FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Rogue Access Point on a WiFi-capable microcontroller
Student:	Vanda Hendrychová
Supervisor:	Ing. Jan Baier
Study Programme:	Informatics
Study Branch:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2020/21

Instructions

The ESP8266 is a low-cost WiFi microchip with the full TCP/IP stack and microcontroller capability. It is frequently used in IoT applications.

1. Explore available versions of ESP8266 modules and compare their possibilities with the intended purpose in mind.
2. Analyze compatibility of the modules with different WiFi operating modes, more specifically with the Client+AP mode.
3. Create a prototype device for a "Rogue AP" that will provide functionalities to:
   - make a detailed scan of nearby WiFi networks to identify a target;
   - connect to a selected network in the Client mode to provide connectivity;
   - disguise itself as a selected target network and perform AP operations;
   - execute a simple packet capture and/or perform an analysis of network traffic.
4. Test and evaluate the device in an approved real-world scenario.

References

Will be provided by the supervisor.

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 23, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Rogue Access Point on a WiFi-capable microcontroller

Vanda Hendrychová

Department of Computer Systems
Supervisor: Ing. Jan Baier

June 4, 2020

Acknowledgements

I would like to thank my supervisor Ing. Jan Baier for his quick and helpful advices. I would also like to thank my boyfriend Pavel for his support.

Declaration

In Prague on June 4, 2020 …………………

Citation of this thesis

Hendrychová, Vanda. *0.0.0.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Wi-Fi sítě poskytující přístup k internetu nás v poslední době obklopují na každém kroku. Kvůli principu jejich fungování je ale velmi obtížné zaručit, kdo je poskytovatelem sítě, ke které se připojíme. Některé přístupové body mohou být nastražené neznámým útočníkem například za účelem sledování provozu cizí sítě či krádeže citlivých dat. Tato bakalářská práce zkoumá možnost vytvoření takového falešného přístupového bodu pomocí cenově dostupných Wi-Fi čipů s nízkou spotřebou energie (ESP8266). Hlavní důraz je kladen na možnost paralelizace činností mezi více zařízeními. Díky vícevrstvé architektuře a využití REST API návrh umožňuje využití různých implementací.

Klíčová slova internet věcí, falešný přístupový bod, wi-fi, mikrokontrolér, ESP8266, směrovač, přístupový bod, zabezpečení bezdrátové sítě, zlé dvojče

Abstract

Wi-Fi networks providing internet access are surrounding us wherever we go. Due to the mechanism of their operation, it is quite difficult to guarantee who is the real provider of the network to which our device is connected. Some access points might be tampered with by an unknown attacker for example to monitor the traffic of a foreign network or to steal sensitive data. This thesis examines the possibility of creating such a rogue access point using affordable Wi-Fi chips with low power consumption (ESP8266). The main focus is on workload parallelization among multiple devices. The prototype features a multilayer architecture which enables the use of multiple different implementations on various devices.

Keywords internet of things, rogue access point, wi-fi, microcontroller, ESP8266, router, access point, wireless security, Evil Twin

Contents

List of Figures

CHAPTER 1

Introduction

In the twenty years since its introduction, Wi-Fi has gained enormous popularity. In the beginnings, the danger of someone attacking the wireless communication was quite negligible due to both the novelty of the technology and the low amount of possible targets. Contrarily, nowadays most people do not have just one but rather multiple Wi-Fi-capable devices. Not only do most phones and laptops now come with integrated wireless cards, but the Internet of Things is becoming more common as well. The growing amount of Wi-Fi-capable devices goes hand in hand with the increasing sensitivity of transmitted information. Both of these factors made the wireless devices a desired target for hackers.

Over the years, a variety of attacks have arisen. For instance, an attacker might monitor someone's conversation in an eavesdropping attack or pretend to be someone else with MAC spoofing. This thesis deals with another wireless network attack — a Rogue Access Point. In general, any wireless access point installed in a network without the knowledge of its administrator is a rogue access point. Yet not all rogue access points are malicious. For example, an employee might want to extend the company's wireless network to get more reliable internet access in their office. When the access point becomes malicious, it is sometimes called an Evil Twin. Such a device might provide an unsecured gateway into an internal network. In another scenario, the device can mimic a public hotspot and perform various man-in-the-middle attacks.

With the Internet of Things on the rise, it is now easier than ever to get access to low-cost Wi-Fi chips and place them without raising much suspicion. This thesis examines the possibilities of a frequently used inexpensive Wi-Fi-capable microcontroller in the context of creating rogue access points. We have developed a prototype device for a Rogue Access Point.

Chapter 2 presents the theory and related projects. The prototype requirements and its design are discussed in Chapter 3. Chapter 4 describes the realisation of the design. A series of tests were performed to examine the prototype functionality. They are described in Chapter 5.

Chapter 2

Analysis

2.1 Wi-Fi

Wi-Fi is a wireless communication technology managed by a non-profit organization, The Wi-Fi Alliance[1]. It is based on the IEEE 802.11 standards. The first version of the IEEE 802.11 standard was released in 1997[1]. Since then, many different versions of the standard were released, typically identified by one or two letters such as 802.11b or 802.11ac. Among other things, the standards specify frequencies used for communication. The earlier versions of the standard are tied with the 2.4 GHz frequency. In a new naming system from the Wi-Fi alliance, the 802.11n technology (2.4 GHz) is known as Wi-Fi 4. Newer standards based on 802.11ac technology and 5 GHz frequency are called Wi-Fi 5. Lastly, the currently newest technology is Wi-Fi 6 which supports 802.11ax and 6 GHz communication frequency.[2]

A part of the IEEE 802 standards is the **medium-access-control** (MAC) layer. It is a sublayer of a data link layer in the OSI network model. It provides a control abstraction of the physical layer. To distinguish different devices, local network addresses called media access control addresses (MAC address) are used. The MAC address consists of six bytes with first three bytes identifying the manufacturer of the network interface controller (NIC). The second triplet is assigned by the manufacturer to provide a unique address for each new device.[3]

The 802.11 standard uses management frames for various actions such as authentication and deauthentication. However, the management frames are not authenticated which allows them being faked. Furthermore, deauthentication and disassociation frames cannot be rejected.[4]

