apache, CertBot
  - **sudo apt-get install apache2 python-certbot-apache**
  - **sudo certbot --authenticator webroot --installer apache**

This will install a SSL certificate for Apache2 and enable SSL. To validate the installed certificate you can use SSLlabs.com. The defined link is highlighted output from the command above and shown below.



- To manually renew a certificate execute the following:
  - **sudo certbot renew --dry-run**

Installing MQTT

It's recommended to follow the full instructions and additional information at (https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-mosquitto-mqtt-messaging-broker-on-debian-8)

Certificates

- **sudo apt update**
- **wget** http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
- **sudo apt-key add mosquitto-repo.gpg.key**
- **sudo apt update**
- **sudo apt-get install mosquitto mosquitto-clients**

```
** Test the agent
           mosquitto_sub -h localhost -t test
           mosquitto_pub -h localhost -t test -m "hello world"

** As of Jun 2018 this is not needed.
      sudo nano /etc/apt/sources.list.d/backports.list
           Add to file
      deb http://mirrors.digitalocean.com/debian jessie-backports main
      sudo apt-get update
```

Special note: Certificates and Firewalls.

To install certificates and gain access to the MQTT from the Internet you need to create pin holes in the firewall that will allow access of the Certbot/Letsencrypt software to function. The basic requirements that need to be in place before any attempt of installing certificates is having a fully qualified domain name in place and pin holes opened within the firewall before the following steps will function correctly.

Dynamic DNS

Dynamic Domain Names and Address will work with the certificates as long as they are properly maintained. However, should your dynamic internet address change, all traffic that is attempting to reach the servers will fail until the dynamic DNS is updated. It is recommended that you follow instructions that are available for most internet routers which understand dynamic DNS so this update can occur rapidly after a change to the internet address of the router.

If the router does not support Dynamic DNS updates, then you can download and install the "ddclient" from SourceForge.net that can be configured to perform an update hourly. The software will check to

see if the address has changed and will only perform the update when required to. There are large number of Dynamic DNS providers available and any will work. Check "ddclient" documentation or code to see what they support. This document has been tested using http://www.dyn.com This code can be installed on the Raspberry Pi.

*Side note: Many home Storage solutions run with Synology NAS software (http://www.synology.com or http://www.readynas.com) which supports Dynamic DNS and can be configured to perform this function instead of the Raspberry Pi.

Firewall Pin Holes

Two items to consider: One Let's Encrypt is free, automated and open. A full description on how it works can be found at (https://letsencrypt.org/getting-started/). Assuming the use of shell access to manage the certificates, the web site directs you to use the Certbot client which is found at (https://certbot.eff.org/). From this screen, select "Apache" and "Debian 9 (stretch)" and it will direct you to the official instructions. I found a few differences when I built this document and they are noted below.

Second: Certbot and MQTT require a few pin holes opened in the firewall to point to the Raspberry Pi which will be hosting the MQTT server. The address of "172.16.10.8" is the address of my local Raspberry Pi MQTT server and I would expect it to be different than what is shown. However, the ports of 80, 443, 1883, 8883 are required for the initial configurations.

NAT/Gaming

Hosted Applications

Service	Ports	Device	Delete
MQTT	TCP/UDP: 1883	172.16.10.8	Delete
MQTT-SSL	TCP/UDP: 8883	172.16.10.8	Delete
HTTP	TCP/UDP: 80	172.16.10.8	Delete
HTTPS	TCP/UDP: 443	172.16.10.8	Delete

Illustration 6: Firewall Pin Holes to Support HTTP/HTTPS/MQTT/ MQTT SSL

Once you have all the software configured and working correctly, it is recommended that you delete the pin hole for the clear text communications of MQTT port 1883. This will not stop you from using port 1883 from your local network, just access from the Internet.

Sample Certificate Install

The following is actual output from a Let's Encrypt install process and comments included.

