Developer Study Guide

Bluetooth® Internet Gateways

Security

Release	:	2.0.1
Document Version:		2.0.0
Last updated	:	24th June 2021

Contents

1. Revision History

Version	Date	Author	Changes
1.0.0	18th November 2020	Martin Woolley Bluetooth SIG	Initial version
1.0.1	6th January 2021	Martin Woolley Bluetooth SIG	Added *enabled* configuration property to the Bluetooth firewall.
2.0.0	24th June 2021	Martin Woolley Bluetooth SIG	**Release** Added new module covering gateways for Bluetooth mesh networks. Modularised the study guide to accommodate the two main cases of LE Peripherals and mesh networks. Updated to be based on Python 3 rather than Python 2. **Document** This document is new in this release

2. Introduction

This module covers the topic of security as it applies to Bluetooth internet gateways. It applies to both gateways for LE Peripheral devices and gateways for Bluetooth mesh networks since many of the issues are the same. Where an issue only applies to one of these gateway types, this is made clear in the text.

3. Security

3.1 Security Review

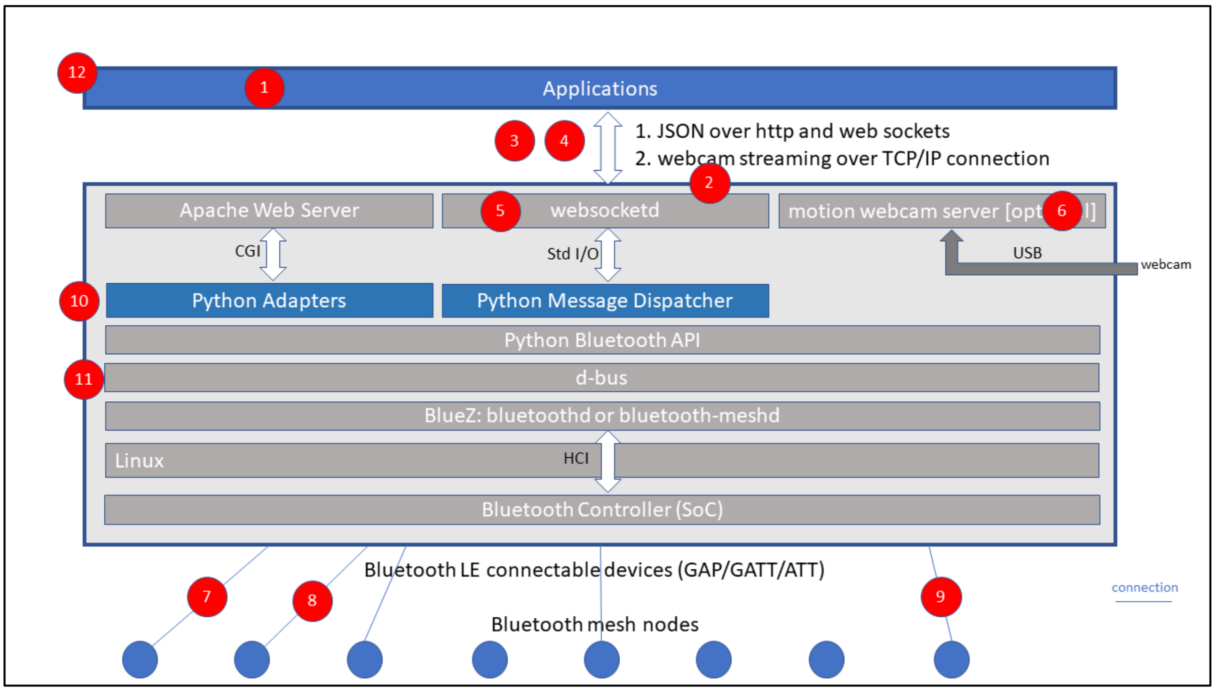
We will start by appraising the security of the gateways implemented but not yet secured in modules 03 and 04. This will generate a list of issues to be addressed. We'll then consider how to address each issue identified.