[1]https://www.wi-fi.org/

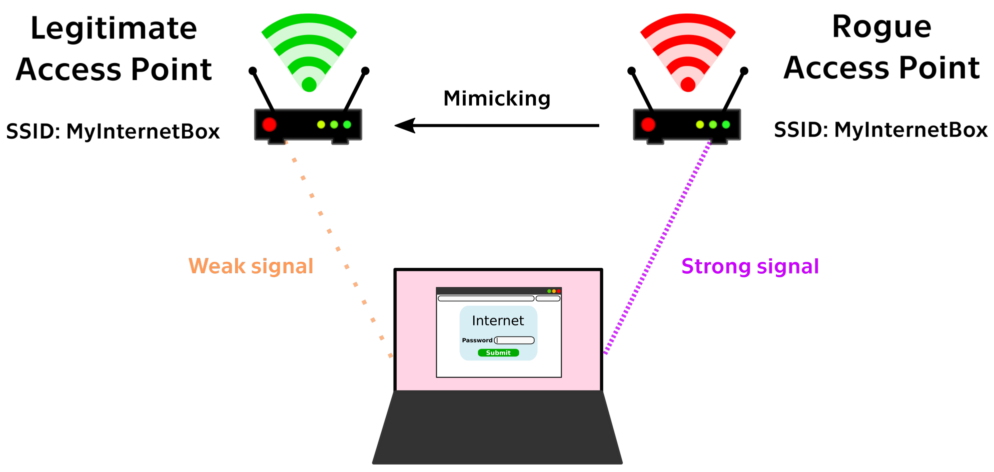


Figure 2.1: Rogue access point mimicking a legitimate AP

2.2 Rogue Access Point

When a client device is presented with a choice of two access point with the same credentials, it typically chooses the one with stronger signal. Regardless of how convenient it might be that the user devices tend to choose an access point with the strongest signal, there is a risk that such access point is owned by an attacker. Figure 2.1 shows an example of a rogue access point mimicking a legitimate AP that deceives the client device to connect to it. Although a rogue access point is usually referred to as any unauthorized access point[5], there are several other scenarios described in Section 2.2.1. The threats posed by rogue access points are further discussed in Section 2.2.2.

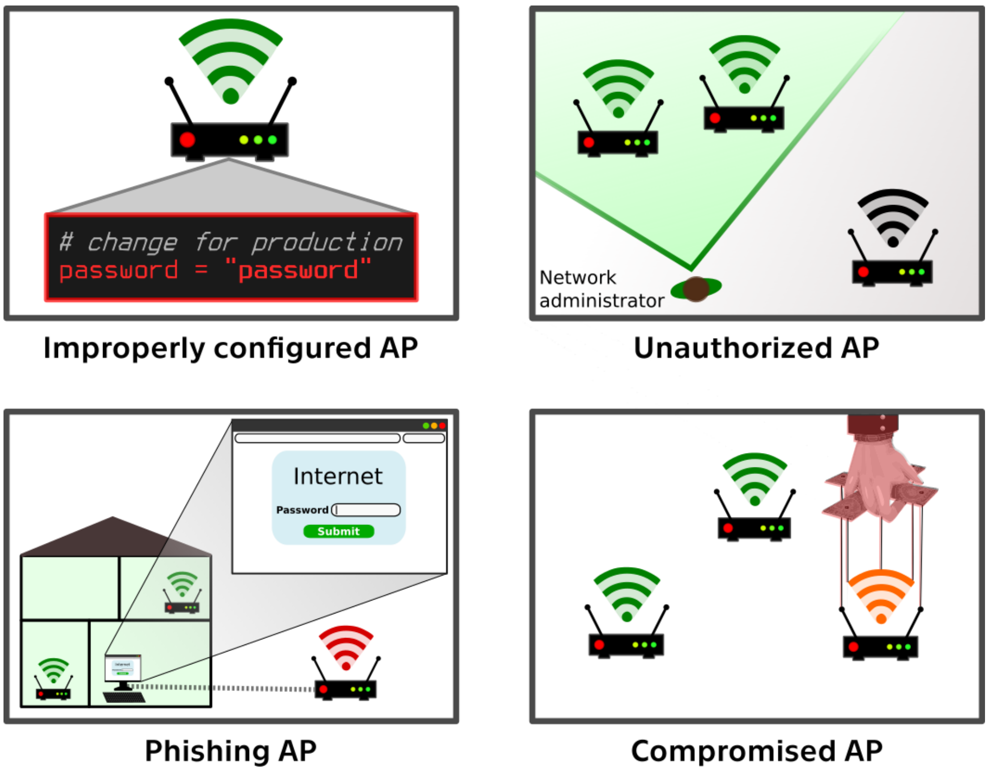
2.2.1 Classification

This section describes the most common types of rogue access points. The four described types are illustrated by Figure 2.2.

An **Unauthorized Access Point** is an access point installed on a network without the knowledge and/or authorization of the network's administrator[5]. Not all unauthorized access points are placed by hackers. For instance, an employee might set up one to get better wireless connectivity in his/her office. Even though the original intent might not have been malicious, an unauthorized access point always poses a security risk.

Some access points managed by the network's administrator might turn into a rogue access point as well. In the case of an **Improperly Configured Access Point**, the device might become vulnerable because of a configuration mistake, a faulty driver or a software update[5].

A **Compromised Access Point** is a legitimate access point with security features enabled. Nevertheless, its credentials have been stolen by an attacker



```
# change for production
password = "password"
```

Figure 2.2: Rogue access point classification

who can then gain access to sensitive data through this channel. The possible reason for the attacker's success include the usage of a vulnerable security protocol such as WEP or securing the access point with a weak password[5].

A **Phishing Access Point** is an access point outside of the network which attempts to trick users to connect to it by masquerading as a legitimate access point or replaying beacon frames from other access points. Consequently the access point can be used to steal users' credentials or perform man-in-the-middle attacks (see Section 2.2.2). Networks using a security protocol without mutual authentication are particularly vulnerable because it is easier to imitate them[5].

2.2.2 Risks

This section describes what are the possible malicious uses of rogue access points, some of which had already been hinted in the previous section.

Firstly, the rogue access points might be used to perform **denial-of-service** (DoS) attacks. A denial-of-service attack is defined as a situation in which actions of an attacker prevent legitimate user from accessing network resources.[6] The rogue access point can mimic a legitimate access point but not provide access to the original network. By dropping the incoming packets, the device creates a simple denial-of-service attack[7]. Having said that, modern operating systems are capable of detecting that the network has no internet connection. Still, the device might perform another DoS attack which consists in sending deauthentication frames to nearby devices. This was shown to drastically decrease the TCP performance[4].

Another risk is providing a **gateway into an internal network**. While the wired network is constrained by physical boundaries, the same does not apply to the wireless network. Even if the known infrastructure is secured, a rogue access point plugged directly into an internal network bypasses any security mechanisms which were applied at the network edge.[7]

Perhaps the most dangerous is the threat of **man-in-the-middle** (MITM) attacks, where the attacker intercepts and controls a conversation between two parties. When a user connects to a rogue access point, all of the user's communication passes through the hands of the attacker thus possibly creating a MITM scenario[8]. One of the forms of a man-in-the-middle attack is a social engineering practice called **web browser phishing**. When the user connects to the access point and attempts to reach some website, the attacker can present the user with a fake version of the site[8]. Such phishing websites usually log users' credentials. Among the most popular phishing targets are online banking[8] and online commerce sites such as eBay and PayPal[9]. Phishing attacks also serve as a way of distributing malware[9].