*** This failed – as the firewall was only letting MQTT traffic into Pi3

sudo certbot certonly --standalone --standalone-supported-challenges http-01 -d mqtt.example.com

```
pi@Pi3:~ $ sudo certbot certonly --standalone --standalone-supported-
challenges http-01 -d Coolwave.Lese-Fowler.US

WARNING: The standalone specific supported challenges flag is deprecated.
Please use the --preferred-challenges flag instead.
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel):jcfowler@pacbell.net

-------------------------------------------------------------------------------
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v01.api.letsencrypt.org/directory
-------------------------------------------------------------------------------
(A)gree/(C)ancel: A
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for coolwave.lese-fowler.us
Waiting for verification...
Cleaning up challenges
Failed authorization procedure. coolwave.lese-fowler.us (http-01): urn:acme:error:unauthorized :: The client lacks sufficient
authorization :: Invalid response from
http://coolwave.lese-fowler.us/.well-known/acme-challenge/iQk6VN40CpoOSnBmmWKkvbqf4Jq9QlUTzHAWx4NeK5A: "<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>html{height:100%}body{margin:0 auto;min-height:600px;min-width:800px"

IMPORTANT NOTES:
 - If you lose your account credentials, you can recover through
   e-mails sent to jcfowler@pacbell.net.
 - The following errors were reported by the server:

   Domain: coolwave.lese-fowler.us
   Type:   unauthorized
   Detail: Invalid response from
   http://coolwave.lese-fowler.us/.well-known/acme-challenge/iQk6VN40CpoOSnBmmWKkvbqf4Jq9QlUTzHAWx4NeK5A:
   "<!DOCTYPE html>
   <html>
   <head>
   <meta charset="utf-8">
   <style>html{height:100%}body{margin:0
   auto;min-height:600px;min-width:800px"

   To fix these errors, please make sure that your domain name was
   entered correctly and the DNS A record(s) for that domain
   contain(s) the right IP address.
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
```

Ran again with hole in firewall for 80/443 to hit the Raspberry Pi.

```
pi@Pi3:~ $ sudo certbot certonly --standalone --standalone-supported-
challenges http-01 -d Coolwave.Lese-Fowler.US
WARNING: The standalone specific supported challenges flag is deprecated.
Please use the --preferred-challenges flag instead.
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for coolwave.lese-fowler.us
Waiting for verification...
Cleaning up challenges
Generating key (2048 bits): /etc/letsencrypt/keys/0000_key-certbot.pem
```

```
Creating CSR: /etc/letsencrypt/csr/0000_csr-certbot.pem

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/coolwave.lese-fowler.us/fullchain.pem. Your
   cert will expire on 2018-04-27. To obtain a new or tweaked version
   of this certificate in the future, simply run certbot again. To
   non-interactively renew *all* of your certificates, run "certbot
   renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

pi@Pi3:~ $
```

Add the following to crontab on the Raspberry Pi to support automatic Certificate Renewals.

sudo crontab -e

15 3 * * * certbot renew --noninteractive --post-hook "systemctl restart mosquitto"

To enable user id and password pairing on the MQTT messages, then create a password file for Mosquitto. Details can be found at the Eclipse Mosquitto Passwd man page at (https://mosquitto.org/man/mosquitto_passwd-1.html) Syntax: sudo mosquitto_passwd -b *passwordfile username password*

sudo mosquitto_passwd -b /etc/mosquitto/passwd kevin kevin
sudo nano /etc/mosquitto/conf.d/default.conf

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Restart Mosquitto to have these changes take affect.

sudo systemctl restart mosquitto

To activate the SSL Certificate on the MQTT server perform the following changes. Please note the certificate file shown below can be found in the output above. **Do not use what is typed here, it will not work for you!

sudo nano /etc/mosquitto/conf.d/default.conf

```
listener 1883 localhost
listener 8883
certfile /etc/letsencrypt/live/coolwave.lese-fowler.us/cert.pem
cafile /etc/letsencrypt/live/coolwave.lese-fowler.us/chain.pem
keyfile /etc/letsencrypt/live/coolwave.lese-fowler.us/privkey.pem
```

We're adding two separate listener blocks to the config. The first, listener 1883 localhost, updates the default MQTT listener on port1883, which is what we've been connecting to so far. 1883 is the standard unencrypted MQTT port. The localhost portion of the line instructs Mosquitto to only bind

this port to the localhost interface, so it's not accessible externally. External requests would have been blocked by our firewall anyway, but it's good to be explicit.