3.1.1 Task - Security Issues List

Write a list of any security concerns or vulnerabilities that you can identify the gateway(s) as having. Then move to the next page and compare your conclusions with those presented.

3.1.2 Gateway Security Analysis

Figure 1 shows a labelled diagram of the gateway architecture, with each label indicating a security issue.



Applications
1. JSON over http and web sockets
2. webcam streaming over TCP/IP connection
Apache Web Server
websocketd
motion webcam server [optional]
CGI
Std I/O
USB
webcam
Python Adapters
Python Message Dispatcher
Python Bluetooth API
d-bus
BlueZ: bluetoothd or bluetooth-meshd
Linux
HCI
Bluetooth Controller (SoC)
Bluetooth LE connectable devices (GAP/GATT/ATT)
Bluetooth mesh nodes
connection
1
2
3
4
5
6
7
8
9
10
11
12

Figure 1 - Security Analysis

Table 1 lists the issues indicated by each label. The sequencing of issues is not significant. Generally, the point of view taken assumes the gateway is connected to the internet.

Table 1 - Security Issues List

Ref	Short Name	Issue
1	No user authentication or access control	There is no mechanism requiring users of the gateway to authenticate and therefore no control over who may or may not access the gateway and the remote devices it exposes.
2	TCP/IP port visibility	There are no controls in place to prevent access to services running on the gateway which listen on TCP/IP ports. If the gateway is accessible on a TCP/IP network, any other TCP/IP device can connect to any of the available ports.
3	Data over TCP/IP is not private	Data which passes between a gateway application and the gateway, using HTTP and web sockets over TCP/IP is not encrypted.
4	Multiple TCP/IP ports in use	The gateway uses three different TCP/IP ports to provide its services; 80 for HTTP, 8081 for the webcam streaming service, *motion* and 8082 for websocketd. The more ports exposed directly to the internet, the greater the security risk.

5	websocketd is directly accessible	Related to (4), websocketd can be connected to directly from the internet.
6	motion is directly accessible	Related to (4), motion can be connected to directly from the internet.
7	Bluetooth data in flight is not necessarily private	A secure gateway should include measures to ensure communication between the gateway and Bluetooth devices is private and protected from eavesdroppers.
8	Not all Bluetooth devices should be accessible	There needs to be a way of restricting access via the gateway to only specific devices. Allowing unrestricted access to any and all discovered devices might be appropriate in some circumstances but in our case, we take the view that this is undesirable.
9	Not all Bluetooth device capabilities should be available	Bluetooth LE devices have many capabilities, and these capabilities manifest themselves as GATT services, characteristics and descriptors. Bluetooth mesh nodes have capabilities that are provided by the choice of mesh models that they implement. There needs to be a way of controlling which of these capabilities is available via the gateway and which ones are not. For example, consider services and characteristics concerned with device firmware updates. We may take the view that we do not want to expose the ability to change device firmware over the internet, even with other security measures in place.
10	Physical access	The gateway computer must be physically secure so that unauthorised personnel may not tamper with it. For example, in the case of a Raspberry Pi, simply swapping the SD card would allow a compromised version of the gateway to be substituted.
11	Gateway server login control	Access to the gateway by logging in via SSH, VNC or other service must be strictly controlled.
12	User access rights	Should all users have the same rights? Should all users be able to access all devices that the gateway exposes and perform all permitted actions? This is a question about requirements.

3.1.3 Task - Countermeasures

Write a list of actions you think could be taken which would act as countermeasures to the issues listed in 3.1.2. Think about either LE Peripheral devices, Bluetooth mesh nodes or both, depending on your area of interest. Move on and review our list when you have spent some time thinking about this.

3.1.4 Countermeasures to be Taken

Table 2 lists the actions we will take to address the security issues identified above. These are generally applicable actions that apply to either of the two gateway types considered in this study guide. Modules 03 and 04 contain further information on security measures that apply to either the LE Peripherals gateway or the Bluetooth mesh network gateway only.