2.3 Microcontroller

The ESP8266[2] is a family of inexpensive WiFi-enabled microchips produced by a Shanghai based company Espressif Systems. All versions of the chip support 802.11 b/g/n standards and are able to operate in Station and AP WiFi mode simultaneously. The device is able to connect to WEP, WPA-PSK and WPA2-PSK secured networks and open networks. Apart from the wireless functionality, the device integrates a Tensilica L106 32-bit RISC microprocessor with Harvard architecture and a running frequency of 80 MHz. The processor is an ultra-low-power MCU with average power consumption of 71 mA peaking at 500 mA. It has tree different sleep modes – the modem sleep in which it consumes around 20 mA, the light sleep with consumption around 2 mA and deep sleep with only 0.02 mA power consumption.

The two most widely available versions are ESP-01 and ESP-12F, both depicted in Figure 2.3. ESP-01 is the very first version of the ESP8266 module. Despite having less GPIO pins and flash memory,the microcontroller itself would be suitable for the intended us. However, the ESP-12F was selected instead because it is available with a convenient MicroUSB connector. More

[2]https://www.espressif.com/en/products/socs/esp8266/overviewMedia

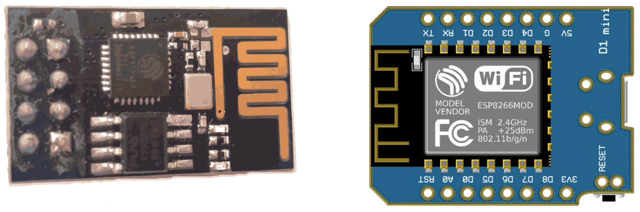


Figure 2.3: ESP-01 (left) and ESP-12F D1 mini (right)

specifically, the ESP-12F D1 mini board was used due to the availability of various shields which makes the device easily extensible.

2.3.1 Software Development Kits

In 2014, Espressif Systems released an official software development kit for programming its' chipsets[10]. The **Espressif SDK**[3] is royalty-free and published on GitHub. There used to be two variants available for ESP8266 — the ESP8266 NonOS SDK and the ESP8266 RTOS Software Development Kit[11]. However, since December 2019, the ESP8266 NonOS SDK is deprecated and no longer updated by Espressif Systems[12].

Since the ESP8266 RTOS SDK is outdated, Espressif Systems is currently working on replacing it with a new project, which should match the style of ESP-IDF[4] — an official framework for ESP32[13].

ESP-Open-SDK[5] is an integrated software development kit which aims to be as open as possible. It builds on the Xtensa lx106 toolchain and previously mentioned ESP NonOS SDK. The SDK includes a few other tools based on other open-source projects, such as a TCP/IP stack based on lwIP[14].

ESP-Open-RTOS[6] is a community developed alternative to the ESP8266 RTOS SDK. Although originally based on the Espressif's project, it was significantly changed throughout the years and the authors claim that it is now substantially different from the ESP8266 RTOS SDK. The project now uses the Xtensa toolchain from ESP-Open-SDK[15].

NodeMCU[7] is an open-source firmware and development kit based on the Lua scripting language. It is implemented in C and originally layered on the Espressif Non-OS SDK. The project was initially built only for NodeMCU derivates of the ESP8266 boards but nowadays the firmware supposedly supports all ESP modules[16].

[3]https://www.espressif.com/en/products/software/esp-sdk/overview
[4]https://github.com/espressif/esp-idf
[5]https://github.com/pfalcon/esp-open-sdk
[6]https://github.com/SuperHouse/esp-open-rtos
[7]https://github.com/nodemcu/nodemcu-firmware

Espruino[8] is JavaScript based firmware for various embedded devices. It has supported ESP8266 since 2015. The project also provides a JavaScript IDE and a variety of modules[17]. Espruino also sells their own devices, including Espruino WiFi which integrates the ESP8266 chip and an ARM Cortex M4 processor[18].

The **Arduino**[9] project develops a C++ based firmware with some of the core functions common for multiple boards. Even though it is possible to connect the ESP8266 to an Arduino microcontroller as an external network card, the ESP8266 can also be programmed directly using the ESP8266 Arduino core. The core builds on top of the Espressif SDK and uses the Xtensa gcc toolchain for compilation and Esptool.py for flashing[19].

For the purposes of this thesis, the Arduino framework was used. The main advantage is its library support and familiarity of the programming language (C++).

2.4 Related work

This section covers related projects and available products including both open-source and commercial solutions.

Wi-Fi Pineapple[10] is a commercial device sold by the company Hak5. It comes two variants — *Tetra* and *Nano*. Both devices provide a complex penetration testing kit with a web and mobile interface. Pineapple Nano is a pocket sized device featuring 2.4 GHz network connectivity, two antennas and a MicroSD card slot. It is currently sold for 100 USD[20].

Wi-Fi Pineapple Tetra, offered for 200 USD, is significantly larger but supports both 2.4 GHz and 5 GHz networks and has four long range antennas[20]. The Tetra version can be used with a Raspberry Pi running Kali Linux for creating what is commonly reffered to as the Pineapple Pi[21].

Wifiphisher[11] is a penetration test framework focused on rogue access points. It features several association methods to lure client devices. The techniques include an *Evil Twin*, where the tool creates a clone of an existing legitimate network and *Known Beacons* where Wifiphisher imitates common networks which are not necessarily nearby using a dictionary of ESSIDs. In another method called *KARMA*, Wifiphisher creates networks which surrounding devices search for. After successfull association, the tool allows users to perform various man-in-the-middle attacks[22].

There is a variety of simple scripts using existing Linux tools bundled together to create fake access points. For instance, **Fake-AP**[12] is a bash

[8]https://www.espruino.com/
[9]https://www.arduino.cc/
[10]https://shop.hak5.org/products/wifi-pineapple
[11][https://github.com/wifiphisher/wifiphisher]
[12]https://github.com/thelinuxchoice/fakeap

script which uses hostapd[13] and dnsmasq[14] to create a rogue access point[23].

ESP8266 Deauther[15] is a project for testing 802.11 wireless networks using the ESP8266 microcontroller. It is able to scan nearby WiFi devices, send deauthentication frames as well as fake beacon frames to confuse other WiFi scanners. The author states that he hopes to raise awareness of the deauthentication vulnerability of the current 802.11 standard by releasing this project for inexpensive devices so that everyone with at least 10 USD can recreate the attack[24].

The **esp_wifi_repeater**[16] is an open-source project based on the ESP-Open-SDK. It implements a complex WiFi NAT router on the ESP8266 and ESP8285 devices. It provides functionalities to extend network range, filter and sniff packets and more[25].

[13]http://w1.fi/hostapd/

[14]http://thekelleys.org.uk/dnsmasq/doc.html

[15]https://github.com/SpacehuhnTech/esp8266_deauther

[16]https://github.com/martin-ger/esp_wifi_repeater

CHAPTER 3

Prototype design

3.1 Requirements

The prototype features an option of an automated target selection mechanism. Using a detailed scan of nearby WiFi networks it identifies viable targets based on several criteria including SSID, signal strength and encryption.

After identifying the target and successful discovery of credentials, the prototype connects to the target network in a client(station) mode. Once connected, it can disguise itself as an access point for the target network and provide connectivity for the client devices.