Listener 8883 sets up an encrypted listener on port 8883. This is the standard port for MQTT + SSL, often referred to as MQTTS. The next three lines, certfile, cafile, and keyfile, all point Mosquitto to the appropriate Let's Encrypt files to set up the encrypted connections.

```
 $ sudo ufw allow 8883
Skipping adding existing rule
Skipping adding existing rule (v6)
pi@Pi3:~ $ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
389                        ALLOW       Anywhere
636                        ALLOW       Anywhere
NFS                        ALLOW       Anywhere
68                         ALLOW       Anywhere
5900                       ALLOW       Anywhere
5353                       ALLOW       Anywhere
58889                      ALLOW       Anywhere
44369                      ALLOW       Anywhere
22022                      ALLOW       Anywhere                   (log)
172.16.10.8 1883           ALLOW       Anywhere                   (log)
8883                       ALLOW       Anywhere
389 (v6)                   ALLOW       Anywhere (v6)
636 (v6)                   ALLOW       Anywhere (v6)
NFS (v6)                   ALLOW       Anywhere (v6)
68 (v6)                    ALLOW       Anywhere (v6)
5900 (v6)                  ALLOW       Anywhere (v6)
5353 (v6)                  ALLOW       Anywhere (v6)
58889 (v6)                 ALLOW       Anywhere (v6)
44369 (v6)                 ALLOW       Anywhere (v6)
22022 (v6)                 ALLOW       Anywhere (v6)              (log)
8883 (v6)                  ALLOW       Anywhere (v6)

68                         ALLOW OUT   Anywhere
514/udp                    ALLOW OUT   Anywhere
22022                      ALLOW OUT   Anywhere                   (log-all)
68 (v6)                    ALLOW OUT   Anywhere (v6)
514/udp (v6)               ALLOW OUT   Anywhere (v6)
22022 (v6)                 ALLOW OUT   Anywhere (v6)              (log-all)
```

Perform some tests using SSL. The items in Red should be changed to values entered above.

```
mosquitto_pub -h Coolwave.Lese-Fowler.US -t test -m "hello again" -p 8883 --
capath /etc/ssl/certs/ -u "kevin" -P "kevin"
```

Note *that we're using the full hostname instead of localhost. Because our SSL certificate is issued for Coolwave.lese-fowler.us which is aliased to Pi3.Lese-Fowler.us. Coolwave.Lese-Fowler.us is the public name of the firewall which is registered at http://www.dyn.com, if an attempt is made using secure connection to localhost we'll get an error saying the hostname does not match the certificate hostname (even though they both point to the same Mosquitto server).*

Additionally:

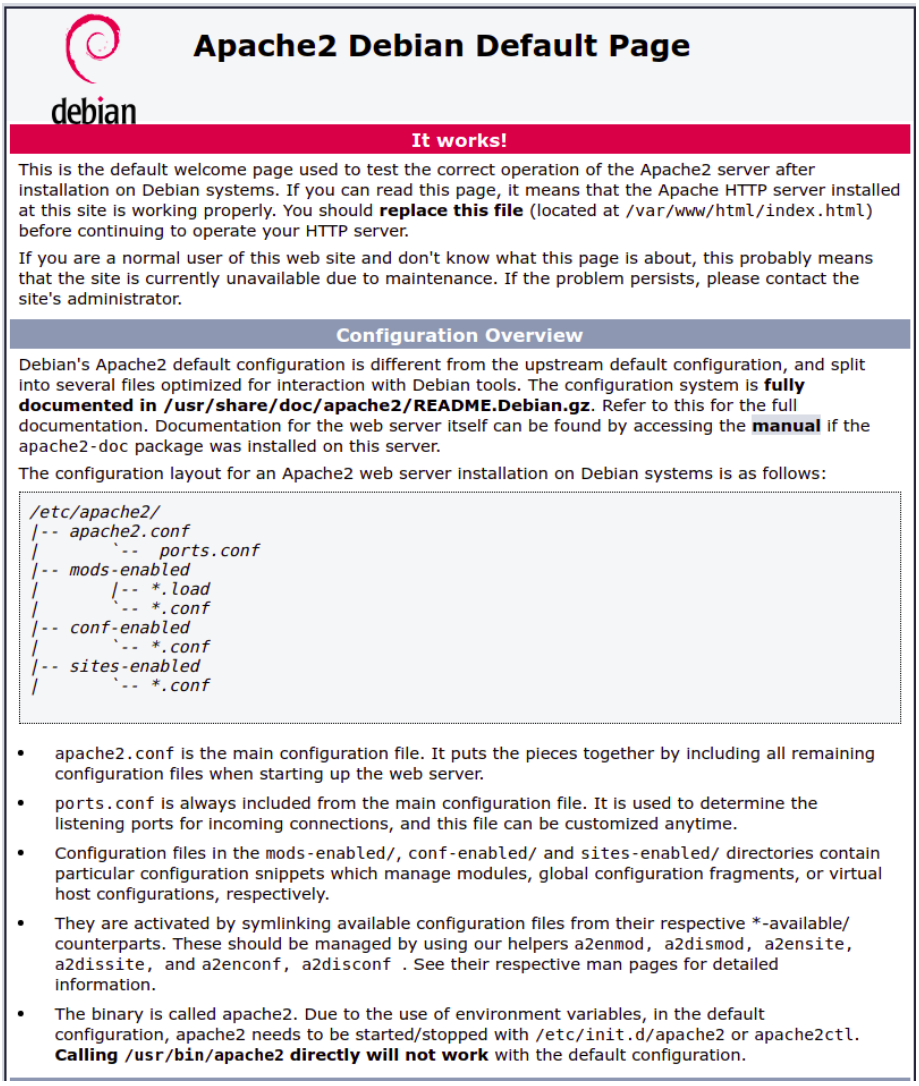
--capath /etc/ssl/certs/ enables SSL for mosquitto_pub, and tells it where to look for root certificates. These are typically installed by your operating system, so the path is different for macOS, Windows, etc. mosquitto_pub uses the root certificate to verify that the Mosquitto server's certificate was properly signed by