Table 2 - Countermeasures

Ref	Short Name	Countermeasure
1	No user authentication or access control	Access to the gateway API will be granted to a list of specific users and each will be issued with a set of credentials. The gateway API will require users to authenticate before being able to exercise the API. The Apache web server supports a number of authentication schemes. It is beyond the scope of this study guide to assess each one. We will use Basic Auth over SSL which is a scheme which requires users to enter a user ID and password and for these credentials to be transferred over an encrypted TCP/IP link to the gateway. For a production gateway, you may decide to use a more secure authentication scheme such as a two-factor approach.
2	TCP/IP port visibility	The gateway computer will have an IP firewall installed on it. It will be configured to allow access to only those ports which must be accessible for the gateway to be usable.
3	Data over TCP/IP is not private	The gateway API will only be available over the encrypted HTTPS protocol. As such, port 443 will be used rather than port 80.
4	Multiple TCP/IP ports in use	We will use a technique called *reverse proxying* to hide all except one of the three ports used by the gateway. Port 443 will handle all communication from gateway applications and redirect to the appropriate gateway service (Apache web server, websocketd or motion) using a URL matching scheme.
5	websocketd is directly accessible	websocketd will not be available directly from the internet. Port 8082 will be blocked by a firewall and reverse proxying (see 4) will be used to allow access via port 443.
6	motion is directly accessible	motion will not be available directly from the internet. Port 8081 will be blocked by a firewall and reverse proxying (see 4) will be used to allow access via port 443.
7	Bluetooth data in flight is not necessarily private	See module 03 for the countermeasure(s) for LE Peripherals. See module 04 for the countermeasure(s) for mesh nodes.

8	Not all Bluetooth devices should be accessible	A software service or component which restricts access to particular IP addresses and ports is called a firewall. We need something which serves an analogous purpose for our Bluetooth devices and the capabilities that they have. We will design and add a software component we shall refer to as a *Bluetooth firewall* which will provide this set of capabilities.
9	Not all Bluetooth device capabilities should be available	See module 03 for the countermeasure(s) for LE Peripherals. See module 04 for the countermeasure(s) for mesh nodes.
10	Physical access	The gateway computer must be installed in a secure room with physical access control mechanisms.
11	Gateway server login control	Access to the gateway using SSH and VNC will be allowed but controlled via standard Linux account authentication mechanisms. Only the gateway system administrator will have root access. For the purposes of this study guide, you are the system administrator.
12	User access rights	We will take the view that all users have the same rights when using the gateway via its API. Therefore, we do not need to take any action relating to this issue.

3.2 Generally Applicable Security Countermeasures Implementation

3.2.1 Task - Securing the Raspberry Pi with a Firewall

Your next task is to install and configure an IP firewall on your Raspberry Pi. We will use a Linux firewall called *ufw* which stands for Uncomplicated Firewall.

Our requirement is to allow access to the following TCP/IP ports only:

```
Port 443 for all gateway client API requests and responses over HTTPS
Port 22 for SSH from the local area network only
Port 5900 for VNC from the local area network only
```

Follow these steps to install and configure a firewall. The examples used here assume that the local area network (LAN) uses IP addresses in the range 192.168.0.n and that there is a router connecting the LAN to the internet with IP address 192.168.0.1. You may need to substitute different values, depending on the address range your LAN uses.

Warning - take care with this process. If you make a mistake, you may find that you are unable to access your Raspberry Pi via the network. If this happens, you will need to connect a screen, keyboard and mouse directly to your Raspberry Pi and fix your firewall configuration.

```
# 1. install the firewall
sudo apt install ufw

# 2. allow access to port 443
sudo ufw allow 443
```

```
#3. allow access to ports 22 and 5900 from the LAN only
sudo ufw allow from 192.168.0.0/24 to any port 22
sudo ufw allow from 192.168.0.0/24 to any port 5900

# 4. deny access to SSH and VNC from the network's router (i.e. from the internet)
sudo ufw deny from 192.168.0.1 to any port 22
sudo ufw deny from 192.168.0.1 to any port 5900

# 5. enable the firewall rules
sudo ufw enable
```

In step 3, we are ensuring that ssh and VNC may only be accessed by machines on the LAN. In step 4 we explicitly block traffic to these ports from the router. We shall discuss router configuration later in this section and you will appreciate that it should not be possible for such traffic to originate from the router. We're playing safe with a "belt and braces" approach here though.