3.2 Architecture

Taking into account that the main advantage of the ESP8266 microcontroller is its low price, the prototype is designed in a way which uses multiple cooperating devices. The system has a centralized client/server architecture as seen in Figure 3.1.

The server is responsible for analyzing the surrounding network, capturing relevant data, dividing workload into chunks and assigning them to workers. The clients (workers) actively request work from the server. Each of the workers can have a different set of actions they are capable to preform.

3.3 Method

The prototype device first scans nearby networks. Several risk factors making the network an easier target were identified from my experience with WiFi router passwords used in my surroundings. Potential targets are ranked based on occurring risk factors.

A **default router SSID** implies that the owner did not put much effort into the configuration. The SSIDs are considered a risk factor since they are

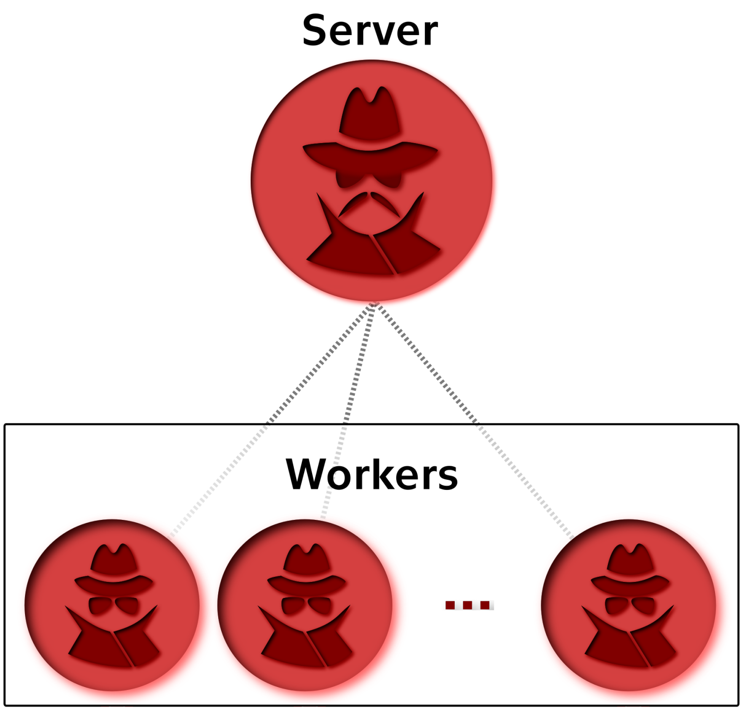


Figure 3.1: Architecture of the prototype

often associated with weak passwords. To target the routers with default SSIDs, a dictionary attack is performed.

Furthermore, older UPC routers are also vulnerable to another attack. A few years ago, the algorithm for deriving the default WPA2 password from their serial number was reverse engineered. To make matters worse, the link between the serial number and default SSID was discovered as well[26]. Since the default passwords seem random, a lot of home users consider them a secure option to use. From my observation, in some households, they are still in use to this day. Specifically, this issue concerns the Ubee EVW3226 and Technicolor TC7200 routers. Their default SSIDs are formed by the string UPC followed seven digits[27]. A simple C program `upc_keys.c`[17] which generates the list of possible passwords is available on GitHub.

Mobile hotspots are often used to share internet access with friends' smartphones. Their SSIDs are also often the default ones, on Android it is typically the name of the device and iPhones have "`<name>`'s iPhone". The passwords need to be typed on a mobile keyboard and consequently tend to be very simple, mostly using eight digits. A specialized dictionary attack can be performed.

Networks with **no wireless security** are relatively easy to imitate since no password retrieval is needed. Although some of them may be guarded by

[17]https://github.com/spaze/upc_keys-lambda

a **captive portal**. A captive portal is a landing page of some WiFi networks. It is often a login page or simply a page with terms and conditions of using the public network.[28] While in the other case, for human users it is still relatively simple to connect to such network and access the internet, it is not as easy for a rogue access point to simulate that browser click. Instead, to bypass this obstacle. The rogue access point needs to change its MAC address to an address of a device which has already clicked that button.

Lastly, **signal strength** is an important criteria. When an access point has weak signal at some location, it makes it easier for the rogue access point to lure users. On the other hand, weak signal might also make it more difficult to connect to the network.

After identifying the potential targets, the prototype chooses appropriate methods to attempt gaining credentials. The methods are simply sequences of jobs which are added to the job queue.

Chapter 4

Realisation

4.1 Hardware

The prototype was realised using four cooperating ESP8266 devices as displayed in Figure 4.2. One device is in the role of a server and the other three are wirelessly connected to it and serve as the workers.

The selected microcontroller has integrated flash storage (SPIFFS). It is primarily used for storing programs but it is possible to upload custom files to it. However, it is only 4 MB in size with the program taking up around 500 kB. To create a self-sufficient prototype, we need to store password lists and other spacious data on the device. Conveniently, the selected D1 mini board is easily extensible by PCB shields. Figure 4.1 shows the MicroSD shield and the schema of its connection to the D1 mini board.

To create the prototype, one SD card shield was soldered to each of the microcontrollers' boards. The server device was given a 64 GB MicroSD. Each of the workers have been equipped with a 16 GB MicroSD card for storing log files, password lists and other data.

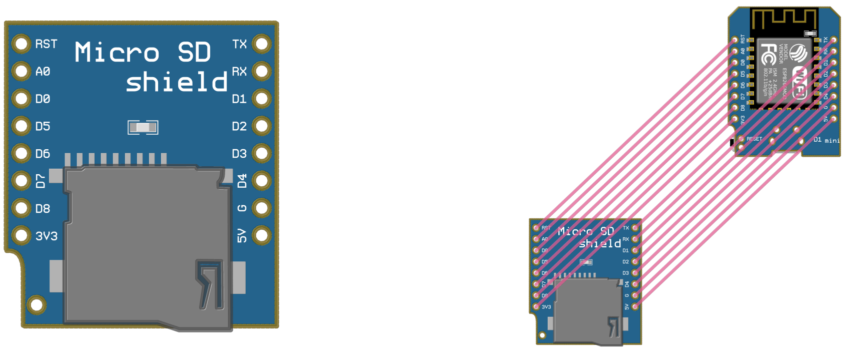


Figure 4.1: MicroSD card shield for ESP8266

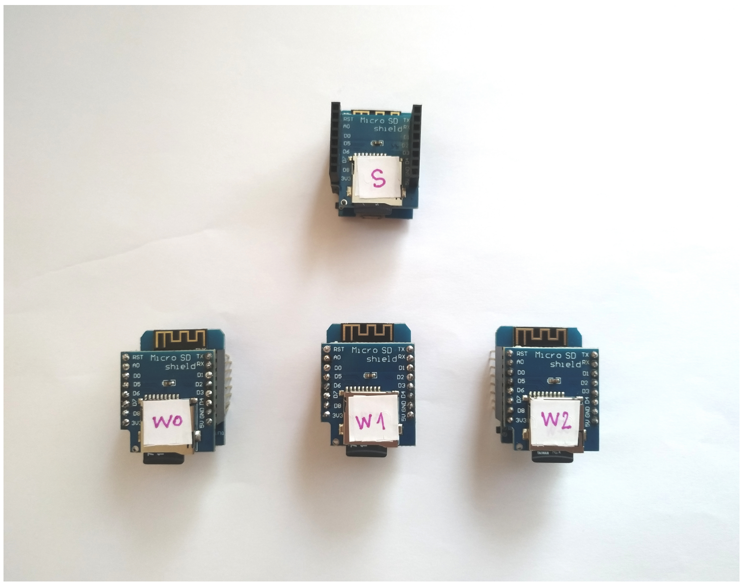


Figure 4.2: Prototype device for rogue access point — the server is the device on top with label *S*, the workers are labeled *W0*, *W1* an *W2*

