

Configuration Manual

MSc Research Project Cyber Security

David Collins Student ID: X09106081

School of Computing National College of Ireland

Supervisor: Dr Paul Stynes, Dr Vanessa Ayala-Riverra

National College of Ireland Project Submission Sheet School of Computing



Student Name:	David Collins
Student ID:	X09106081
Programme:	Cyber Security
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr Paul Stynes, Dr Vanessa Ayala-Riverra
Submission Due Date:	23/09/2021
Project Title:	Configuration Manual
Word Count:	1136
Page Count:	13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	David Collins
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	\square
Attach a Moodle submission receipt of the online project submission, to	\square
each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for	\square
your own reference and in case a project is lost or mislaid. It is not sufficient to keep	
a copy on computer.	

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

David Collins X09106081

1 Introduction

In my paper, Pen-Testing Framework for IoT Devices. A suitable framework for performing a penetration test on an ESP32 Microcontroller is proposed. This configuration manual describes the steps taken to set up the lab environment which would allow the test to be conducted. The tests are derived from the evolving legal regulations and demonstrate how these can be carried out practically.

2 Lab Set-Up

The high-level view of the Lab setup is shown in the Network Diagram. The subsequent parts of Section 2 comprise the instruction required to configure each of the components of this Network.

- OpenWrt router will act as a Firewall.
- ESP32 Nat Router will provide an isolated WiFi network to carry out the tests.
- Kali Linux will have the appropriate tools to carry out the test.
- The laptop will have any additional tools required.
- Raspberry pi zero -w set up to allow testing of the MQTT protocol.
- IoT Device under test.

The ESP32 Nat Router is configured using the WiFi Manager On the Laptop connect to this network the Initial settings has this open network. Once connected navigate to 192.168.4.1. New values for SSID, Password can be Entered. Add the credentials for the OpenWrt Server allowing Internet connection, firewall rules on the OpenWrt router permitting. Static IP settings can be entered if required. Connect the Kali Linux, Raspberry Pi zero, and Laptop to the ESP32 Nat Router WiFi.

2.1 Network Diagram



Figure 1: Network Lab Diagram

2.2 OpenWrt Server

- $\checkmark\,$ connect to Web Gui 192.168.1.1
- $\checkmark\,$ go to Networks Wireless
- $\checkmark\,$ create a WiFi Network by clicking add.
- $\checkmark\,$ go to Device Configuration
- $\checkmark\,$ open Advance Settings.
- $\checkmark\,$ fill in country code to meet legal regulations.
- $\checkmark\,$ in General, Security enters SSID for the name of the network.
- $\checkmark\,$ in Wireless Security select encryption method WPA2-PSK
- $\checkmark\,$ in Wireless Security select a cypher auto.
- $\checkmark\,$ in Wireless Security enter a wireless password.
- $\checkmark\,$ click save and apply