Run the command *sudo ufw status* and review the output. It should look something like this:

```
pi@raspberrypi:~ $ sudo ufw status
Status: active

To                           Action      From
--                           ------      ----
443                          ALLOW       Anywhere
22                           DENY        192.168.0.1
5900                         DENY        192.168.0.1
22                           ALLOW       192.168.0.0/24
5900                         ALLOW       192.168.0.0/24
443 (v6)                     ALLOW       Anywhere (v6)
```

3.2.2 Task - Securing the Apache Web Server

To secure the Apache web server, we need to take a number of actions.

3.2.2.1 HTTPS Only

We need to enable HTTPS access to the Apache web server and block access to unencrypted HTTP so that all gateway API traffic is handled by HTTPS and is therefore, encrypted. HTTPS requires an SSL certificate to be installed. For a production gateway, you should acquire a certificate from a certificate authority. For our educational gateway, we will generate a self-signed certificate.

Run these commands:

```
sudo mkdir /etc/apache2/ssl
sudo mkdir /etc/openssl
sudo vi /etc/openssl/openssl.conf
```

In the newly created openssl.conf file, add entries such as the following (you may want to vary your values).

```
[req]
default_bits = 2048
prompt = no
default_md = sha256
distinguished_name = dn
req_extensions        = req_ext
x509_extensions       = x509_ext

[dn]
C = UK
ST = Surrey
L = London
O = Your Organisation
CN = raspberrypi
```

```
[v3_req]
subjectAltName = @alt_names

[alt_names]
DNS.1 = squirtle

[ req_ext ]
subjectKeyIdentifier   = hash
basicConstraints       = CA:FALSE
keyUsage               = digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth, clientAuth
subjectAltName         = @alt_names
nsComment              = "Self-Signed SSL Certificate"

[ x509_ext ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints       = CA:FALSE
keyUsage               = digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth, clientAuth
subjectAltName         = @alt_names
nsComment              = "Self-Signed SSL Certificate"
```

Note: Web browsers treat self-signed certificates with caution and at best require the user to acknowledge the risk of proceeding to a web site which is secured with a certificate that was not issued by a trusted certificate authority (CA). The openssl configuration specified above is sufficient for Google Chrome version 91.0.4472.77 to permit the site to be visited after the user has accepted the risk. Other browsers may not trust this certificate under any circumstance. If you do not wish to use Chrome with this study guide you will need to research how to create a self-signed certificate that your chosen browser will sufficiently trust or use a CA issued certificate.

Generate your SSL certificate by running this command:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt -config
/etc/openssl/openssl.conf
```

Make sure the certificate and key files are readable by users other than root:

```
sudo chmod 755 /etc/apache2/ssl/apache.*
```

Next, enable the Apache SSL module:

```
sudo a2enmod ssl
sudo systemctl restart apache2
```

Edit the apache SSL configuration file and set these properties to the values shown:

```
sudo vi /etc/apache2/sites-available/default-ssl.conf

SSLEngine on
SSLCertificateFile    /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

Activate this configuration by creating a symlink to it in the sites-enabled directory:

```
sudo ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-enabled/000-
default-ssl.conf
```