4.2 Software

The software implementation comprises two different projects: the server and the worker. Both projects use the Arduino firmware for ESP8266 and are written in C++. The communication between server and workers is realised using REST API over Wi-Fi connection. The server creates a soft access point to which the workers connect.

Both projects provide a Logger class which outputs to the Serial interface and an arbitrary number of selected files. The logger supports 5 levels: Trace, Debug, Info, Warn and Error. The log records contain the log level, log message and a timestamp given in milliseconds from booting up the microcontroller. It is possible to choose the minimum log level for each of the individual log files. The log files are stored to the SD card. If the files are kept open for writing once the device turns off, their contents are not written. Opening and closing the files are relatively expensive operations (opening a file, writing a line and closing it right after takes about forty times more time than just writing the line to the open file). To ensure that the log files contain as much information as possible while keeping logging relatively fast, the log files are divided into multiple separate files. Every few lines, the current file is closed and a new one is open.

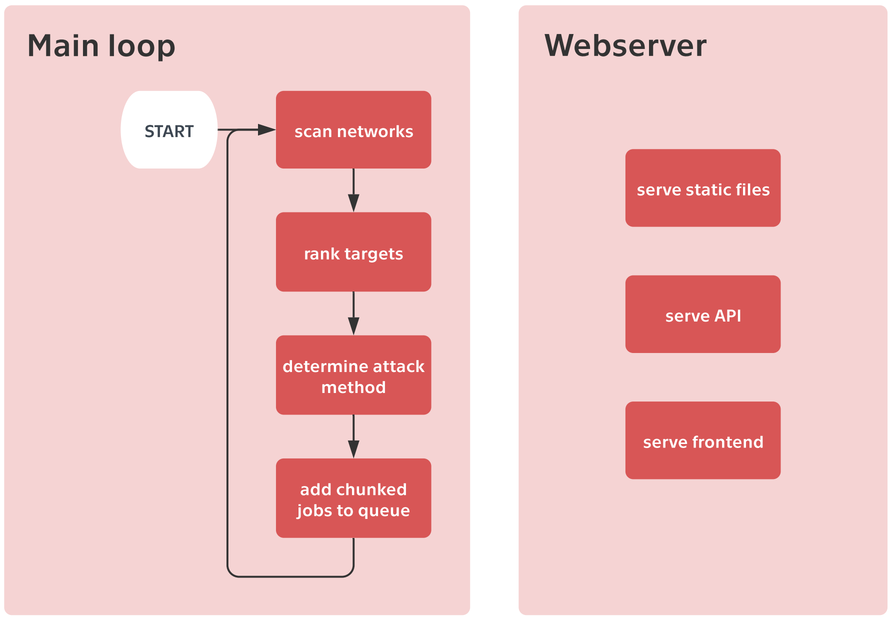


Figure 4.3: Server loop and webserver

4.2.1 Server

The server has several responsibilities: it hosts a webserver and an API, handles job assignment and performs detailed surrounding network scans.

The webserver is a simple asynchronous webserver based on the ESPAsyncWebServer library. It serves static files from the SD card and derives their mimetypes from their extension. The prototype can be monitored and managed by a frontend web application hosted on the device. For demonstration purposes, a simple single page application was created in Rust using the Yew framework. However, the design allows building custom frontends with any technology able to consume the REST API. The REST interface is another responsibility of the webserver. The webserver is configured to listen for the API requests on the `/api/` route. The API is discussed further in Section 4.2.3.

Thanks to the fact that the webserver is asynchronous, it does not block the main loop. In the meantime between answering to workers' requests, the server also manages to perform network scans and identify targets. After target identification, methods for connecting to the available networks are selected and the server divides the needed workload into smaller individual parts. These chunks of work are added to the job queue and serve as a pool of available assignments for the workers. Figure 4.3 depicts the actions performed by the server.

Even though the image might suggest that the main loop and the webserver

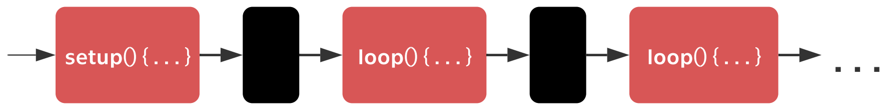


Figure 4.4: Program execution — the `setup()` and `loop()` functions are user defined while the black rectangles signify the asynchronous functions and WiFi management

are running in parallel and that adding jobs to the queue in the main loop and removing them by the webserver would cause race conditions, the selected microcontroller does not support threading. Instead, there is a loop function which is run over and over again. The loop function is not interrupted and all asynchronous functions are run only in between calling the loop function as illustrated by Figure 4.4. These functions include the asynchronous webserver request handling as well as connection of devices to the access point. When the loop function is blocking the processor, the access point is not responsive. It is therefore desirable not to prolong the loop function as it would otherwise block the asynchronous routines. Thus the main loop is implemented as a simple finite state machine.

4.2.2 Worker

The worker's only purpose is to try to complete any jobs assigned by the server. Therefore, the worker continuously requests new work as depicted in Figure 4.5. Each worker can have a different set of capabilities. The worker's capabilities determine what kinds of jobs can the server assign to that specific worker.

Firstly, the worker can have the ability to execute simple dictionary attacks. Each worker has a set of wordlists saved on the SD card. The wordlists are divided into a relatively large number of files to facilitate the workload division. The dictionary attack is implemented as simply trying to connect to the specified network with each password in the wordlist. Since some routers were observed to automatically filter devices which have connected earlier with a wrong password, the workers keep changing their MAC address.