<u>g</u>	Gener Channe	ric MAC80211 802.11 el: 11 (2.462 GHz) Bitra	bgn (radio0) te: 56.5 Mbit/s				🖸 Scan 🎦 Add
	# SSI 84% BS	D: OpenWrt Mode: Mas SID: 54:36:9B:33:F4:C7	ter Encryption: WPA2	PSK (CCMP)		Ø Disable	Z Edit Remove
Ass	eninted	Stations					
	SSID	MAC-Address	IPv4-Address	Signal	Noise	RX Rate	TX Rate
all	SSID OpenWrt	MAC-Address AC:67:B2:37:33:64	IPv4-Address	Signal	Noise 0 dBm	RX Rate 6.0 Mbit/s, MCS 0, 20MHz	TX Rate 1.0 Mbit/s, MCS 0, 20MHz
لله لله	SSID OpenWrt OpenWrt	MAC-Address AC:67:B2:37:33:64 CC:AF:78:08:90:16	IPv4-Address 192.168.1.136 ?	Signal -44 dBm -50 dBm	Noise 0 dBm 0 dBm	RX Rate 6.0 Mbit/s, MCS 0, 20MHz 117.0 Mbit/s, MCS 14, 20MHz	TX Rate 1.0 Mbit/s, MCS 0, 20MHz 78.0 Mbit/s, MCS 12, 20MHz
لله لله لله	SSID OpenWrt OpenWrt OpenWrt	MAC-Address AC:67:B2:37:33:64 CC:AF:78:08:90:16 DC:A6:32:66:69:31	IPv4-Address 192.168.1.136 ? 192.168.1.103	Signal -44 dBm -50 dBm -46 dBm	Noise 0 dBm 0 dBm 0 dBm	RX Rate 6.0 Mbit/s, MCS 0, 20MHz 117.0 Mbit/s, MCS 14, 20MHz 24.0 Mbit/s, MCS 0, 20MHz	TX Rate 1.0 Mbit/s, MCS 0, 20MHz 78.0 Mbit/s, MCS 12, 20MHz 1.0 Mbit/s, MCS 0, 20MHz
له له له له	SSID OpenWrt OpenWrt OpenWrt OpenWrt	MAC-Address AC-67:B2:37:33:64 CC:AF:78:08:90:16 DC:A6:32:66:69:31 9C:9C:1F:E3:E9:6C	IPv4-Address 192.168.1.136 ? 192.168.1.103 192.168.1.103	Signal -44 dBm -50 dBm -46 dBm -54 dBm	Noise 0 dBm 0 dBm 0 dBm 0 dBm	RX Rate 6.0 Mbit/s, MCS 0, 20MHz 117.0 Mbit/s, MCS 14, 20MHz 24.0 Mbit/s, MCS 0, 20MHz 6.0 Mbit/s, MCS 0, 20MHz	TX Rate 1.0 Mbl/s, MCS 0, 20MHz 78.0 Mbl/s, MCS 12, 20MHz 1.0 Mbl/s, MCS 0, 20MHz 50.5 Mbl/s, MCS 6, 20MHz

Figure 2: OpenWrt Showing Connected devices

2.3 ESP32 Nat Router

Carry out the following steps to create the ESP32 Nat router.

Step 1: Download the espressif Download tool. from https://www.espressif.com/en/support/download/other-tools

SPIDownload	HSPIDownload	RFConfig	GPIOConfig	MultiDownload			
		,					
C:\Users\				ESP32\repeater\bootloader.bin		@	0x1000
C:\Users\				E\$P32\repeater\ esp32_nat_router	bin	0	0x10000
C:\Users\				\ESP32\repeater\partitions_example	bin	ø	0x8000
						0	
						0	
						0	
						0	
⊃ 20MHz ⊃ 80MHz	O QIO O QOUT	 32Mbit 64Mbit 128Mbit 	DETE flash 5Eh : flash 4016f QUAI crysta 40 M	CTED INFO vendor: NVA NVA evilo: 5;32Mbit sk:			
Download Panel FINISH 完成	1 2: 246F28A9F7F5 5 2: 246F28A9F7F6 E	TA: 246F28A9 THERNET: 246	0F7F4 5F28A9F7F7				
START STOP FRASE COM:		COM18	COM18				

Figure 3: Download tool settings

Step 2: Download the binaries from Github https://github.com/martin-ger/esp32_nat_router/tree/master/build.

Step 3: Install the binaries at the following locations:

- bootloader.bin @0x1000.
- esp32_nat_router.bin @ 0x10000

- partitions_example.bin @8000
- $\checkmark~$ set SPI Speed to 40MHz.
- $\checkmark\,$ SPI Mode to DIO
- $\checkmark\,$ Flash size 32Mbit.
- $\checkmark\,$ Click start and wait for the download to complete.

2.4 MQTT Broker

Download the Raspbian Jessie image and install it to SD Card. Attach SD card to Raspberry Pi zero w. This device comes with added Wireless and Bluetooth Connectivity. Connect Monitor and Bluetooth Mouse and Keyboard to the PI. Open a command prompt and update the software with the commands. sudo apt update sudo apt dist-upgrade. To install an MQTT Broker and MQTT Client on the Raspberry PI zero the following steps are used.