the Let's Encrypt certificate authority. It's important to note that mosquitto_pub and mosquitto_sub will not attempt an SSL connection without this option (or the similar --cafile option), even if you're connecting to the standard secure port of 8883.

If all goes well with the test, you'll see hello again show up in the other mosquitto_sub terminal. This means your server is fully set up! If you'd like to extend the MQTT protocol to work with websockets, you can follow the final step.

Apache

Apache2 is required by Let's Encrypt to generate and test the certificates. Therefore you need a minimum web site. I would recommend that you verify that no php or any other executable code is within the Apache website directory tree. Apache will install a default page of the following:



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

I would recommend that you change the index.html and redirect people away from the website as the web-server is only used for the Certbot and Certificate validation.

There is a section on Raspberry Pi that describes how to install Apache2.
https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md

Securing Apache

I would recommend tightening up Apache2 security and there are many sites and books available that discuss the the topic. One site that has a good set up recommendations is from TecMint.com.
https://www.tecmint.com/apache-security-tips/

Installing from Image

TBD

~~*If from Image:*~~

- ~~Download the CYVA MQTT Image~~
- ~~Write image to MicroSD Card~~
- ~~Boot Node~~
  - ~~Node will boot looking for a DHCP on the Ethernet port~~
  -

Appendix A

The UFW Active Profile should look something like the following export from a running CYVA MQTT Server. This file is an export from the GUFW application.

```
[fwBasic]
status = enabled
incoming = allow
outgoing = allow
routed = disabled

[Rule0]
ufw_rule = 68 ALLOW OUT Anywhere (out)
description = Local
command = /usr/sbin/ufw allow out from any to any port 68
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 68
iface =
routed =
logging =

[Rule1]
ufw_rule = 80 ALLOW IN Anywhere (log)
description = Apache
command = /usr/sbin/ufw insert 6 allow in log from any to any port 80
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 80
iface =
routed =
logging = log

[Rule2]
ufw_rule = 443 ALLOW IN Anywhere (log)
description = Apache
command = /usr/sbin/ufw insert 6 allow in log from any to any port 443
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 443
iface =
routed =
logging = log

[Rule3]
ufw_rule = 68 ALLOW IN Anywhere
description = DHCP
command = /usr/sbin/ufw allow in from any to any port 68
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 68
iface =
routed =
logging =
```

```
[Rule4]
ufw_rule = 5900 ALLOW IN Anywhere
description = VNC
command = /usr/sbin/ufw allow in from any to any port 5900
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 5900
iface =
routed =
logging =

[Rule5]
ufw_rule = 5353 ALLOW IN Anywhere
description = AVAHI
command = /usr/sbin/ufw allow in from any to any port 5353
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 5353
iface =
routed =
logging =

[Rule6]
ufw_rule = 58889 ALLOW IN Anywhere
description = AVAHI
command = /usr/sbin/ufw allow in from any to any port 58889
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 58889
iface =
routed =
logging =

[Rule7]
ufw_rule = 44369 ALLOW IN Anywhere
description = AVAHI
command = /usr/sbin/ufw allow in from any to any port 44369
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 44369
iface =
routed =
logging =

[Rule8]
ufw_rule = 514/udp ALLOW OUT Anywhere (out)
description = Syslog
command = /usr/sbin/ufw allow out proto udp from any to any port 514
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 514/udp
iface =
routed =
logging =