Disguising as another access point is another capability of the workers. Granted they are given the right credentials, the workers are able to connect to the specified access point, mimic it and provide connectivity through the original access point. There is an ongoing effort[18] to merge a netdump

[18]https://github.com/esp8266/Arduino/pull/6518

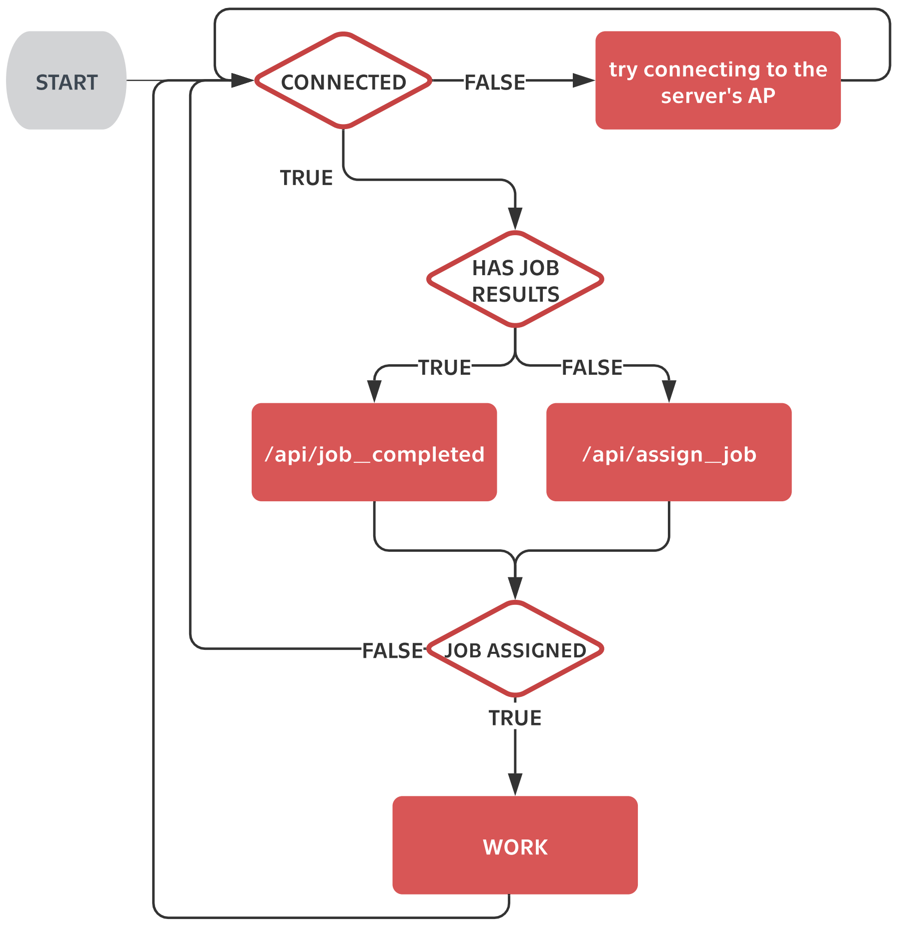


Figure 4.5: Worker loop

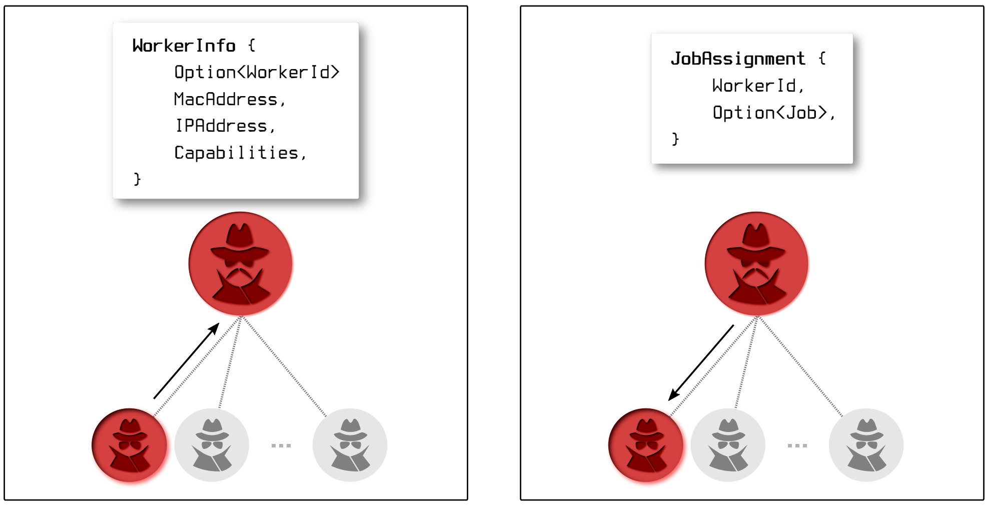
library into the Arduino ESP8266 core. Once that is done, it is going to be possible to easily monitor the packets in a tcdump[19] style.

4.2.3 API

This section contains description of the REST API endpoints both the ones intended for the workers' use and those for frontend use.

The API intended for workers contains two endpoints, both of which only handle the HTTP POST method. `/api/assign_job` expects the worker to send information about himself including an identifier if he has received one (see Figure 4.6). The server responds with a new identifier to use in future com-

[19]https://www.tcpdump.org/manpages/tcpdump.1.html



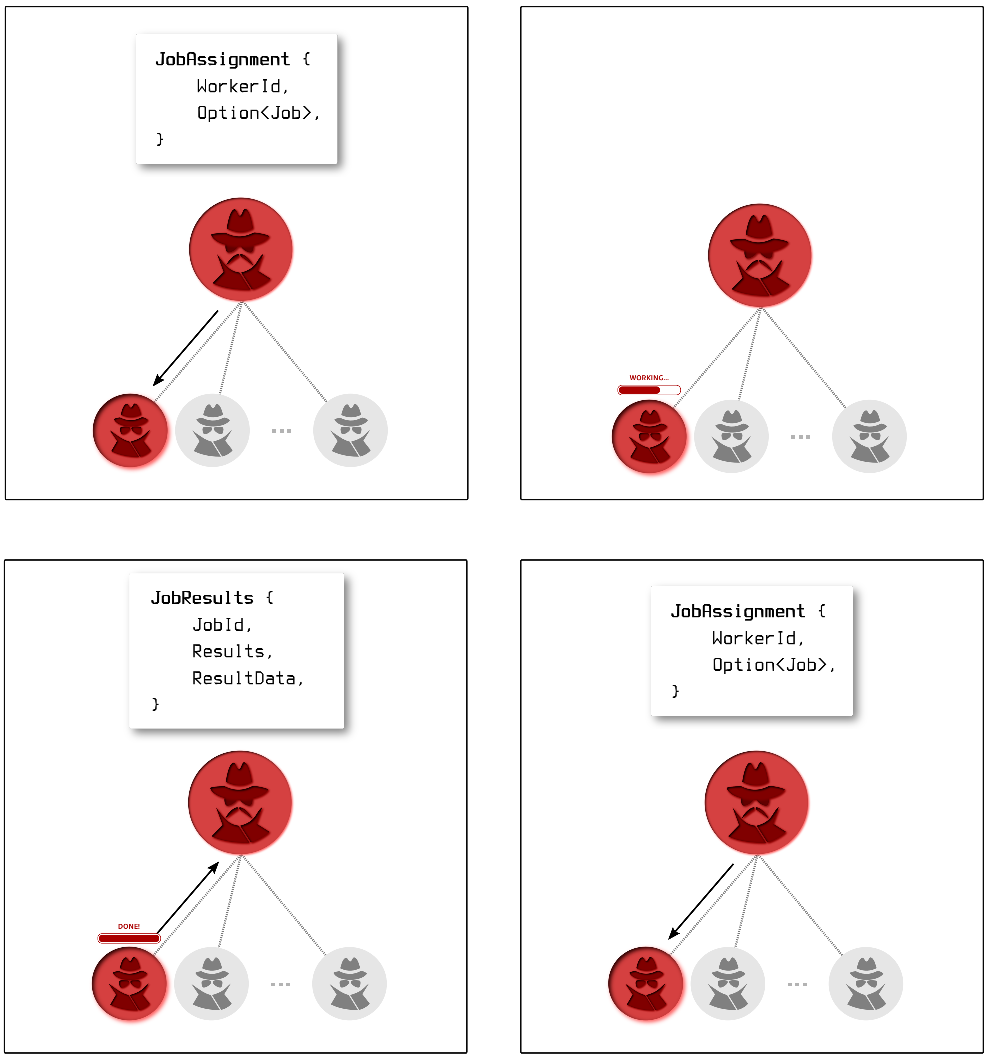
```
WorkerInfo {
    Option<WorkerId>
    MacAddress,
    IPAddress,
    Capabilities,
}
```

```
JobAssignment {
    WorkerId,
    Option<Job>,
}
```

Figure 4.6: Worker requesting job from the server

munication and a job assignment if there are any jobs available. The identifier is sent every time to allow the server to dynamically manage the identifiers and not run out of them after running for prolonged periods of time. When a worker finishes his work, he can report the results to `/api/job_results` as seen in Figure 4.7.

There are four more endpoints for management of the prototype. They can be used to create a simple frontend management system. Three of them respond to the HTTP GET requests. The endpoint `/api/networks` triggers a background network scan and returns a list of networks which were found in the previous scan. `/api/workers` returns a list of registered workers and their details. The list of available, assigned and recently completed jobs is available through the /api/workers `/api/jobs` endpoint. The last endpoint is `/api/add_job`. It responds to a HTTP POST request and serves for creating new jobs for the workers. After adding the new job, the server responds with an updated list of jobs.



```
JobAssignment {
    WorkerId,
    Option<Job>,
}
```

```
JobResults {
    JobId,
    Results,
    ResultData,
}
```

```
JobAssignment {
    WorkerId,
    Option<Job>,
}
```

Figure 4.7: Worker job completion

Chapter 5

Testing

This section describes performed tests to verify the designed functionalities. Though it was intended to do a series of real-world tests, the current pandemic situation limited these plans. Despite this, a simple real-world test scenario is presented along with a performance test.

5.1 Real-world test

A small scale real-world test of mimicking another access point have been performed. At the beginning of the test several devices were connected to the legitimate access point in this scenario. After setting up a rogue access point with the same SSID, PSK and BSSID as the legitimate access point, **four nearby devices connected to it in less than a minute**. The legitimate access point in question was a Turris 1.0 router approximately 5 meters away from the rogue AP prototype. The rogue access point was able to provide the connected devices with internet connection from the legitimate AP. Figure 5.1 shows the number of connected devices over the first 100 seconds.

5.2 Performance

The following test compared the performance of the system based on the number of active workers. Although the prototype has more functionalities, in this experiment it was set up to only perform dictionary attacks. The given wordlists do not contain the correct password to examine the worst case scenario. The prototype was tested with one, two and three workers separately. The tests have shown that the time decrease is indirectly proportionate to the number of active workers. Figure 5.2 depicts the trend of the results. It further visualises the trend by estimating the time required by four workers to perform the task.

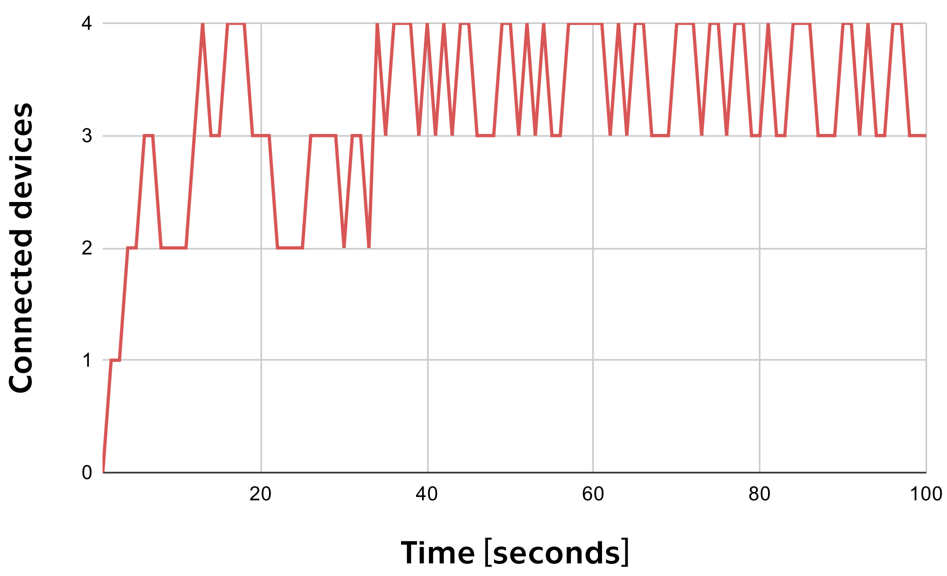


Figure 5.1: Devices connected to the rogue access point over 100 seconds

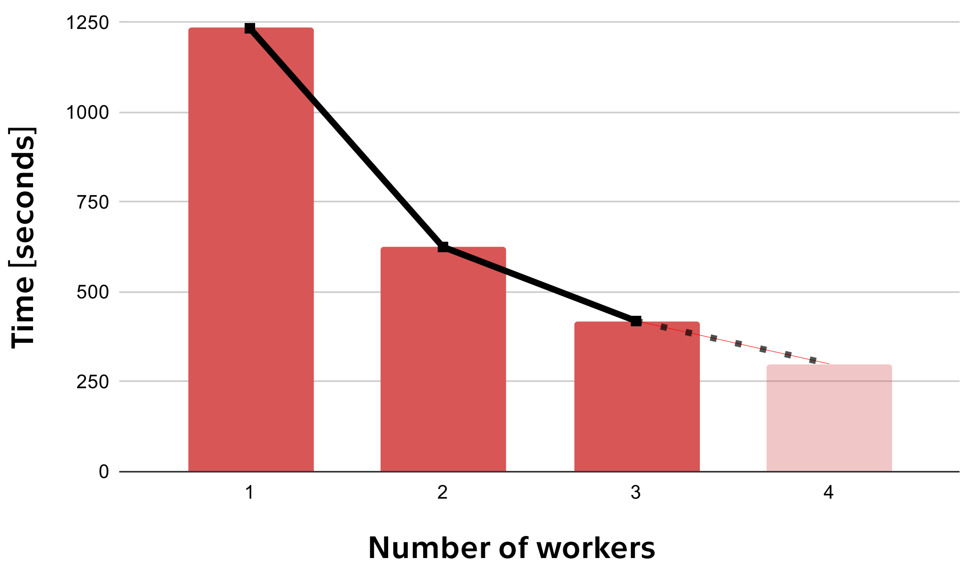


Figure 5.2: Required time to perform dictionary attack based on number of workers, An estimated time required to perform the dictionary attack by four workers is also shown. The time is calculated by taking into account the increasing overhead, i.e, by linearly extrapolating the overhead.