- ✓ wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
- $\checkmark\,$ sudo apt-key add mosquitto-repo.gpg.key
- \checkmark cd /etc/apt/sources.list.d/
- ✓ sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
- $\checkmark~$ apt-get update
- $\checkmark~$ apt-get install mosquitto
- \checkmark apt-get install mosquitto-clients

To install NODE-RED on the PI Zero run the following installation script

 $bash>(curl - sL \ https: //raw.githubusercontent.com/node - red/raspbian - deb - package/master/resources/update - nodejs - and - nodered)$

	A
Once Node-RED has started, point a browser a	t http://192.168.1.179:1880
On Pi Node-RED works better with the Firefox	or Chrome browser
Use node-red-stop	to stop Node-RED
Use node-red-start	to start Node-RED again
Use node-red-log	to view the recent log output
Use sudo systemctl enable nodered.service	to autostart Node-RED at every b
oot	
Use sudo systemctl disable nodered.service	to disable autostart on boot
To find more nodes and example flows - go to	http://flows.nodered.org
Starting as a systemd service.	
14 Jul 21:47:03 - [info]	
Welcome to Node-RED	
=================	
14 Jul 21:47:03 - [info] Node-RED version: v	1.3.5
14 Jul 21:47:03 - [info] Node.js version: v	14.17.3
14 Jul 21:47:03 - [info] Linux 5.10.17+ arm	LE
14 Jul 21:47:12 - [info] Loading palette nod	es
14 Jul 21:47:35 - [info] Settings file : /h	ome/pi/.node-red/settings.js
14 Jul 21:47:35 - [info] Context store : 'd	efault' [module=memory]

Figure 4: NODE RED Start Stop Commands

\leftarrow \rightarrow C \blacktriangle Not security	re 192.168.1.179:1880/#flow/ad	0b6280.6e543			
Node-RED					
Q filter nodes	Flow 1	Flow 2	+	≣	
✓ common				^	
inject debug	<u>ا</u>	/sensor1/temperature msg.payload			
complete catch)	/sensor1/humidity			
		⇒ test /sensor1/temperature /) ■ connected			
b link in		⇒ sensor1/in:true 1			
link out		sensor1/in:false mqtt) connected			

Figure 5: NODE RED MQTT Sketch

Start Node-Red. Create the MQTT sketch and configure the nodes with topic and configuration parameters.

Table 1: Configure MQTT topic and security.

				9, 11		Flow 1	Edit mott in node a	Edit mqtt-bro	oker node		
_				~ 0	common		Delete			Cancel	Update
-<					A Date of the local date of th		O Properties				o B
Q filter nodes	Flow 1 Flow 2	Edit mqtt in node	1		y nyes	le le					
~ common		Delete	Cancel Done		debug		Name	sensor1/hum	idity		
🔅 inject 🔉	/sensor1/temperat	© Properties			cetch		Connection		Security	Messages	
debug	Connected	@ Server	sensor1/humidity 🗸		status D		@ Server	http://192.168	3.1.179	Port 1883	
complete	/sensor1/humidity	I Topic	/sensor1/temperature		ink in			Use TLS			
catch P	test -	⊕ QoS	2 *		Enk out		O Protocol	MQTT V3.1.	1	~	
e link in o		0+ Output	auto-detect (string or buffer)	2	comment		Sclient ID	MQTT V3.1	(legacy) 1		
fink out	Sensor Init and	Name	Name	~ n	function		🕏 Keep Alive	MQTT V5			
comment	sensor1/in:taise				f function		i Session	🗹 Use clean s	session		

2.5 Kali Linux

To conduct the penetration test extra tools are installed on Kali Linux. For this Lab environment, a Raspberry pi 4 Model B 8GB Ram was used. The kali image was downloaded from https://www.kali.org/get-kali/#kali-arm and flashed to a 32 GB sd card using Balana Etcher. On start-up, Kali was updated with the commands.

sudo	apt	update
sudo	apt	upgrade

2.5.1 Kali Tools

esptool git clone https://github.com/espressif/esptool.git

cd esptool

pip install –user -e.

to test installation use command python3 esptool - h (Fredrik; 2021)

RouterSploit sudo apt-get install python3-pip

git clone https://www.github.com/threat9/routersploit.

cd routersploit

python3 - m pip install - r requirements.txt(kleo; 2018)

Wifite git clone https://github.com/derv82/wifite2.git.

cd Wifite

sudo python3 setup.py install

To test installation sudo wifite -h will provide a list of all available commands

(derv; 2018)

nmap nmap should come preloaded on Kali Linux if not it can be installed with the following command.

sudo apt - get install -ynmapto test nmap -version(Lyon; 2012)

WireShark sudo apt - get install -y wireshark to test wireshark -h tshark -h(Wireshark \cdot Go Deep.; 2021)

Visual Studio Code Open a web browser and navigate to https://code.visualstudio.com/download

and download the latest armhf.deb file.

to install run the command sudo apt install ./code-xxx-armhf.deb

PlatformIO

- Open VSCode Package Manager.
- Search for the official platformio ide extension.
- Install PlatformIO IDE check that toolbar is present.



(Kravets; 2020)

3 ESP32 IoT Test Subject

The IoT device used to test the penetration test framework uses an ESP32 Microcontroller to act as a Web Server and MQTT client. The following steps were used to program the project. Open Visual Studio Code and create a PlatformIO application. Add the following details to the config file. The libraries can be added manually or by using the Libraries tab of PIO Home. The full code can be found on my GitHub Repository https://github.com/2davecollins/esp. (Collins; 2021) For this test, sensitive data was extracted to separate credentials.h file for evaluation. Other options for storing credentials would use the Preferences library. (Santos; 2021). The initial plan for this paper was to test an insure IoT devise add security features secure MQTT, HTTPS and test again.



Figure 7: PIO Home add Libraries

Table 2: Configure Credentials platformIO



4 Penetration Test

The following checks are carried out as part of the penetration test.

- Identification of IoT Device.
- Ensure Firmware has not been tampered with.
- Perform Static code analysis on firmware.
- Check router for vulnerabilities (default credentials).
- Network Scan for open ports.
- Check for data exposure using Wireshark.

4.1 ESP32 Identification

cd /home/kali/esptool

use esptool chip id function to report on esp32 chip identification.

python3 esptool.py chip_id

additional information can be found using esptool flash id function.

python3 esptool.py flash_id

4.2 ESP32 Check firmware for tampering

cd /home/kali/esptool

check binary at location 0x1000 against local file bootloader.bin

```
python 3 - m esptool verify_flash - -dif yes 0x1000 bootloader.bin
```

check binary at location 0x10000 against local file esp32_nat_router.bin

 $python 3 - m esptool verify_flash - -dif yes 0x10000 esp32_nat_router.bin$

check binary at location 0x8000 against local file partitions_example.bin

 $python 3 - m \ esptool \ verify_flash \ --dif \ yes \ 0x10000 \ partitions_example.bin$

alternately the hash of the binary can be found by loading it into the Cutter tool and comparing the hash against a previously recorded value.

Info					
File:	C:\Users\2dave\WCIRL\binary\flash_cc	FD:	3	Architecture:	N/A
Format:	any	Base addr:	0x00000000	Machine:	N/A
Bits:	0	Virtual addr:	N/A	05:	N/A
Class:	N/A	Canary:	N/A	Subsystem:	N/A
Mode:	r-x	Crypto:	N/A	Stripped:	N/A
Size:	4 MB	NX bit:	N/A	Relocs:	N/A
Type:	N/A	PIC:	N/A	Endianness:	N/A
Language:	N/A	Static:	N/A	Compiled:	N/A
		Reiro:	N/A	Compiler:	N/A
Hash	Certifica	tes		Version info	20
Hash MD5:	Certifical ES f4f0ac5c8e4ca1ae03588a5435c2c263	ies		Version info	25
Hash	Certifical Certifical	ies 163fb29c		Version info	25
Hashe MD5: SHA1: SHA256:	Certifical ES f4f0ac5c8e4ca1ae03588a5435c2c263 31a6db4d44b0e88547647d3e67b535b33 2ca6280d7b3e07015f4c14358f5f1e03a5	163fb29c e498143293ee8	b24136a47fc069e6b	Version info	25
Hash MD5: SHA1: SHA256: Entropy:	Certifical Certif	163fb29c 143293ee8	b24136a47fc069e6b	Version info	25
Hashe MD5: SHA1: SHA256: Entropy: Analy Functions:	Certifical Certifical Certifical Certifical Certifical Certifical Certifical Certifical Carterial Contemporation of the contemporation of the con	163fb29c 1e498143293ee8	b24136a47fc069e6b	Version info	25
Hashe MD5: SHA1: SHA256: Entropy: Analy Functions: X-Refs:	Certifical Certif	163fb29c 163fb29c 16498143293ee8	b24136a47fc069e6b	Version info	2S

Figure 8: Reading binary hash using cutter tool

4.3 Perform Static code analysis on firmware

Open up the firmware using Visual Studio Code with platformIO extension. Open PIO Home and navigate to the Inspect Tab and Toggle Inspect Memory and Check Code and initiate test with Inspect Button.

(で) () () () () () () () () () (Projects\ESP32-WEB-SER\ ESP32 240MHz, 320 KB RAM, 1 MB Flash	/ER-SPIFFS env:esp32dev	🖻 Reveal C Refresh
Home	∠ Statistics 몲 Explorer	🖉 Symbols 🛛 🗄 Sections 🏵	Defects
Projects	87%	68%	73
Inspect			
Г	RAM	Flash	Defects

Figure 9: Result Statistics

(j) 			1	•					
ل Home	Defects Summary					Top Defects			
	Component	High	Medium	Low		Level	Message		
Projects	src		73			MEDIUM	header is missing header guard		
Q						MEDIUM	#includes are not sorted properly		
Inspect	Total		73			MEDIUM	consider replacing 'long' with 'int32'		
Ш						MEDIUM	consider replacing 'long' with 'int32'		
Libraries						MEDIUM	do not declare C-style arrays, use		
±=							std::array<> instead		
Boards									

Figure 10: Top Defects Summary

4.4 Check router for vulnerabilities (default credentials) using RouterSploit

cd /home/kali/routersploit

python3 rsf.py

 $rsf > use \quad scanners \, autopwn$

rsf > (AutoPwn) show options

 $rsf > (AutoPwn) \quad set \, target \, < i paddress of router >$

rsf > (AutoPwn) run

4.5 Network Scans for open ports

Use nmap to scan a network for open ports. Use a basic scan to determine the IP addresses on the network.

nmap 192.168.4.0/24

using zenmap for a graphical view of a network using a shorter range and searching for MQTT protocol.

Zenmap Scan Tools Profile Help 192.168.4.1-7 Target: nmap -p 1883 -T4 -A -v --open 192.168.4.1-7 Command: Hosts Services Nmap Output Ports / Hosts Topology Host Details Scans Service Hosts Viewer Fisheye Controls ٠ http mqtt 192.168.4.5 192.168.4.7 192.168.4.1

 $nmap - p \ 1833 \ T4 - A - v - open \ 192.168.4.1 - 7$

Figure 11: nmap scan for MQTT

👁 Zenmap											
Sc <u>a</u> n <u>T</u> ools <u>P</u> rofile <u>H</u> elp											
Target: 192.168.4.1-7											
Command: nmap -p 1883 -T4 -A -vopen 192.168.4.1-7											
Hosts Services	Nmap Output Ports / Hosts Topology Host Details Scans										
OS ◀ Host ▲	Port Protocol State Service Version										
192.168.4.1	😑 1883 tcp open mqtt										
192.168.4.5											
3 192.168.4.7											

Figure 12: nmap scan results

4.6 Check for data exposure using WireShark

Open Wireshark on Kali Linux. Start NODE-Red and wait for traffic from the MQTT client on IoT under test to be observed at debug node on Node-Red. Open a web browser and observe temp and humidity sensor data from the Web Server. Stop Wireshark and examine the pcap file for exposed data.

Wireshark - Follow TCP Stream (tcp.stream eq 1) - Ethernet	-	o x							
<pre>.~.[["topic":"debug","data":{"id":"c4f1fe43.49fba","z":"ad0b6280.6e543","path":"ad0b6280.6e543","name":"","topic":"/sensorl/ temperature","property":"payload","msg":"27.80","format":"string[5]"}}.{Topic":","topic":","bug","data": {"id":"c4f1fe43.49fba","z":"ad0b6280.6e543","path":"ad0b6280.6e543","name":"","topic":"/sensorl/ humidity","property":"payload","msg":"41.00","format":"string[5]"}}.{Topic":",bb","data":","topic":",bc","data": ["topic":"debug","data":{"id":"c4f1fe43.49fba","z":"ad0b6280.6e543","name":"","topic":",bb","data":1626295656535}].~.] [{"topic":"debug","data":{"id":"c4f1fe43.49fba","z":"ad0b6280.6e543","name":"","topic":",sensorl/ temperature","property":"payload","msg":"27.80","format":"string[5]"}}.{Topic":",bb","data": ["id":"c4f1fe43.49fba","z":"ad0b6280.6e543","name":"","topic":",sensorl/ humidity","property:":payload","msg":"41.00","format":"string[5]"}].*.[[{Topic":",sensorl/ temperature","property:":payload","msg":"41.00","format":"string[5]"}].*.[[{Topic":",sensorl/ humidity","property:":payload","msg":"27.80","format":"string[5]"}].*.[[{Topic":",sensorl/ humidity","property:":payload","msg":"27.80","format":"string[5]"}].*.[[{Topic":",sensorl/ humidity","property:":payload","msg":"27.80","format":"string[5]"}].*.[[{Topic":",sensorl/ temperature","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ temperature","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ temperature","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ temperature","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ humidity","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ humidity","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ humidity","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ humidity","property::payload","msg":"27.80","format":"string[5]"},"topic":",sensorl/ humidity","property::payload","msg":"27.80","format":"string[5]"},"t</pre>									
8 clert pitz, 0 server pitz, 0 zonz.									
Entre conversation (1658 bytes) Show data as ASCII	_	Stream 1 👽							
Find:		Find Next							
Filter Out This Stream Print Save as Back Cl	ose	Help							

Figure 13: Wireshark exposed data

References

Collins, D. (2021). 2davecollins esp, [Online] Available: https://github.com/ 2davecollins/esp. [Accessed on: August, 10, 2021].

derv (2018). derv82 wifite, [Online] Available: https://github.com/derv82/wifite. [Accessed on: August, 13, 2021].

- Fredrik, A. (2021). espressif esptool, [Online] Available : https://github.com/ espressif/esptool. [Accessed on: July, 12, 2021].
- kleo (2018). threat9 routersploit, [Online] Available: https://github.com/threat9/ routersploit. [Accessed on: August, 14, 2021].
- Kravets, I. (2020). Professional collaborative platform for embedded development, [Online] Available: https://docs.platformio.org/en/latest/.
- Lyon, G. (2012). Nmap reference guide, [Online] Available : https://nmap.org/book/man.html. [Accessed on: August, 10, 2021].
- Santos, R. (2021). Esp32 save data permantly using preferences library, [Online] Available : https://randomnerdtutorials.com/ esp32-save-data-permanently-preferences/. [Accessed on: August, 10, 2021].
- Wireshark · Go Deep. (2021). [Online] Available: https://www.wireshark.org/. [Accessed on: August, 13, 2021].