Project torpi

Ver 0.01

Pynckels & Pynckels

January 12, 2022

Abstract:

This document describes the elements to take into account and the steps to take to create a reliable, confidential and cheap Internet environment that can be used to prevent eavesdropping on what you are communicating by means of Internet or on what you are doing on Internet.

There will be more technical parts to explain the exact issues at hand, but at the end of the document, there will be a guideline (steps to take in order to create the confidential Internet environment).

The authors of this document will be happy to help you out in case you have problems with the contents of this document, or with the creation of the described confidential Internet environment.

Links to documents:

http://pynckels.github.io/torpi.pdf

Contents

List of Figures

List of Tables

Introduction

Not so long ago, I heard a director-general of the Belgian government reason that his organisation should be an exception on the GDPR. Meaning that his direction should be able to store data without listing up what data they keep, nor how they keep it safe. The only argument that was given as to why this should be the case was: "workload". His colleagues didn't seem to have a problem with the idea nor with the argument.

It seemed strange at the moment, and in all honesty even stranger now that I have had time to think it through, that a governmental organisation is not willing to invest into applying the GDPR whereas private organisations don't have much of a choice to do so. On top of that, it is worrying that a governmental organisation wants to store information without citizens being aware that their data is stored and used, and without citizens being able to count on the fact that there is enough transparency to permit specialized thirds to audit the security of the data storage environment.

Since this is hardly the only case of an organisation that, for all kinds of good or other reasons, wants to capture and store data on citizens, it only seems fair for citizens to have the possibility to prevent certain information to be obtained by their, or for that matter, any government or organisation. So, the authors of this document have created a confidential Internet environment that makes it harder for governmental or other organisation (such as Internet providers) to gather information about the data that is send, the sites that are visited, etc. Note that there are no guarantees in life, but that accessing your data can be made very hard. And that is what this document is all about.

1 Confidential Internet architecture

1.1 General design

1.2 Connection to Internet

1.3 Internet Tor router

1.4 Wireless access point

1.5 Client devices

2 Preventive measures

2.1 Supply chain attack

A supply chain attack is an action to compromise you by targeting your supply chain. More specific, within the frame of this project, this would mean that one or more components of the confidential Internet environment that we are creating would already be compromised before arriving at your premises.

There are two key issues here: 'one or more components' and 'before arriving at your premises'.

There are also two key actions that are important: 'prevention' and 'remediation'. Note that auditing if one or more components are compromised is difficult to almost impossible. So that is a ship that already left the port.

One or more components:

We will take the point of view that we have to prevent having compromised components whatever those components are.

We will also take the point of view that despite preventive measures, all components are compromised anyway.

Before arriving at your premises:

We will take the point of view that we can only take preventive measures by choosing a supply chain that has the smallest chance (per component) of being compromised.

We will also take the point of view that all components are compromised anyway, since this will be done before we have bought them.

3 Installation instructions

This chapter will guide you through the installation process of the different components of the confidential Internet environment.

3.1 Raspberry TOR router

3.1.1 The notations we use

Please note that in the below text, we will indent blue text. This text can be a command to type in on the command line, a text to enter into a text editor, options to choose in a user interface, etc.

A command on the command line will be shown as (execute 'command'):

```
$ command
```

A text to enter in a text editro will be shown as (enter the lines 'text1' and 'text2'):

```
text1
text2
```

Options to choose in a user interface (choose suboption1 under option1):

```
-> Option1 -> Suboption1
```

The above notations seemed the best choice at the moment, but we are all ears if you have a more transparent way of representing commands, texts, options, ...

3.1.2 Choices to make

In the below text, we will make a number of choices that will make the reading of this text easier. However, it is strongly advised that you make your own choices (making names blend in into the environment where you will use this device, making passwords long and with enough variation, etc.)

The second column in the below table is the way we indicate in this document that a command, a sript, ... must be changed by substituting the item (e.g. <hostname>, also substitute the brackets!) with the choice you have made for it. In our case <hostname> inclusive the brackets at the left and right will be replaced with duckster.

The choices we have made are the following:

host sd card device	<hostsddevice>	/dev/sda
device host name	<hostname>	duckster
device user id	<userid>	ducky
device pass word	<password>	mallard
device mac address	<macaddress>	11:22:33:44:55:66
access point name	<apname>	duckstersnetwork
access point pass word	<appassword>	ducksterspassword
access point channel	<apchannel>	11

Again: the choices we have made above are more to make reading this document easy. You are strongly encouraged to change these choices so to make the Internet environment you are creating is as confidential and secure as possible.

3.1.3 Prepare the SD card

The following steps will prepare an SD card for your Raspberry Pi. This can be done by means of a program called Raspberry Pi Imager (that can be downloaded from https://www.raspberrypi.com/software/) that copies the necessary software on the SD card.

If you are using a Linux pc to prepare the SD card, the above can also be done on the command line by following the steps below. Note that ¡hostsddevice¿ must be substituted with the specific device that applies to your case.

It is advisable to use a new empty SD card. The card should be 4Gb or 8Gb (may be larger although unnecessary). This way, we have a guarantee that nobody has fiddled with the contents of the card. We will wipe the card nevertheless.

The card speed is preferably V30, V60 or V90. This will speed up the functioning of the device substantially. In the case of this project, speed is more important than size for what concerns the SD card.

An easy way to find the name of the SD card device is to go to the command line and to execute the command:

```
$ lsblk
```

When you get the results, connect the SD card and reexecute the command:

```
$ lsblk
```

Then, it's just a matter of finding the difference...

To prepare the SD card on your Linux pc, execute the following commands on the command line:

```
$ sudo apt-get -y install buffer
$ sudo apt-get -y install gunzip
$ sudo apt-get -y install wget

$ sudo dd if=/dev/zero of=<hostsddevice> bs=16M status=progress

$ wget -qO- https://downloads.raspberrypi.org/raspios_lite_armhf
  /images/raspios_lite_armhf-2021-11-08/2021-10-30-raspios-
  bullseye-armhf-lite.zip
  | buffer
  | gunzip -c -
  | sudo dd of=<hostsddevice> bs=16M status=progress

$ sudo umount <hostsddevice>1
$ sudo umount <hostsddevice>2
```

Note that this method has been chosen so that as little disc space as possible is used.

To prevent misunderstandings: the name of the file will change with future versions of Raspberry OS. It's advisable to use the latest version that is available to prevent as many security issues as possible.

3.1.4 Prepare the Raspberry Pi

Put the SD card in the Raspberry Pi. From here on, we work on the device itself except when otherwise stated.

— DO NOT CONNECT THE DEVICE TO A NETWORK YET —

Power on the device. After a couple of seconds, a command prompt will be visible, asking for a userid. The default userid is ***pi***. Then a password is asked for. The default password is ***raspberry***. Please note that the default keyboard is qwerty, meaning that, e.g. on an azerty keyboard, the password must be typed as ***rqspberry***.

Log into the device using the default userid and the default password. You will see a command prompt.

Execute the Raspberry configuration program:

```
$ sudo raspi-config
```

Setup the keyboard:

If you have a non qwerty keyboard you can alter the keyboard. Choose the options:

```
-> Localisation options -> Keyboard
```

Choose the suitable keyboard layout and country setting (these will not be visible to the 'outside world'). The keyboard should now have the correct layout. From now on, the password at logon can be typed exactly as it is.

Setup the device host name:

While in raspi-config choose:

```
-> System Options -> Hostname
```

and follow the program directives. It is advisable to choose a <hostname> that blends in into the environment where the device will be used. A strange name will get more attention, and that is exactly what you do not want when you wish to stay under the radar.

Change user password:

While in raspi-config choose:

```
-> System Options -> Password
```

and follow the program directives. It is advisable to choose a strong <password> that is at least 20 tokens long and that can not be found in the usual password lists. The password can be changed again whenever you like.

To change the password, you can also use the terminal command:

```
$ passwd
```

and follow the instructions. Since we were already in the raspi-config program, it seemed appropriate to work from within that program.

Enable remote access via SSH:

While in raspi-config, we will enable SSH to provide remote access to the device. In first instance, this access will go through the normal network. Later, we will provide access

through and only through the wireless access point that we will install on the device. Especially during the first phase, a strong password is important, as annoying as it may be to type it in time after time.

Choose:

```
-> Interface Options -> SSH -> Yes
```

To close raspi-config choose:

```
-> Finish
```

To check if the SSH service will be running from now on, even after a reboot, enter the command:

```
$ sudo reboot
```

Wait for the reboot to finish (this can take 30 seconds, and the screen may go dark in between), log on with the default userid and with the newly chosen password and enter the command:

```
$ sudo systemctl status ssh
```

Check if somewhere in the result of the last statement, a green text "active (running)" is shown.

Change user name:

To make life of intruders more difficult, we will also change the name of the default user to something only you know. We have listed our choices (within the frame of this document) at the beginning of this chapter.

Please do all of the below actions consecutive without powering the device off, or without logging off when not explicitly mentioned.

Please chose an appropriate name for safety sake. Preferably, your user name will be something that is not in a dictionary word list, and that is compliant with the regular expression ^[a-z][-a-z0-9]*$ Meaning that a user name must start with a..z, then can contain a..z and 0..9 multiple times. The longer the name the safer it gets.

Substitute <userid> by the name you have chosen in the following commands:

```
$ cd ~
$ sudo mkdir           /home/<userid>
$ sudo cp .bash_logout /home/<userid>
$ sudo cp .bashrc      /home/<userid>
$ sudo cp .profile     /home/<userid>
$ sudo chown -R pi:pi  /home/<userid>
$ sudo nano /etc/passwd
```

Now find the line:

```
pi:x:1000:1000:,,,:/home/pi:/bin/bash
```

and change the line as follows:

```
pi:x:1000:1000:,,,:/home/ducky:/bin/bash
```

Save the file and quit the editor.

Enable root by changing the the root password:

```
$ sudo su
$ passwd
```

Choose a temporary password for root (for instance "toor"). This password can be short and simple since we will disable root immediately after renaming the default user name, and since for the moment nobody can access the device (not connected to a network).

Exit from root and from pi:

```
$ exit
$ exit
```

Log in as root and rename the default user:

```
$ usermod -l <userid> pi
$ groupmod -n <userid> pi
$ exit
```

Log in as the user you chose for <userid> with the correct password, disable root and clean up the pi directory:

```
$ sudo usermod --pass="*" root
$ sudo rm -fr /home/pi
$ sudo cat /etc/shadow
```

The last statement should show a list of users. Check that there is an asterix (*) on the line of the root user. This is the proof that the root user is disabled again. The user <userid> should contain a lot of clutter after the user name. This is the encrypted <password>.

Secure root access:

Now is the moment to enforce the security for the root user. The root user can not log due to the asterix in the /etc/shadow file, but we can do better than that:

```
$ sudo nano /etc/passwd
```

Find a line that starts with root, such as:

```
root:x:0:0:root:/root:/bin/bash
```

and change the text after the last colon. In case of the example above, this would become

```
root:x:0:0:root:/root:/usr/bin/nologin
```

Save the file and quit the editor.

This way, even if one could log in as root, one does not fall into a shell which makes it difficult to continue as root.

Now, we need to force asking the password each time when the user <userid> wants to execute a "super user do" (sudo) command. On the one side, this prevents making unintended errors, on the other side, it makes the lives of intruders so much more difficult since sudo scripting will be partially blocked.

Enter the command:

```
$ sudo rm /etc/soduers.d/010_pi-nopasswd
$ sudo nano /etc/sudoers
```

We are aware that in theory a per user sudo setup in /etc/sudoers.d is the way to go, but in this case the goal is not to open a door for a user, but to close the door for all users. Hence editing the sudoers file immediately.

Find the lines comparable to:

```
Defaults   env_reset
Defaults   mail_badpass
Defaults   secure_path="/usr/local/sbin:/usr/local/bin:..."
```

and insert an extra line (default timestamp). In case of the example above, this would become:

```
Defaults   env_reset
Defaults   mail_badpass
Defaults   timestamp_timeout=0
Defaults   secure_path="/usr/local/sbin:/usr/local/bin:..."
```

Save the file and quit the editor.

All of the above makes that each user without NOPASSWD privileges will have to give his password for each sudo command that he wants to execute, and that even 'sudo su' is not available due to unavailability of the default shell for root. *Take that, sudo scripters...*

One could opt to drop the secure paths in the sudoers file, but that would be overkill. However, if you want to go for top security, it is an option. In that case, almost all system commands will need a password and a sudo to execute. Note that this can endanger a lot of scripts however. So care should be taken while fiddling with the secure_path parameter.

3.1.5 Connect the Raspberry Pi to a local network

If the Raspberry is still functioning, shut it down:

```
$ sudo shutdown now
```

Now is the moment to connect the device to a network with an Ethernet cable. The device is sufficiently protected against intruders. We are connecting on a network that may be compromised, but due to the fact that root logon is blocked, that the default user id has changed and that the password of the user id is strong, we can take the risc.

— CONNECT THE DEVICE TO A LOCAL NETWORK —

Note however that the device should not be left powered on if you are not continuing the setup below. It is advisable in that case to power off the device after the shutdown is finished. After all, the less time one has to try to hack the device, the better our chances are to keep the device uncompromised.

3.1.6 Connect to the device with SSH

The first thing to do is to find the ip address of the device we want to install.

If you are on the Raspberry Pi itself, enter the command:

```
$ ip address show eth0
  | grep 'inet '
  | sed -e 's/^ *//'
  | cut -d " " -f 2
```

If you are using an external pc (say a Linux machine) then look up the hostname you have given your raspberry and execute the command (substitute <hostname> with the name you have chosen):

```
$ ping -c 1 -w 1 <hostname>.local
  | grep 'bytes from'
  | cut -d " " -f 4
```

and Bob's your uncle.

We have sufficient information to connect to the Raspberry Pi via the network. We can use a plethora of free packages on a pc such as OpenSSH, Putty, kitty, etc. Make your choice and ***connect to the IP address on port 22***.

From here on, we will enter all commands via the terminal program on a pc. The Raspberry Pi does not need a keyboard nor a display any longer. From now on, it will be a headless device.

3.1.7 Update software

We will now give the device a full update/upgrade:

```
$ sudo apt-get -y update
$ sudo apt-get -y dist-upgrade
$ sudo apt-get -y autoremove --purge
$ sudo reboot
```

Log on by means of the terminal program after the reboot has finished. We can safely assume that the latest patches are applied and that the system is up to date. The latest security patches should be applied.

3.1.8 Install an alternate MAC address

Since the MAC address of the device gives away to other devices on the network that our device is a Raspberry Pi we have a vested interest to change the MAC address. If you want to mimic a device that is often used in the region where you want to use the Raspberry Pi, prepare yourself before going to the region in question. Connecting the device on a network in the region where you want to have confidentiality, even for a couple of seconds, with the original MAC address can get you in trouble.

Please verify that changing a MAC address is legal in the region where you want to use the device. If it isn't, you shouldn't, or at least you do so at your own risc. Consider this as the authors of this document covering their asses.

An entire list of MAC address prefixes (to spoof a device of a certain manufacturer) can be found in an appendix of this document. Spoofing a MAC address of a device that is used frequently in the region gives you the best chances to stay under the radar (together with a common <hostname>).

The method we are using to set the MAC address permits to do so at startup BEFORE the network is started and hence, before the real MAC address is already used for a few milliseconds on the network. Our method prevents the provider or a local sniffing device to capture the real MAC address before it changes.

We suppose that the new MAC address for the eth0 interface should be <macaddress>. A MAC address that blends in gives you better chances.

Show the original MAC address:

```
$ ip link show eth0
  | grep 'link/ether'
  | sed -e 's/^ *//'
  | cut -d " " -f 2
```

Create a service file:

```
$ sudo nano /etc/systemd/system/changemac.service
```

Enter the following contents in the created service file:

```
[Unit]
Description=Set eth0 MAC address
Before=dhcpcd.service
[Service]
Type=exec
ExecStart=/usr/sbin/ip link set dev eth0 address <macaddress>
[Install]
WantedBy=networking.service
```

Save the file and exit the editor. Enable the service:

```
$ sudo nano /etc/systemd/system/changemac.service
```

3.1.9 Configure the wireless access point

Install the necessary software and activate the installed service:

```
$ sudo apt-get -y install hostapd
$ sudo systemctl unmask hostapd
$ sudo systemctl enable hostapd
$ sudo apt-get -y install dnsmasq
$ sudo apt-get -y install iptables
$ sudo DEBIAN_FRONTEND=noninteractive apt-get -y install netfilter-persistent
$ sudo DEBIAN_FRONTEND=noninteractive apt-get -y install iptables-persistent
```

Set up the network router. Edit the dhcp client deamon configuration file to define the wireless interface ip configuration:

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

and enter the following contents at the end:

```
interface wlan0
    static ip_address=192.168.128.1/24
    nohook wpa_supplicant
```

Save the file and exit the editor.

Edit the sysctl file to enable routing and ip masquerading:

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

Uncomment (remove the # in front of) the following text;

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

Save the file and exit the editor. Activate the new sysctl:

```
$ sudo sysctl -p
```

Change the firewall so to enable routing between the 192.168.128.x network and the 192.168.0.x network and save the new firewall rules so to make them permanent:

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo netfilter-persistent save
```

Filtering rules are saved to the directory /etc/iptables/. If in the future you change the configuration of your firewall, make sure to save the configuration before rebooting.

Edit the dnsmasq package to configure the DHCP and the DNS services it provides:

```
$ sudo nano /etc/dnsmasq.conf
```

Add the following text to the end of the file (listening intervace, dhcp addresses that will be attributedm local DNS domain, router alias):

```
interface=wlan0
dhcp-range=192.168.128.10,192.168.128.20,255.255.255.0,24h
domain=wlan
address=/gw.wlan/192.168.128.1
```

Note that if you only need one device to communicate with the Thor network, the dhcp range can be reduced so to prevent other devices from trying to access this access point.

Save the file and exit the editor. When operational, you should be able to find the wireless access point under the name gw.wlan

Enable the wifi radio on the Raspberry Pi in case it is blocked:

```
$ sudo rfkill unblock wlan
```

Edit the access point software configuration file:

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

We assume we will use a network name 'duckstersnetwork' and a password 'ducksterspassword'. Please choose your own network name so to make it blend into the environment where you will use the device. Choose a network password that is long and alternates alphanumeric signs with other characters. In our case, we have chosen the GB country code, but this can be adjusted to your choice in order to cloak the place where you are (we are not in GB). We will use wifi channel 11. We will use 2.4G (hw_mode=g). Should you need 5G then set hw_mode=a. Using 2.4G mimics an older device and should be less suspicious in some regions. On the other hand, in modern cities 5G seems more appropriate to stay hidden. Note that 5G should use channels like 36, 38, 40, 44, 46, 48, ... The channels that are permitted per country can be found on https://en.wikipedia.org/wiki/List_of_WLAN_channels.

Change the file as follows:

```
country_code=GB
interface=wlan0
ssid=duckstersnetwork
hw_mode=g
channel=11
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=ducksterspassword
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Note that, in a next version of this document the MAC address access control list will be used as an extra safeguard. This will make life of thirds that want to enter your access point so much more difficult.

Finally, reboot the device and check that the access point has come online:

```
$ sudo reboot
```

Conclusion

To do:

* deactivate IPv6

* make iptables rules stricter

* handle eeprom supply chain attack

MAC address prefix list

```
000000  Officially Xerox
000001  SuperLAN-2U
...
```