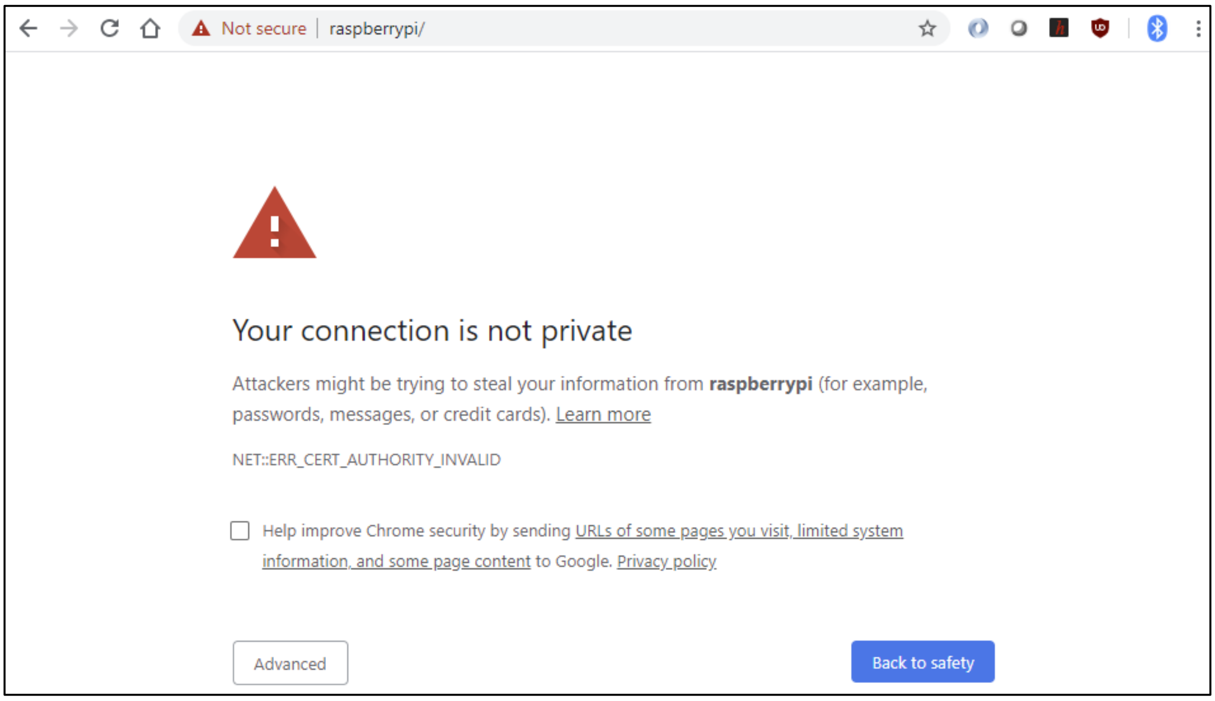
Stop apache from listening on port 80 by commenting out the *Listen 80* directive so that only port 443 is available.

```
sudo vi /etc/apache2/ports.conf

# Comment out this property:
# Listen 80

# Ensure the configuration includes this entry
<IfModule ssl_module>
        Listen 443
</IfModule>
```

Restart apache with `sudo service apache2 restart`

Launch a browser on a desktop machine and try to access the gateway default home page with the HTTP URL http://raspberrypi/ . This should fail and the browser should support something like ERR_CONNECTION_TIMED_OUT. This is because the apache web server is no longer listening on port 80.

Now try to access using URL https://raspberrypi/ . This should work but you will be warned by your browser of a security issue. Chrome shows this page:



Your connection is not private

Attackers might be trying to steal your information from **raspberrypi** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Chrome security by sending URLs of some pages you visit, limited system information, and some page content to Google. Privacy policy

This is because your self-signed certificate is not trusted by the browser. Choose the Advanced button and opt to proceed. Alternatively, you can export your self-signed certificate and configure your browser to trust it. There are instructions on how to do this at various internet sites, including this one: https://www.pico.net/kb/how-do-you-get-chrome-to-accept-a-self-signed-certificate

The Apache web server now supports HTTPS and can only be accessed via HTTPS.

3.2.2.2 Encrypted web sockets communication

We need to run the websocketd daemon in such a way that it will use SSL for encryption. The script to be used with websocketd varies depending on whether the gateway works with LE Peripherals or mesh nodes. See modules 03 and 04 for details.

3.2.2.3 Authentication

Execute each of the following steps to set up authentication and access control on your gateway:

```
# 1. Create a set of user credentials using htpasswd. Choose your own user name!
pi@raspberrypi:~ $ sudo htpasswd -c -B /etc/apache2/auth.users martin
New password:
Re-type new password:
Adding password for user martin
```

Note that the -c option is only required the first time htpasswd is run. It creates the credentials file.

Now secure each web application directory by following the steps described in modules 03 for LE Peripherals or 04 for the gateway for Bluetooth mesh nodes and then continue here.

3.2.2.4 Reverse Proxying

Reverse proxying is a technique which allows access to system services listening on a variety of TCP/IP ports to take place over a single port, with a server acting as a proxy, forwarding requests to the appropriate service and on the correct port by matching URL patterns. Figure 2 illustrates the basic idea of reverse proxying using the *Explore Blue* application used with the LE Peripherals gateway from module 03 as an example.

The benefit of reverse proxying is that we reduce the number of ports that we need to have open to the internet, which reduces risk and simplifies firewall management.

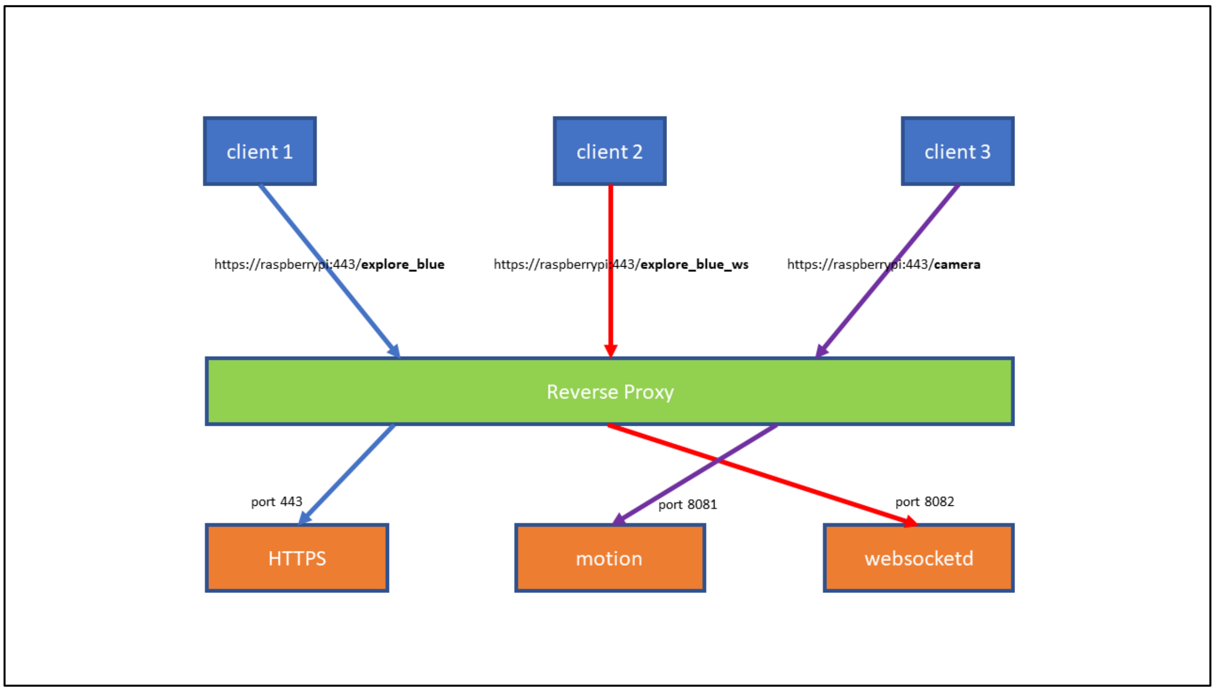

client 1
client 2
client 3
https://raspberrypi:443/explore_blue
https://raspberrypi:443/explore_blue_ws
https://raspberrypi:443/camera
Reverse Proxy
port 443
port 8081
port 8082
HTTPS
motion
websocketd

Figure 2 - Reverse Proxying

The Apache web server is able to act as a reverse proxy. Your next task is to configure reverse proxying and make adjustments to the two applications so that they use modified URLs for webcam streaming and web sockets communication.

For Apache to act as a reverse proxy for both HTTP and web sockets communication requires three modules to be enabled and configured. Run the following commands to enable the required modules:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod proxy_wstunnel
```

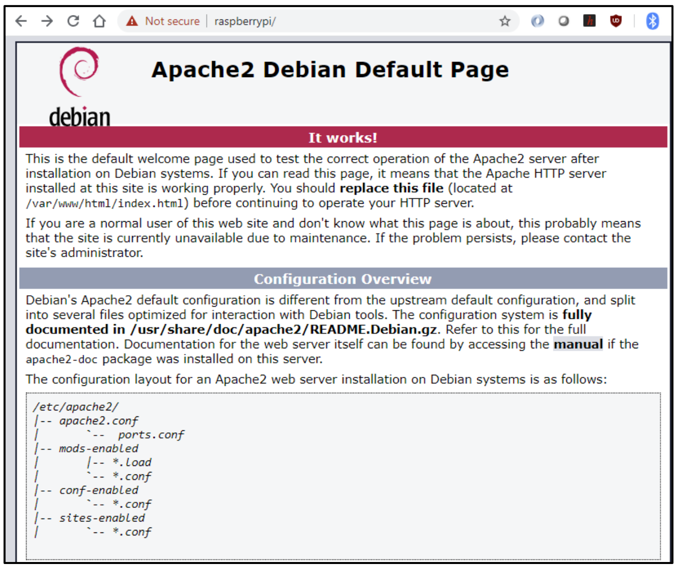
These two commands should have resulted in these modules being enabled. Check with the following command:

```
pi@raspberrypi:~ $ sudo apache2ctl -t -D DUMP_MODULES|grep proxy
 proxy_module (shared)
 proxy_http_module (shared)
 proxy_wstunnel_module (shared)
```

Further steps required for each of the LE Peripherals and mesh gateway types are documented in modules 03 and 04.

3.2.2.6 Disabling the home page

The default home page provides information about the web server running on the gateway. It makes sense to block this information by forbidding access to the home page directory or changing the default page in /var/www/html to something that offers no clues about the web server.



Apache2 Debian Default Page

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.

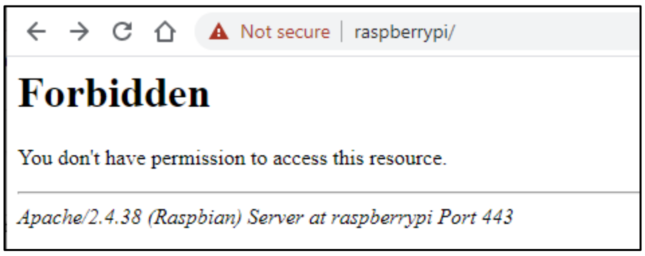
The configuration layout for an Apache2 web server installation on Debian systems is as follows:

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

Edit /etc/apache2/sites-enabled/000-default-ssl.conf and add an additional Directory configuration block to disable access to the default home page.

```
<Directory "/var/www/html">
    Require all denied
</Directory>
```

Restart the apache service to make this setting active. Try to access the default page again and you should see:



Forbidden

You don't have permission to access this resource.

Apache/2.4.38 (Raspbian) Server at raspberrypi Port 443

3.2.3 Task - Securing the Bluetooth Devices

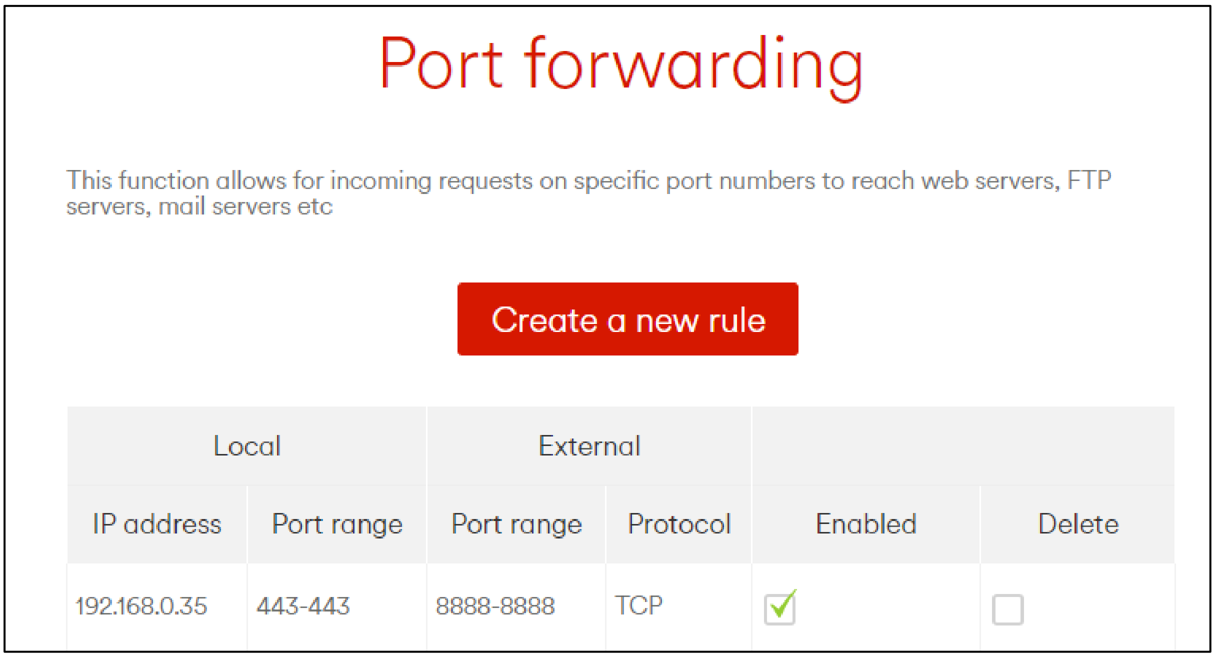
To secure communication with and access to Bluetooth devices via the gateway requires steps that vary according to the type of gateway. Modules 03 and 04 provide details for the two gateway types. Follow the steps that relate to your current project now.

3.2.4 Task - Enable access via the external router

Your gateway should now be secure with respect to the list of issues identified earlier. At this stage though, it is almost certainly only accessible from machines connected to the same local area network. What might be involved in making it available from the internet?

There are a number of ways in which this issue could be approached, depending on the security infrastructure on your network and any IT policies which might apply. A full discussion is beyond the scope of this study guide. However, if you are able to use your gateway on your home network or if you happen to have a very good relationship with your local systems administrator, you may be able to allow access to the gateway through whatever external firewall you might have in place.

Home networks often sit behind a broadband router. Routers often support a capability called *port forwarding* which allows you to configure the router to forward any packets received from the internet which are addressed to a particular TCP/IP port, to a machine with a specific IP address on your LAN. Figure 3 shows an example port forwarding router configuration. Any TCP/IP packets addressed to port 8888 are forwarded to the machine with IP address 192.168.0.35 on port 443. And it so happens that this machine is a Raspberry Pi running a Bluetooth internet gateway.


Port forwarding

This function allows for incoming requests on specific port numbers to reach web servers, FTP servers, mail servers etc

Local		External			
IP address	Port range	Port range	Protocol	Enabled	Delete
192.168.0.35	443-443	8888-8888	TCP	✓	☐

Figure 3 - Port Forwarding in a router configuration page

4. Close

Further security measures may still need to be applied to your gateway. Return now to module 03 or 04 and complete the remainder of those modules.