[Rule9]
```

```
ufw_rule = 8883 ALLOW IN Anywhere (log-all)
description = Mosquitto
command = /usr/sbin/ufw insert 12 allow in log-all from any to any port 8883
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 8883
iface =
routed =
logging = log-all

[Rule10]
ufw_rule = 22022 ALLOW OUT Anywhere (log-all, out)
description = SSH
command = /usr/sbin/ufw allow out log-all from any to any port 22022
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 22022
iface =
routed =
logging = log-all

[Rule11]
ufw_rule = 22022 ALLOW IN Anywhere (log)
description = SSH
command = /usr/sbin/ufw allow in log from any to any port 22022
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 22022
iface =
routed =
logging = log

[Rule12]
ufw_rule = 1883 ALLOW IN Anywhere (log)
description = Mosquitto
command = /usr/sbin/ufw allow in log from any to any port 1883
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 1883
iface =
routed =
logging = log

[Rule13]
ufw_rule = 68 (v6) ALLOW OUT Anywhere (v6) (out)
description = Local
command = /usr/sbin/ufw allow out from any to any port 68
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 68
iface =
routed =
logging =

[Rule14]
ufw_rule = 80 (v6) ALLOW IN Anywhere (v6) (log)
```

```
description = Apache
command = /usr/sbin/ufw insert 6 allow in log from any to any port 80
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 80
iface =
routed =
logging = log

[Rule15]
ufw_rule = 68 (v6) ALLOW IN Anywhere (v6)
description = DHCP
command = /usr/sbin/ufw allow in from any to any port 68
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 68
iface =
routed =
logging =

[Rule16]
ufw_rule = 443 (v6) ALLOW IN Anywhere (v6) (log)
description = Apache
command = /usr/sbin/ufw insert 6 allow in log from any to any port 443
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 443
iface =
routed =
logging = log

[Rule17]
ufw_rule = 5900 (v6) ALLOW IN Anywhere (v6)
description = VNC
command = /usr/sbin/ufw allow in from any to any port 5900
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 5900
iface =
routed =
logging =

[Rule18]
ufw_rule = 5353 (v6) ALLOW IN Anywhere (v6)
description = AVAHI
command = /usr/sbin/ufw allow in from any to any port 5353
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 5353
iface =
routed =
logging =

[Rule19]
ufw_rule = 58889 (v6) ALLOW IN Anywhere (v6)
description = AVAHI
```

```
command = /usr/sbin/ufw allow in from any to any port 58889
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 58889
iface =
routed =
logging =

[Rule20]
ufw_rule = 44369 (v6) ALLOW IN Anywhere (v6)
description = AVAHI
command = /usr/sbin/ufw allow in from any to any port 44369
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 44369
iface =
routed =
logging =

[Rule21]
ufw_rule = 514/udp (v6) ALLOW OUT Anywhere (v6) (out)
description = Syslog
command = /usr/sbin/ufw allow out proto udp from any to any port 514
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 514/udp
iface =
routed =
logging =

[Rule22]
ufw_rule = 22022 (v6) ALLOW OUT Anywhere (v6) (log-all, out)
description = SSH
command = /usr/sbin/ufw allow out log-all from any to any port 22022
policy = allow
direction = out
protocol =
from_ip =
from_port =
to_ip =
to_port = 22022
iface =
routed =
logging = log-all

[Rule23]
ufw_rule = 22022 (v6) ALLOW IN Anywhere (v6) (log)
description = SSH
command = /usr/sbin/ufw allow in log from any to any port 22022
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 22022
iface =
routed =
logging = log

[Rule24]
ufw_rule = 1883 (v6) ALLOW IN Anywhere (v6) (log)
description = Mosquitto
command = /usr/sbin/ufw allow in log from any to any port 1883
```

```
policy = allow
direction = in
protocol =
from_ip =
from_port =
to_ip =
to_port = 1883
iface =
routed =
logging = log
```

Appendix B: