Using Let's Encrypt As A Certificate Authority on a Raspberry Pi

Mark G.

Mar. 23, 2019

A demonstration is presented on how to provide a web server with a certificate, for Transport Layer Security (TLS), using Let's Encrypt as a Certificate Authority (CA). The demo uses a Raspberry Pi 3B (RPi) running the Raspbian operating system. The web server used is Apache. The server has a static IP version 4 address and a domain name system (DNS) name of www.palaceofretention.ca. A configuration from scratch will be described, allowing the reader to learn how to initially setup their system in a secure manner.

Note that the securing of Apache beyond the encrypted transmission of web pages is not provided. Once an arbitrary web framework, such as PHP or Django, is installed the securing of the web server takes on a much larger dimension. In those cases, it is vital to keep up with security patches and minimize the installed or enabled functionality of said frameworks. Permissions on files and databases must be set to "need to know only" style of implementation, that is, only the least privilege required should be used. Therefore, a web server must be run as an unprivileged user, the database user used by the web server should be restricted to its own database entries and must be unprivileged as well. There is much more that can be done to tighten up security.

Note also that allowing the certbot software to run as a root user is not ideal. The system administrator should migrate to using the DNS-01 method, rather than the HTTP-01 method described herein, as soon as feasible. The DNS-01 method allows for the use of an unprivileged user to manage inbound certificates, with strict controls on what said user can write to the DNS database.

Contents

1 Introduction

There are several moving parts in this deployment. The subsystems needed are:

1. A platform (RPi) and operating system (OS) Raspbian Stretch Lite (or later).
2. An internet connection that provides for incoming connections and has a named IP address controllable by us.
3. A firewall (iptables).
4. A management interface using the secure shell (SSH).
5. A web server (Apache 2.x).
6. The Certbot software and some registration information.
7. Certbot apache plugin to automate the apache TLS configuration.

2 Operating System

The operating system is Raspbian Stretch Lite[1]. We use the lite version because the system has only server roles. That is, no desktop environment is needed to operate the web server. This is typical for a web server and it keeps system updates small. This reduction in installed software also reduces the attack surface of the system.

To get started, the OS must be written to a microSD card and booted for the first time. There is a great set of instructions on how to do this at the raspberry pi installing operating system images web page[2].

We will initially boot the system with a monitor, keyboard and mouse connected. The network will not be connected initially. This will allow us to configure the network interfaces, management services and firewall before attaching to the network. Once the network and SSH are running properly, we can remove the monitor, keyboard and mouse. Further configuration can then use our SSH management interface.

2.1 Changing Password for User pi

The system will start and display a command line login prompt. Enter user `pi`, followed by pressing ENTER, then type password `raspberry`, also followed by ENTER. Immediately type the `passwd` command, enter the current password of `raspberry` and then create a new strong password for the `pi` user. Enter it twice and keep a record of it.

[1] https://www.raspberrypi.org/downloads/raspbian/
[2] https://www.raspberrypi.org/documentation/installation/installing-images/README.md

```
pi@raspberrypi:~ $ passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2.2 Changing Password for User root

Since we are logged in as the pi user, which has `sudo` rights, we will become the `root` user in order to reset its password.

```
pi@raspberrypi:~ $ sudo su -

Wi-Fi is disabled because the country is not set.
Use raspi-config to set the country before use.

root@raspberrypi:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

'sudo su -' calls the `su` command with a single dash, which forces the `root` user's environment to be created, rather than using the `pi` user's environment, which would be a security risk. One risk is altering executable path searches, wherein, for example, the `pi` user's search path may contain a ~/bin entry at the front with commands named `ls` or similar. This could override the system `ls` command and execute arbitrary code.

2.3 "Wi-Fi is disabled" Message

The Wi-Fi is disabled message is annoying, but setting the Wi-Fi country in `raspi-config` failed with a wpa_supplicant error. We'll put an entry directly into the *wpa_supplicant.conf* file and hope this gets rid of the message.

```
pi@www:~ $ sudo more /etc/wpa_supplicant/wpa_supplicant.conf

country=CA
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

2.4 Hostname

We refer to the debian documentation[3] on how to set the hostname. We can use the `hostname` command to view the current value:

```
pi@raspberrypi:~ $ hostname
raspberrypi
```

To set the host name using a systemd compatible method, the `hostnamectl` command is used, with a subcommand `set-hostname`, and the fully qualified domain name (FQDN) of the system.

```
pi@raspberrypi:~ $ hostnamectl set-hostname www.palaceofretention.ca

==== AUTHENTICATING FOR org.freedesktop.hostname1.set-static-hostname ===
Authentication is required to set the statically configured local host name,
 as well as the pretty host name.
Authenticating as: root
Password:
==== AUTHENTICATION COMPLETE ===
```

Review our change:

```
pi@raspberrypi:~ $ hostname
www.palaceofretention.ca
```

Add an entry to the */etc/hosts* file for our new name. We know what our IP address will be, so we enter that now. We also comment out the old raspberrypi entry (note that we needed to logout/login to reset the prompt to show the new hostname).

```
pi@www:~ $ sudo vi /etc/hosts

pi@www:~ $ more /etc/hosts

127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

#127.0.1.1 raspberrypi
192.168.60.130  www.palaceofretention.ca
192.168.60.130  palaceofretention.ca
```

Obviously, you should use your own IP address and hostname in the above instructions.

[3] https://wiki.debian.org/HowTo/ChangeHostname

3 Conventions / Working Practices

This section lists a couple of conventions or working practices that are used during the installation and configuration processes.

3.1 Command Line Convenience

A great time saver for the command line is to map your UP and DOWN arrow keys to search through your command history. This functionality is the default in FreeBSD's `tcsh`, but is sadly, not equipped in the `bash` shell. To enable it, we add these two lines to the *~/.bashrc* file.

```
bind '"\e[A":history-search-backward'
bind '"\e[B":history-search-forward'
```

Do this for all the users (pi and root alike). You can thank me later.

3.2 Be the root User (Trust In Yourself)

To reduce the number of times `sudo` must be typed, we will simply operate as the root user for most of the session.

```
pi@www:~ $ sudo su -
root@www:~#
```

The `su -` mechanic was explained in the "Changing the root user's password" section.

3.3 Command Aliases for the root User (Your Trust is Misplaced)

Since we will make mistakes, it is prudent to create aliases for the `rm`, `cp`, and `mv` commands. Happily, the root user's *.bashrc* file has these command aliases pre-defined and we need only uncomment them. Edit the /ROOT/.BASHRC file and modify the entries at the end of the file.

Change:

```
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
```

To:

```
# Some more alias to avoid making mistakes:
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

Exit your bash session, restart it and observe your new aliases.

```
root@www:~# logout
pi@www:~ $ sudo su -
root@www:~# alias
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
```

Now whenever you use a command that can remove or overwrite a file, you will be asked to confirm the change.

```
root@www:~# touch foo
root@www:~# rm foo
rm: remove regular empty file 'foo'? y
```

4 Network

Having reset passwords and setup our environment, we can begin to configure the network. We want to keep networking simple and minimal.

4.1 Addressing and Interfaces

In almost all cases, you want a web server to have a static, i.e. unchanging, IP address. The best way to do this is to have the dynamic host configuration protocol (DHCP) server reserve an IP address for a specific client base on a unique ID. Most home routers will allow this action via their management interfaces. The web server system's network interface's, media access control (MAC) address is often used as the ID.

We will allow DHCP to give us our network parameters, such as DNS entries and default gateway, so no other configuration is necessary in regards to IP addresses or routing. In the case of IP version 6, a similar mechanism must be used for the server's static IPv6 address (not covered here).

4.1.1 Special Case MAC Reservation

In my case, I must set a certain media access control (MAC) address for my ethernet port. This is because the DHCP server that I connect to has a client ID, or reservation,

for a specific MAC address. This client ID was put in place in the past and instead of having to contact the ISP and change this reservation, we simply set the MAC address explicitly on the interface to match our ISP's expected reservation. You DO NOT NEED to do this step on your RPi.

4.1.2 The Systemd Way To Set The MAC

The systemd ecosystem has a method for configuring link parameters on network interfaces. Such parameters include the MAC address.

The `ip link` command will show current MAC addresses, also known as the link address.

```
pi@www:/etc/network $ ip link

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP mode DEFAULT group default qlen 1000
    link/ether b8:27:eb:80:bf:50 brd ff:ff:ff:ff:ff:ff
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
    state DOWN mode DORMANT group default qlen 1000
    link/ether b8:27:eb:d5:ea:05 brd ff:ff:ff:ff:ff:ff
```

We see eth0 has a default address of b8:27:eb:80:bf:50. We will change this to 00:3b:61:a7:d7:2a (not real) so as to match our ISP's expected reservation.

We must create a *00-default.link* file within the */etc/systemd/network/* directory. It contains the following:

```
[Match]
OriginalName=eth0

[Link]
MACAddress=00:3b:61:a7:d7:2a
```

The above `Match` section contains the requirements that must match in order for the `Link` section statements to be implemented or set. We match on the original kernel name (`OriginalName=eth0`) for the device, knowing that these can be unstable, but with only one ethernet interface on the RPi, this instability shouldn't affect us.

Therefore, the `eth0` interface will get assigned the MAC address specified in the `Link` section. After a reboot we see the new MAC address on the interface:

```
root@mail:~# ip link show eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAU
    link/ether 00:3b:61:a7:d7:2a brd ff:ff:ff:ff:ff:ff
```

4.1.3 The Wrong Way To Set The MAC

I leave this section in the document so that others may learn that the following method cannot be used reliably.

A deprecated reference for setting MAC addresses using the Debian Distribution is at Working With MAC Addresses[4]. The use of */etc/network/interfaces* file and the `hwaddress` command may not be the best way to do this, but it was the simplest way that initially worked.

It only worked for a short time, though. After the DHCP lease expired, errors like the following showed up:

```
ssh: connect to host 192.168.60.130 port 22443: Operation timed out
mv@nas:~ % ssh wwwpor
ssh: connect to host 192.168.60.130 port 22443: No route to host
mv@nas:~ %
```

The web server also failed to respond to requests, and in fact there was no network connectivity whatsoever.

The interfaces still claimed to be UP and have an IP address and the routing table still had its default route, the network was just dead.

So using the following method does not function correctly.

```
pi@www:~ $ more /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)

auto eth0
iface eth0 inet dhcp
  hwaddress ether 00:3b:61:a7:d7:2a
```

The above configures the `eth0` interface to start at boot (`auto`). The `iface` command specifies that `eth0` will use DHCP for internet protocol 4 addressing, and to set the MAC address of the interface to the value specified. Note that this will make the DHCP client daemon (`dhcpcd`) complain.

It turns out we couldn't ignore the complaints and had to use the `systemd` mechanisms to set the MAC address.

[4] https://debian-administration.org/article/501/Working_with_MAC_addresses

4.2 Disable Unneeded Services

A very secure system listens on no network ports. So we must turn off all network services that listen on ports we don't use or need. We need world wide web (80 and 443), DHCP (68), and SSH ports, and no others at this point. We must determine what other system processes may be listening on network ports and stop them.

There is a useful tool for listing processes and their open files, which also happens to list processes that are listening on network ports. It is called `lsof` (list open files).

```
root@www:~# apt-get install lsof
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsof is already the newest version (4.89+dfsg-0.1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

We can now use the command to show the processes listening on network ports for IPv4 (`-i4` parameter) and IPv6 (`-i6` parameter).

```
root@www:~# lsof -i4 -i6

COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae 329 avahi   12u  IPv4  11474      0t0  UDP *:mdns
avahi-dae 329 avahi   13u  IPv6  11475      0t0  UDP *:mdns
avahi-dae 329 avahi   14u  IPv4  11476      0t0  UDP *:32808
avahi-dae 329 avahi   15u  IPv6  11477      0t0  UDP *:49322
dhclient  463  root    6u  IPv4  10585      0t0  UDP *:bootpc
sshd      531  root    3u  IPv4  12605      0t0  TCP *:2443 (LISTEN)
sshd      531  root    4u  IPv6  12607      0t0  TCP *:2443 (LISTEN)
```

We will use the `systemctl` command to `disable`, `mask` and `stop` the `avahi-daemon`, since it is not needed on a web server.

Disable it:

```
root@www:~# systemctl disable avahi-daemon.socket avahi-daemon.service

Synchronizing state of avahi-daemon.service with SysV service script
 with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable avahi-daemon
Removed /etc/systemd/system/dbus-org.freedesktop.Avahi.service.
Removed /etc/systemd/system/sockets.target.wants/avahi-daemon.socket.
```

Mask it from being started by some unknown dependency:

```
root@www:~# systemctl mask avahi-daemon.socket avahi-daemon.service
Created symlink /etc/systemd/system/avahi-daemon.socket /dev/null.
```

```
Created symlink /etc/systemd/system/avahi-daemon.service /dev/null.
```

Finally, stop it:

```
root@www:~# systemctl stop avahi-daemon.socket avahi-daemon.service
```

Check that it's no longer active on the network:

```
root@www:~# netstat -an46

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:2443            0.0.0.0:*               LISTEN
tcp6       0      0 :::2443                 :::*                    LISTEN

root@www:~# lsof -i4 -i6

COMMAND PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    555 root    3u  IPv4  12584      0t0  TCP *:2443 (LISTEN)
sshd    555 root    4u  IPv6  12586      0t0  TCP *:2443 (LISTEN)
```

There may be other services that need to be disabled, so look carefully and don't assume that the above service was the only one that was listening.

4.3 Firewall (iptables)

A basic set of rules are needed for the system. These rules must be loaded at system startup and are policy based.

4.4 Firewall Policies

The policies we wish to enforce are as follows:

1. Block all incoming connections except those we explicitly allow.
2. Do not allow forwarding (we are not a router).
3. Allow outgoing connections originating on the server.

   Optionally, we could block output by default and restrict outgoing traffic to ports needed for our services, maintenance and updates (both certificates and software). We would have to be careful to add rules for other outgoing packets, such as DHCP and possibly others.
4. Allow all the loopback device's packets to move freely.
5. Allow DHCP packets to pass between the DHCP server and our client.

6. Allow incoming web server requests on standard ports 80 and 443.
7. Allow incoming SSH requests on our non-standard port.
8. Optionally, immediately reject connections on port 22.
9. Allow incoming packets that are part of established connections (stateful).

4.4.1 iptables-persistent

There exists a package called `iptables-persistent` that is very helpful for making the `iptables` rulesets load at system startup. We install it:

```
root@www~# apt-get install iptables-persistent
```

During installation two textbox pop-ups are sent asking whether to store IPv4 and IPv6 rules in the following two files:

- */etc/iptables/rules.v4* - the internet protocol version 4 rules, and;
- */etc/iptables/rules.v6* - the internet protocol version 6 rules.

We answered yes to both and the two files were created for us.

To control the service use the following `service` commands:

```
root@www:/etc# service netfilter-persistent save
root@www:/etc# service netfilter-persistent stop
root@www:/etc# service netfilter-persistent start
```

We edit the *rules.v4* file and makes changes to implement the policies we listed above. Note these rules are modified later on, so this set remains only to match the explanatory text which follows them.

```
*filter
:INPUT DROP [1:76]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [1:76]
-A INPUT -i lo -j ACCEPT
-A INPUT -s 192.168.0.0/16 -p udp -m udp --dport 68 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22443 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j REJECT
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
```

These rules have the following meanings:

***filter** start the definitions for iptables' internal `filter` table.

:INPUT DROP [1:76] default to dropping packets for the input chain.

:FORWARD DROP [0:0] default to dropping packets for the forward chain.

:OUTPUT ACCEPT [1:76] default to accepting all packets on the output chain. This applies to the loopback interface as well as the ethernet interface. It allows connections originating from our server to go out to the Internet. It also allows our web and ssh servers to return their answers/responses to traffic from the outside world.

-A INPUT -i lo -j ACCEPT allow incoming traffic to the loopback interface.

-A INPUT -s 192.168.0.0/16 -p udp -m udp –dport 68 -j ACCEPT allow packets with a source address of our DHCP server, or its subnet, to send us DHCP packets. Our outgoing packets originate from UDP port 68 and are sent to port 67 on the server. These packets are allowed by the default output policy of accept. The server response packets come from port 67 and are sent to port 68 and are allowed by this rule. In a home router case, the source address would be the address of the router itself, which is likely to be of the form 192.168.1.1 or similar. This rules uses a CIDR subnet specification as a source address because I cannot guarantee which host will act as the DHCP server on the ISP's network side.

-A INPUT -p tcp -m tcp –dport 80 -j ACCEPT allow incoming requests on port 80 for our web server.

-A INPUT -p tcp -m tcp –dport 443 -j ACCEPT allow incoming requests on port 443 for our web server.

-A INPUT -p tcp -m tcp –dport 22443 -j ACCEPT allow incoming requests on port 22443 for our SSH server.

-A INPUT -p tcp -m tcp –dport 22 -j REJECT immediately reject connection requests to port 22.

-A INPUT -p icmp -m icmp –icmp-type 8 -j ACCEPT allow the ICMP ping command on our server.

-A INPUT -m state –state RELATED,ESTABLISHED -j ACCEPT allow incoming requests that match an established or related connection that previously originated from our system. It's possible that this would allow server DHCP responses to be allowed, but it's best to be explicit, as above, in some cases.

COMMIT activate the rules.

The rules are loaded on system boot using systemd's startup mechanisms. The `iptables-persistent` package created the proper systemd unit files for systemd integration.

We edit the *rules.v6* file and makes changes to implement blocking of all IPv6 packets, since we don't have IPv6 connectivity.

```
root@www:/etc# more /etc/iptables/rules.v6

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -j REJECT --reject-with icmp6-port-unreachable
-A FORWARD -j REJECT --reject-with icmp6-port-unreachable
-A OUTPUT -j REJECT --reject-with icmp6-port-unreachable
COMMIT
```

4.4.2 Updated IPv4 Rules

The following contains an updated list of rules that are a little more useful, but have the same spirit as the original set:

```
root@www:/etc# more iptables/rules.v4

# Generated by iptables-save v1.6.0 on Sun Mar 24 23:26:47 2019
*filter
:INPUT DROP [7:296]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -s 127.0.0.0/8 ! -i lo -j REJECT --reject-with icmp-port-unreachable
-A INPUT -s 192.168.60.0/24 -p udp -m udp --dport 68 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 80 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 443 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 2443 -j ACCEPT
-A INPUT -i eth0 -p udp -m state --state ESTABLISHED -m udp --sport 53 -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state ESTABLISHED -m tcp --sport 53 -j ACCEPT
-A INPUT -i eth0 -p udp -m state --state ESTABLISHED -m udp --sport 123 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j REJECT --reject-with icmp-port-unreachable
-A INPUT -p icmp -m state --state NEW -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m limit --limit 3/min -j LOG --log-prefix "iptables_INPUT_denied: "
-A FORWARD -m limit --limit 3/min -j LOG --log-prefix "iptables_FORWARD_denied: "
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m state --state ESTABLISHED -m tcp --sport 80 -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m state --state ESTABLISHED -m tcp --sport 443 -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m state --state ESTABLISHED -m tcp --sport 2443 -j ACCEPT
-A OUTPUT -o eth0 -p udp -m state --state NEW,ESTABLISHED -m udp --dport 53 -j ACCEPT
```

```
-A OUTPUT -o eth0 -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 53 -j ACCEPT
-A OUTPUT -o eth0 -p udp -m state --state NEW,ESTABLISHED -m udp --dport 123 -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -m limit --limit 3/min -j LOG --log-prefix "iptables_OUTPUT: "
COMMIT
# Completed on Sun Mar 24 23:26:47 2019
```

Additions include:

- some logging (`-j LOG`) statements;
- more explicit input chain `state` statements, and;
- more explicit output chain statements.

4.5 Plug in the Network Cable

A subsection is dedicated to this action so as to stress the order in which events should occur when connecting directly to the Internet. The network is connected only when we are sure that passwords have been changed, the firewall is running and network services are reduced to a minimum.

Ideally, the system will be configured on a private network prior to deployment.

In many cases, your system will be behind a network address translating (NAT) router and not directly connected to the Internet. This NAT layer affords a modicum of protection, since most routers won't forward incoming connection requests to hosts on the local area network, unless told to. Being disconnected is still a good habit to learn, but is not as urgent in this case. You will also need to create port forwarding rules in your NATing router to redirect both SSH and web server traffic to your RPi.

4.6 Secure Shell (SSH)

The secure shell protocol will act as our management interface, allowing us command line control of the server over the network. The SSH connection will also be used to transfer HTML files, as needed, using either scp or sftp based software.

4.6.1 Policies

We need to configure the SSH server to follow these policies:

1. Listen on a non-standard port to ignore most robotic attackers and keep our logs cleaner.
2. Disallow root login over SSH entirely. We always login as an unprivileged user and switch to the root user, or use `sudo` as needed.

3. Require public/private keys for login, that is, no password based logins allowed. Obviously, we must initially allow passwords until we setup our key pairs and transfer the public portion to the pi user's *authorized_keys* file.

The configuration file for the ssh daemon has a path of */etc/ssh/sshd_config*. We edit it and make sure the following configuration options are specified so as to implement our policy. The ellipses indicate other options that are not shown and unchanged from their defaults.

```
Port 22443
...
PermitRootLogin no
...
```

The key based login policy will be implemented after we have created and transferred the required public and private key pairs.

4.6.2 Startup

The easiest way to enable SSH at system startup, on Raspbian, is to create an empty file named *ssh* in the */boot* directory and reboot the system.

```
root@www:~# cd /boot
root@www:/boot# touch ssh
root@www:/boot# reboot
```

After the reboot, SSH will be started:

```
root@www:~# netstat -an46

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22443           0.0.0.0:*               LISTEN
tcp6       0      0 :::22443                :::*                    LISTEN
```

We can also see its status:

```
root@www:~# service ssh status

 ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-03-21 05:49:55 GMT; 45min ago
  Process: 444 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 447 (sshd)
   CGroup: /system.slice/ssh.service
           447 /usr/sbin/sshd -D
```

```
Mar 21 05:49:55 raspberrypi systemd[1]: Starting OpenBSD Secure Shell server...
Mar 21 05:49:55 raspberrypi sshd[447]: Server listening on 0.0.0.0 port 22443.
Mar 21 05:49:55 raspberrypi sshd[447]: Server listening on :: port 22443.
Mar 21 05:49:55 raspberrypi systemd[1]: Started OpenBSD Secure Shell server.
```

4.6.3 Public / Private Key Authentication

The use of public / private key pairs is the best way to connect to the SSH management interface. It completely defeats password attacks on the SSH server and, provided that the private key files are suitably protected by a passphrase, is very secure.

Key pairs are generated on a system which is intended to connect to the RPi so as to manage it. In my case it would be my macbook laptop, or my FreeBSD workstation. In either case, we refer to that system as the management workstation. You can use the same key pair on many different systems, or have a public/private key pair specific to each system from which you plan to connect to the RPi. Using a separate key pair for each connecting device is preferred and allows the cancellation of only that key in the case that its private key is lost or stolen.

You can also carry your SSH keys on a USB stick if you are very brave.

Use `ssh-keygen` on Unix or MacOS systems to create a key pair. The `PuTTy` program on Microsoft platforms can also be used, but won't be covered here. The next set of commands are run in a terminal on the management workstation.

One always protects a private key with a passphrase, so enter a strong one when prompted below.

```
[fruit:~] student% ssh-keygen -C for_rpi_www.por.ca -f ~/.ssh/for_rpi_www.por.ca

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in for_rpi_www.por.ca.
Your public key has been saved in for_rpi_www.por.ca.pub.
```

In the above command the `-C` argument adds a comment to the public key to help in identifying it later. The `-f` argument specifies the file name for the private key. Here we store it in the *.ssh* folder, found in the management user's home directory. We give it a meaningful name so as to distinguish it from other keys in the *.ssh* directory. The public key will be named the same, but have a *.pub* extension. It is the public key we copy to the RPi server.

Copy the public key to the RPi while we still allow password logins:

```
[fruit:~] student% cat ~/.ssh/for_rpi_www.por.ca.pub | ssh pi@192.168.60.130 \
 'umask 0077; mkdir -p .ssh; cat >> .ssh/authorized_keys && echo "Key copied"'
```

```
ssh: connect to host 192.168.60.130 port 22: Connection refused
```

Of course, we've changed the port, so we must specify it in the `ssh` command.

```
[fruit:~] student% cat ~/.ssh/for_rpi_www.por.ca.pub | \
  ssh -p 22443 pi@192.168.60.130 \
 'umask 0077; mkdir -p .ssh; cat >> .ssh/authorized_keys && echo "Key copied"'

The authenticity of host '[192.168.60.130]:22443 ([192.168.60.130]:22443)'
can't be established.

ECDSA key fingerprint is SHA256:GsXT3CFc+e52+zGSs+10nsLWJaMKisFtSMt6i+OyKG8.
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[192.168.60.130]:22443' (ECDSA) to the list of
known hosts.
pi@192.168.60.130's password:
Key copied
```

We decompose the above command:

cat ~/.ssh/for_rpi_www.por.ca.pub send the contents of the public key file to standard output.

| take standard output and pipe it to the next command (ssh).

ssh -p 22443 pi@192.168.60.130 use the ssh command to connect to port 22443 of our RPi host at the given IP address using the username pi.

'umask 0077; mkdir -p .ssh; cat >> .ssh/authorized_keys && echo "Key copied"' the command that `ssh` will execute on the remote system.

This is a compound command that has four parts and must be entirely enclosed in single quotes. The first three commands are separated by semi-colons, while the fourth is separated by a double ampersand. The semi-colon allows the separate commands to be executed one after the other regardless of success or failure. For example, the *.ssh* directory may already exist and `mkdir` will exit with an error that we don't care about. The double ampersand will only allow the next command to execute if the preceding command was successful.

umask 0077 temporarily set the file creation bits to remove group and world permissions for any new files or directories created.

mkdir -p .ssh in the pi user's home directory, create the *.ssh* directory if it doesn't exist, using the `umask` previously set.

cat >> .ssh/authorized_keys take the standard input that was sent to the `ssh` command and concatenate it to the file named *.ssh/authorized_keys* (relative

to the pi user's home directory).

echo "Key copied" a success indicator returned only if the previous commands were successful.

Some systems support a command called `ssh-copy-id`, which does the above, but wasn't available on macOS. Here is an example from a FreeBSD unix workstation:

```
mv@nas:~ % ssh-copy-id -i .ssh/nas2rpi_www.por.ca.pub -p 22443 pi@192.168.60.121
pi@192.168.60.121's password:
mv@nas:~ %
```

Now that keys have been configured and transferred, we can disallow password based authentication by editing */etc/ssh/sshd_config* and setting the line below as shown:

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
```

Restart SSH.

```
root@www:~# service ssh restart
root@www:~# service ssh status
 ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-03-21 07:06:06 GMT; 19s ago
  Process: 1129 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 1132 (sshd)
   CGroup: /system.slice/ssh.service
           1132 /usr/sbin/sshd -D

Mar 21 07:06:06 raspberrypi systemd[1]: Starting OpenBSD Secure Shell server...
Mar 21 07:06:06 raspberrypi sshd[1132]: Server listening on 0.0.0.0 port 22443.
Mar 21 07:06:06 raspberrypi sshd[1132]: Server listening on :: port 22443.
Mar 21 07:06:06 raspberrypi systemd[1]: Started OpenBSD Secure Shell server.
```

Test login with a password:

```
mv@nas:~ % ssh -p 22443 pi@192.168.60.121
Permission denied (publickey).
```

Test login with our key identity:

```
mv@nas:~ % ssh -i .ssh/nas2rpi_www.por.ca -p 22443 pi@192.168.60.121
Enter passphrase for key '.ssh/nas2rpi_www.por.ca':
Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20:27:48 GMT 2019 armv7l

[...]
pi@www:~ $
```

4.7 SSH Client Config File Convenience

It is useful to create an entry in the *~/.ssh/config* file that specifies all the information required to connect to the web server via ssh. This *config* file is on the management workstation and NOT on the RPi.

```
mv@nas:~ % more .ssh/config
Host rpiwww
 Port 22443
 IdentityFile ~/.ssh/nas2rpi_www.por.ca
 User pi
 Hostname 192.168.60.130
```

Now we can simply type `ssh rpiwww` to login:

```
mv@nas:~ % ssh rpiwww
Enter passphrase for key '/home/mv/.ssh/nas2rpi_www.por.ca':
[...]
pi@www:~ $
```

5 Updating and Upgrading

The next priority in the configuration is to bring the system software up to date. Excellent instructions can be found at Updating and Upgrading Raspbian[5].

Once again, to reduce the number of times `sudo` must be typed, we will simply operate as the root user for the entire session.

```
pi@www:~ $ sudo su -
root@www:~#
```

Update the repositories.

```
root@www:~# apt-get update

Get:1 http://archive.raspberrypi.org/debian stretch InRelease [25.4 kB]
Hit:2 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Get:3 http://archive.raspberrypi.org/debian stretch/main armhf Packages [222 kB]
Get:4 http://archive.raspberrypi.org/debian stretch/ui armhf Packages [44.4 kB]
Fetched 292 kB in 2s (102 kB/s)
Reading package lists... Done
```

Now do a 'dist-upgrade' which will upgrade the OS and the installed software.

```
root@www:~# apt-get dist-upgrade
```

[5] https://www.raspberrypi.org/documentation/raspbian/updating.md

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  ...
59 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 112 MB of archives.
After this operation, 848 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
...
```

And one final repository update:

```
root@www:~# apt-get update

Hit:1 http://archive.raspberrypi.org/debian stretch InRelease
Get:2 http://raspbian.raspberrypi.org/raspbian stretch InRelease [15.0 kB]
Fetched 15.0 kB in 1s (12.7 kB/s)
Reading package lists... Done
```

We should have an up-to-date system now. Reboot to make sure we're using the new software.

6 Apache Web Server

We will install the latest version of apache. For our purposes, we will stick to just the web server. Software such as PHP or databases are not covered.

```
root@www:~# apt-cache search apache
[...] Too much output.
```

Scrolling through the large list, we see that the package is called apache2.

```
pi@www:~ $ sudo apt-get install apache2

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine |
```

```
  apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,953 kB of archives.
After this operation, 6,275 kB of additional disk space will be used.
Do you want to continue? [Y/n] y

[...]
Enabling module mpm_event.
Enabling module authz_core.
Enabling module authz_host.
Enabling module authn_core.
Enabling module auth_basic.
Enabling module access_compat.
Enabling module authn_file.
Enabling module authz_user.
Enabling module alias.
Enabling module dir.
Enabling module autoindex.
Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.

Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service
  /lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service
 /lib/systemd/system/apache-htcacheclean.service.
Processing triggers for libc-bin (2.24-11+deb9u4) ...
Processing triggers for systemd (232-25+deb9u9) ...
```

We should look through the configuration directory which is located in */etc/apache2*.

6.1 Initial Configuration Settings

We have to change the default configuration slightly. The */etc/apache2* directory has a *sites-enabled* folder that contains the configuration files for virtual hosts and the default server in *sites-enabled/000-default.conf*.

We updated the `ServerName` and `ServerAdmin` directives.

```
root@www:/etc/apache2# cd sites-enabled/
root@www:/etc/apache2/sites-enabled# more 000-default.conf

<VirtualHost *:80>

    ServerName www.palaceofretention.ca

    ServerAdmin webmaster@palacofretention.ca
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

6.2 Test HTML File

Create a unique *index.html* for the web server and ensure there are no other default pages being served. The web root directory is located at */var/www/html/*.

```
root@www:~# cd /var/www/html/

root@www:/var/www/html# ls -la
total 20
drwxr-xr-x 2 root root  4096 Mar 21 23:34 .
drwxr-xr-x 3 root root  4096 Mar 21 23:34 ..
-rw-r--r-- 1 root root 10701 Mar 21 23:34 index.html
```

Move the distribution's *index.html* file out of the way.

```
root@www:/var/www/html# mv index.html ../index.html-dist
```

Create a new basic *index.html* file.

```
root@www:/var/www/html# vi index.html

<html>
<head>
```

```
    <title>Let's Encrypt Demo Host</title>
</head>
<body>
    <p>Do we have a green lock yet?</p>
</body>
</html>
```

6.3 Starting and Testing Apache

Starting up apache2:

```
root@www:/etc/apache2# service apache2 start
```

Point a browser at our web server to see what we get:

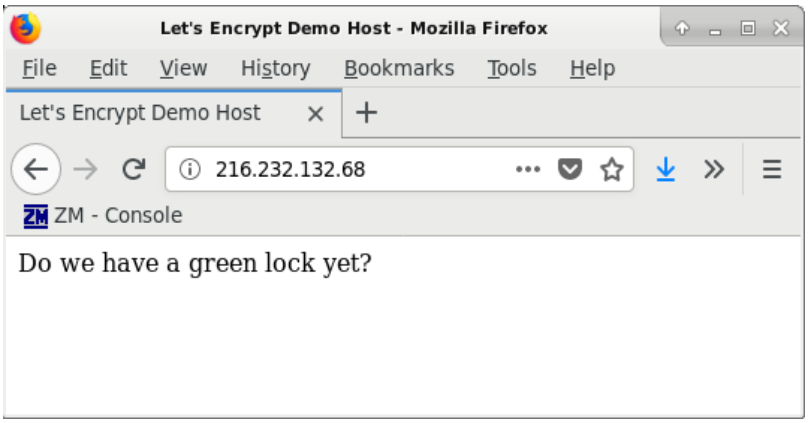

Figure 1: **On the Internet Without a Certificate**

We can stop the web server:

```
root@www:/etc/apache2# service apache2 stop
```

7 Certbot

According to the official installation instructions[6] for apache and debian, we are supposed to add the backports repository to the apt-get system. This turned out to be unnecessary, however, as the certbot and its apache plugin are in the normal repositories.

Let's look for certbot software:

```
root@www:~# apt-cache search certbot
certbot - automatically configure HTTPS using Let's Encrypt
letsencrypt - transitional dummy package
python-certbot-doc - client documentation for certbot
python3-certbot - main library for certbot
python-certbot-apache - transitional dummy package
python-certbot-apache-doc - Apache plugin documentation for Certbot
python3-certbot-apache - Apache plugin for Certbot
python-certbot-dns-cloudflare-doc - Documentation for the Cloudflare DNS plugin for Certb
```

[6] https://certbot.eff.org

```
python3-certbot-dns-cloudflare - Cloudflare DNS plugin for Certbot
python-certbot-dns-rfc2136-doc - Documentation for the RFC 2136 DNS plugin for Certbot
python3-certbot-dns-rfc2136 - RFC 2136 DNS plugin for Certbot
python-certbot-nginx - transitional dummy package
python-certbot-nginx-doc - Nginx plugin documentation for Certbot
python3-certbot-nginx - Nginx plugin for Certbot
```

Install the certbot software and the python 3 version of the certbot-apache plugin.

```
root@www:~# apt-get install certbot python3-certbot-apache

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  augeas-lenses libaugeas0 python-pyicu python3-acme python3-asn1crypto
  python3-augeas python3-certbot python3-cffi-backend python3-configargparse
  python3-configobj python3-cryptography python3-future python3-idna
  python3-josepy python3-mock python3-openssl python3-parsedatetime
  python3-pbr python3-requests-toolbelt python3-rfc3339 python3-setuptools
  python3-tz python3-zope.component python3-zope.event python3-zope.hookable
  python3-zope.interface
Suggested packages:
  augeas-doc python3-certbot-nginx python-certbot-doc augeas-tools
  python-acme-doc python-certbot-apache-doc python-configobj-doc
  python-cryptography-doc python3-cryptography-vectors python-future-doc
  python-mock-doc python-openssl-doc python3-openssl-dbg python-setuptools-doc
The following NEW packages will be installed:
  augeas-lenses certbot libaugeas0 python-pyicu python3-acme
  python3-asn1crypto python3-augeas python3-certbot python3-certbot-apache
  python3-cffi-backend python3-configargparse python3-configobj
  python3-cryptography python3-future python3-idna python3-josepy python3-mock
  python3-openssl python3-parsedatetime python3-pbr python3-requests-toolbelt
  python3-rfc3339 python3-setuptools python3-tz python3-zope.component
  python3-zope.event python3-zope.hookable python3-zope.interface
0 upgraded, 28 newly installed, 0 to remove and 43 not upgraded.
Need to get 2,645 kB of archives.
After this operation, 12.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

[...]

Setting up python3-certbot (0.28.0-1~bpo9+1) ...
Setting up certbot (0.28.0-1~bpo9+1) ...
Created symlink /etc/systemd/system/timers.target.wants/certbot.timer
```

```
 /lib/systemd/system/certbot.timer.
Setting up python3-certbot-apache (0.28.0-1~bpo9+1) ...
Processing triggers for libc-bin (2.24-11+deb9u4) ...
```

Test run and show the version of `certbot`:

```
root@www:~# certbot --version
certbot 0.28.0
```

7.1 ACME Protocol

It is important to be aware of and look over the ACME Protocol standard[7]. A reading of this document provides background knowledge on how the `certbot` client and server exchange information.

7.1.1 HTTP Challenge

We use the HTTP-01 challenge method, since we control the web server directly and it is the simplest way to get our Let's Encrypt (LE) certificate. The process goes something like this:

1. We register with the LE ACME server and receive our account credentials. These credentials are used for signing the various requests and challenges sent by us.
2. Certbot creates a private key and certificate signing request (CSR). It sends the CSR to the ACME server.
3. Certbot asks the ACME server for a special token.
4. The server replies with a token.
5. Certbot writes this token to a file in our web server's document root directory at */var/www/html/.well-known/acme-challenge/*token**
6. The ACME server requests this file from our web server, and if successful generates a certificate for us.
7. Certbot retrieves the certificate and places it in a standard directory.
8. Certbot reconfigures apache to use this certificate and, optionally, force TLS use on our server by configuring apache to redirect port 80 requests to port 443.

[7] https://tools.ietf.org/html/rfc8555

7.1.2 DNS Challenge

The DNS-01 challenge is similar to the HTTP-01 challenge except that it does not change any web server configurations directly (hooks are used). It also sends DNS dynamic updates containing special `TXT` records to a domain name server. These `TXT` records contain the token, which the ACME server retrieves using DNS requests.

This is the only way to get a wildcard certificate for your domain. We won't be covering this method in this talk, but this is the method I use for getting LE certificates.

7.2 Location of Files

This short section details where files and software are generally found for this implementation. The directory */etc/letsencrypt* will contain the configuration files and certificates.

```
root@www:/etc# ls letsencrypt/
cli.ini
```

Log files are found in: */var/log/letsencrypt/letsencrypt.log*

7.2.1 Certificates

The certificates are stored in the following directory: */etc/letsencrypt/live* under a subdirectory named after the domain.

7.2.2 Systemd files

The certbot service file that systemd uses for the certbot timer unit can be found in */lib/systemd/system/certbot.service*.

The certbot timer file is found in */etc/systemd/system/timers.target.wants/certbot.timer*.

7.3 Required Credentials

To register, an optional email address to receive renewal notifications can be provided.

7.3.1 Email Address: letsencrypt@palacofretention.ca

An email address is optional, but useful, for the LE registration. LE promises that only certificate expiry warnings and service notifications are used with this email address.

7.3.2 Let's Encrypt Account Authorized Key Pair

Let's Encrypt creates a user account using a public/private key pair. We have the private key and with it, for example, we sign nonce challenges issued by the LE automation systems. This key pair is managed by the certbot software and we can take a fairly hands off approach when dealing with it (i.e. we needn't do anything except back it up).

7.3.3 Registration

We'll need to register with LE and do so with the following command:

```
certbot register \
  --no-eff-email \
  --email letsencrypt@palacofretention.ca \
  --agree-tos \
  --debug
```

The command's output:

```
root@www:/etc/letsencrypt# certbot register \
>   --no-eff-email \
>   --email letsencrypt@palacofretention.ca \
>   --agree-tos \
>   --debug
Saving debug log to /var/log/letsencrypt/letsencrypt.log

IMPORTANT NOTES:
 - 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.
```

We should be ready to issue our `certbot` commands now.

7.4 Staging

The LE service allows for using staging servers to work out any issues before getting a real certificate. We will make use of this.

```
certbot \
  --staging \
  --dry-run \
  --apache \
```

```
  --debug \
  -d palaceofretention.ca -d www.palaceofretention.ca
```

The results of the staging dry-run:

```
root@www:/etc/letsencrypt# certbot \
>   --staging \
>   --dry-run \
>   --apache \
>   --debug \
>   -d palaceofretention.ca -d www.palaceofretention.ca
Exiting abnormally:
Traceback (most recent call last):
[...]
certbot.errors.Error: --dry-run currently only works with the
'certonly' or 'renew' subcommands ('run')
Please see the logfile '/tmp/tmpkcmhqbb_' for more details.
```

The error says `dry-run` isn't available for staging so we'll remove the `dry-run` parameter and try again:

```
certbot \
  --staging \
  --apache \
  --debug \
  -d palaceofretention.ca -d www.palaceofretention.ca
```

The results are a staging certificate!

```
root@www:/etc/letsencrypt# certbot
                           --staging
                           --apache
                           --debug
                           -d palaceofretention.ca
                           -d www.palaceofretention.ca
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): letsencrypt@palacofretention.ca

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
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-staging-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: a
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: n
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for palaceofretention.ca
http-01 challenge for www.palaceofretention.ca
Enabled Apache rewrite module
Waiting for verification...
Cleaning up challenges
Created an SSL vhost at /etc/apache2/sites-available/000-default-le-ssl.conf
Enabled Apache socache_shmcb module
Enabled Apache ssl module
Deploying Certificate to VirtualHost /etc/apache2/sites-available/000-default-le-ssl.conf
Enabling available site: /etc/apache2/sites-available/000-default-le-ssl.conf
Deploying Certificate to VirtualHost /etc/apache2/sites-available/000-default-le-ssl.conf

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Enabled Apache rewrite module
Redirecting vhost in /etc/apache2/sites-enabled/000-default.conf to ssl vhost in /etc/apa

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled https://palaceofretention.ca and
https://www.palaceofretention.ca

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=palaceofretention.ca
https://www.ssllabs.com/ssltest/analyze.html?d=www.palaceofretention.ca
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
```

```
   /etc/letsencrypt/live/palaceofretention.ca/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/palaceofretention.ca/privkey.pem
   Your cert will expire on 2019-06-21. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - 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.
root@www:/etc/letsencrypt#
```

I guess we were asked for email and terms of service agreements because we are using the staging servers, and registered with the non-staging servers.

The important things to note are the creation of this file:

```
root@mail:~# more /etc/apache2/sites-enabled/000-default-le-ssl.conf
<IfModule mod_ssl.c>
<VirtualHost *:443>
    #[...]
ServerName www.palaceofretention.ca

ServerAdmin webmaster@palacofretention.ca
DocumentRoot /var/www/html

    #[...]

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

    #[...]

ServerAlias palaceofretention.ca
Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateFile /etc/letsencrypt/live/palaceofretention.ca/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/palaceofretention.ca/privkey.pem
</VirtualHost>
</IfModule>
```

Changes were added at the end of the file and include the `ServerAlias`, `Include`, `SSLCertificateFile`, and `SSLCertificateKeyFile` directives.

There are also alterations of the *sites-enabled/000-default.conf* file:

```
root@mail:~# more /etc/apache2/sites-enabled/000-default.conf
<VirtualHost *:80>
    #[...]
ServerName www.palaceofretention.ca

ServerAdmin webmaster@palacofretention.ca
DocumentRoot /var/www/html

    #[...]

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

    #[...]

RewriteEngine on
RewriteCond %{SERVER_NAME} =palaceofretention.ca [OR]
RewriteCond %{SERVER_NAME} =www.palaceofretention.ca
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Changes were added at the end of the file and include the `RewriteEngine`, `RewriteCond`, and `RewriteRule` directives.

7.5 Production

Since our staging tests were successful, we can get our actual production certificate by removing the staging argument from the command:

```
certbot \
  --apache \
  --debug \
  -d palaceofretention.ca -d www.palaceofretention.ca
```

Results:

```
root@www:/etc/letsencrypt# certbot
                            --apache
                            --debug
                            -d palaceofretention.ca
                            -d www.palaceofretention.ca
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Cert not yet due for renewal
```

```
You have an existing certificate that has exactly the same domains or certificate name yo
(ref: /etc/letsencrypt/renewal/palaceofretention.ca.conf)

What would you like to do?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Attempt to reinstall this existing certificate
2: Renew & replace the cert (limit ~5 per 7 days)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Renewing an existing certificate
Performing the following challenges:
http-01 challenge for palaceofretention.ca
http-01 challenge for www.palaceofretention.ca
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/apache2/sites-enabled/000-default-le-ssl.conf
Deploying Certificate to VirtualHost /etc/apache2/sites-enabled/000-default-le-ssl.conf

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Failed redirect for palaceofretention.ca
Unable to set enhancement redirect for palaceofretention.ca
Exiting abnormally:
[...]
 File "/usr/lib/python3/dist-packages/certbot_apache/configurator.py", line 1860, in _ena
    "Unable to find corresponding HTTP vhost; "
certbot.errors.PluginError: Unable to find corresponding HTTP vhost; Unable to create one
Please see the logfiles in /var/log/letsencrypt for more details.

IMPORTANT NOTES:
 - We were unable to set up enhancement redirect for your server,
   however, we successfully installed your certificate.
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/palaceofretention.ca/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/palaceofretention.ca/privkey.pem
   Your cert will expire on 2019-06-21. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
```

```
with the "certonly" option. To non-interactively renew *all* of
your certificates, run "certbot renew"
```

We received an error regarding the redirect setup, likely due to it already having been performed by the staging process. This means that we can select option 1 for that question from now on.

Despite the error, we received our new certificate on our production server.

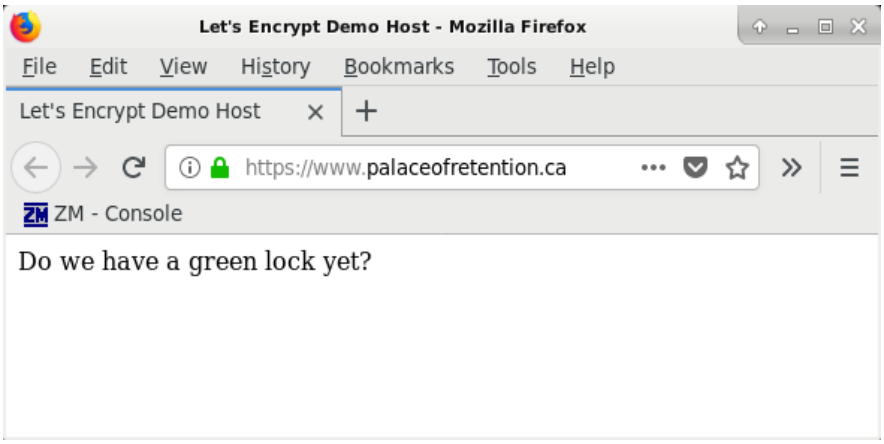

Figure 2: **On the Internet With an Accepted Certificate**

7.6 Automatic Renewal

The renew command is used with the `certbot` program to automatically renew certificates. Here is a dry-run:

```
root@www:/etc/letsencrypt# certbot renew --dry-run

Saving debug log to /var/log/letsencrypt/letsencrypt.log

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing /etc/letsencrypt/renewal/palaceofretention.ca.conf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Cert not due for renewal, but simulating renewal for dry run
Plugins selected: Authenticator apache, Installer apache
Renewing an existing certificate
Performing the following challenges:
http-01 challenge for palaceofretention.ca
http-01 challenge for www.palaceofretention.ca
Waiting for verification...
Cleaning up challenges
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
new certificate deployed with reload of apache server; fullchain is
/etc/letsencrypt/live/palaceofretention.ca/fullchain.pem
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
** DRY RUN: simulating 'certbot renew' close to cert expiry
**          (The test certificates below have not been saved.)

Congratulations, all renewals succeeded. The following certs have been renewed:
  /etc/letsencrypt/live/palaceofretention.ca/fullchain.pem (success)
** DRY RUN: simulating 'certbot renew' close to cert expiry
**          (The test certificates above have not been saved.)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
root@www:/etc/letsencrypt#
```

We note that automatic renewal was installed via a `systemd` timer unit file that came with the `certbot` installation.

```
root@www:~# more /lib/systemd/system/certbot.timer
[Unit]
Description=Run certbot twice daily

[Timer]
OnCalendar=*-*-* 00,12:00:00
RandomizedDelaySec=43200
Persistent=true

[Install]
WantedBy=timers.target
```

Therefore, our renewal is already setup.

```
root@www:~# systemctl status certbot.timer
 certbot.timer - Run certbot twice daily
   Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: en
   Active: active (waiting) since Fri 2019-03-22 21:20:43 PDT; 53min ago

Mar 22 21:20:43 www.palaceofretention.ca systemd[1]: Started Run certbot twice daily
```

7.6.1 Post Renewal Hook

The *certbot.service* file found in */lib/systemd/system/* has the command that `certbot` will use to renew certificates.

We need to make a slight change to this file so we can restart the apache service upon renewal.

```
root@www:/lib/systemd/system# more certbot.service
[Unit]
Description=Certbot
Documentation=file:///usr/share/doc/python-certbot-doc/html/index.html
Documentation=https://letsencrypt.readthedocs.io/en/latest/
[Service]
Type=oneshot
ExecStart=/usr/bin/certbot -q renew --post-hook "service apache2 restart"
PrivateTmp=true
```

We added the `post-hook` argument. Let's check the status:

```
root@www:~# systemctl status certbot.service

Warning: certbot.service changed on disk. Run 'systemctl daemon-reload' to reload units.

   certbot.service - Certbot
   Loaded: loaded (/lib/systemd/system/certbot.service; static; vendor preset: enabled)
   Active: inactive (dead)
     Docs: file:///usr/share/doc/python-certbot-doc/html/index.html
           https://letsencrypt.readthedocs.io/en/latest/
```

Given the warning above, we had better run the `systemctl daemon-reload` command:

```
root@www:~# systemctl daemon-reload
root@www:~# systemctl status certbot.service
   certbot.service - Certbot
   Loaded: loaded (/lib/systemd/system/certbot.service; static; vendor preset: enabled)
   Active: inactive (dead)
```

The `inactive (dead)` status is of no concern, since the service is called as needed and not run continuously.

7.7 Backup of letsencrypt Directory

A manual backup, using the `tar` command (or similar), should be made of the */etc/letsencrypt* folder and copied to another system.

```
root@www:~#  tar -cvf /home/pi/letsencrypt-post-production.tar /etc/letsencrypt
```

Copy the tar file to another system.

Automated backups of the server are also required.

8 Optional: CAA Record in DNS

There exists a mechanism for specifying which certificate authorities can issue certificates for your domain. It is optional, though prudent, to use this mechanism on your domain. We do this by creating CAA records in our DNS zone data.

The CAA record lowers the risk that a mis-issued certificate will be created by a conforming Certificate Authority. It maintains that the only CA authorized to create certificates for palaceofretention.ca is letsencrypt.org.

The main reference for this section is found at Let's Encrypt's Certificate Authority Authorization (CAA) documentation[8]. A useful resource at SSLMate.com[9] allows for the generation of CAA records. We had the following generated:

```
palaceofretention.ca. IN CAA 0 issue "letsencrypt.org"
palaceofretention.ca. IN CAA 0 iodef "mailto:letsencrypt@palacofretention.ca"
```

[8] https://letsencrypt.org/docs/caa/
[9] https://sslmate.com/caa/