Chapter 6

Conclusion

The goal of the thesis was to compare different versions of the ESP8266 microcontroller and create a prototype rogue access point device. The ESP8266 ESP-12F D1 mini board was selected as the basis for the prototype due to the simplicity of programming and running compared to the ESP-01. Another factor was the availability of specialized shields for the D1 mini board. However, with a bit more soldering effort, the prototype could make use of ESP-01 as well.

The prototype was designed to work with multiple cooperating devices using a centralized architecture. More devices not only bring more computation power but also add the option of making more things at once. Parallelization in terms of CPU cores is quite frequent but this prototype also allows using multiple network devices simultaneously.

The developed prototype is able to automatically select a target network based on a set of simple rules and try appropriate methods for association with it. Furthermore, given the right credentials, the prototype can mimic another access point and provide the connected devices with internet connectivity from the legitimate access point.

A small scale real world test was performed. The system showed robust performance while successfully disguising as another access port. During the test 4 devices connected through the access point.

The prototype currently supports only 2.4 GHz WiFi networks due to the limitations of the selected microcontroller. Considering that 5 GHz networks are becoming more frequent, the prototype would benefit from adding a device supporting the newer 802.11 standards. In the future, the prototype can be extended by adding other workers' capabilities as well as implementing the workers on other platforms.

References

1. HIERTZ, G. R.; DENTENEER, D.; STIBOR, L.; ZANG, Y.; COSTA, X. P.; WALKE, B. The IEEE 802.11 universe. *IEEE Communications Magazine*. 2010, vol. 48, no. 1, pp. 62–70.
2. ALLIANCE, Wi-Fi. *Wi-Fi Generations* [online] [visited on 2020-06-04]. Available from: `https://www.wi-fi.org/discover-wi-fi`.
3. *Media Access Control* [online] [visited on 2020-06-04]. Available from: `https://www.sciencedirect.com/topics/computer-science/media-access-control`.
4. LIU, Chibiao; YU, James. Rogue Access Point Based DoS Attacks against 802.11 WLANs. In: *Proceedings of the 2008 Fourth Advanced International Conference on Telecommunications*. USA: IEEE Computer Society, 2008, pp. 271–276. AICT '08. ISBN 9780769531625. Available from DOI: `10.1109/AICT.2008.54`.
5. MA, Liran; TEYMORIAN, Amin Y.; CHENG, Xiuzhen; SONG, Min. RAP: Protecting Commodity Wi-Fi Networks from Rogue Access Points. In: *The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness & Workshops*. Vancouver, Canada: Association for Computing Machinery, 2007. QSHINE '07. ISBN 9781595937568. Available from DOI: `10.1145/1577222.1577252`.
6. CYBERSECURITY AND INFRASTRUCTURE SECURITY AGENCY CISA. *Security Tip (ST04-015)* [online] [visited on 2020-06-04]. Available from: `https://www.us-cert.gov/ncas/tips/ST04-015`.
7. Rogue Access Points — threat to enterprise security: Bruce Potter. *Network Security*. 2003, vol. 2003, no. 4, pp. 4–5. ISSN 1353-4858. Available from DOI: `https://doi.org/10.1016/S1353-4858(03)00407-0`.

8. PATANGE, Tanmay. *How to defend yourself against MITM or Man-in-the-middle attack* [online] [visited on 2020-05-31]. Available from: https://hackerspace.kinja.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382.
9. PROVOS, Niels. *Safe Browsing — Protecting Web Users for 5 Years and Counting* [online] [visited on 2020-05-31]. Available from: https://security.googleblog.com/2012/06/safe-browsing-protecting-web-users-for.html.
10. BENCHOFF, Brian. *An SDK For The ESP8266 WiFi Chip* [online] [visited on 2020-05-28]. Available from: https://hackaday.com/2014/10/25/an-sdk-for-the-esp8266-wifi-chip/.
11. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *Espressif SDK* [online] [visited on 2020-05-28]. Available from: https://www.espressif.com/en/products/software/esp-sdk/overview.
12. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *ESP8266 RTOS Software Development Kit* [online] [visited on 2020-05-28]. Available from: https://github.com/espressif/ESP8266_NONOS_SDK/blob/master/README.md.
13. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *ESP8266 RTOS Software Development Kit* [online] [visited on 2020-05-28]. Available from: https://github.com/espressif/ESP8266_RTOS_SDK/blob/master/README.md.
14. SOKOLOVSKY, Paul. *esp-open-sdk* [online] [visited on 2020-05-28]. Available from: https://github.com/pfalcon/esp-open-sdk/blob/master/README.md.
15. AUTOMATION, SuperHouse. *esp-open-rtos* [online] [visited on 2020-05-28]. Available from: https://github.com/SuperHouse/esp-open-rtos/blob/master/README.md.
16. TEAM, NodeMCU. *NodeMCU 3.0.0* [online] [visited on 2020-05-28]. Available from: https://github.com/nodemcu/nodemcu-firmware/blob/master/README.md.
17. *Espruino JavaScript on the ESP8266* [online] [visited on 2020-05-28]. Available from: https://www.esp8266.com/wiki/doku.php?id=espruino_javascript.
18. LTD, Pur3. *Espruino WiFi* [online] [visited on 2020-05-28]. Available from: http://www.espruino.com/WiFi.
19. ARDUINO. *Arduino core for ESP8266 WiFi chip* [online] [visited on 2020-05-30]. Available from: https://github.com/esp8266/Arduino.
20. HAK5. *WiFi Pineapple* [online] [visited on 2020-05-03]. Available from: https://shop.hak5.org/products/wifi-pineapple.

21. SUPERTECHGUY. *Pineapple Pi* [online] [visited on 2020-05-03]. Available from: `https://www.supertechguy.com/projects/pineapplepi/`.
22. *Wifiphisher* [software]. 2018. Version v1.4. Available also from: `https://wifiphisher.org/`.
23. THELINUXCHOICE. *Fake-AP* [software]. 2018. Version v1.0. Available also from: `https://github.com/thelinuxchoice/fakeap`.
24. TECHNOLOGIES, Spacehuhn. *ESP8266 Deauther 2.0* [online] [visited on 2020-05-28]. Available from: `https://github.com/SpacehuhnTech/esp8266_deauther`.
25. MARTIN-GER. *esp_wifi_repeater* [online] [visited on 2020-05-31]. Available from: `https://github.com/martin-ger/esp_wifi_repeater`.
26. KLINEC, Dušan; SVÍTOK, Miroslav. *UPC UBEE EVW3226 WPA2 Password Reverse Engineering, rev 3* [online] [visited on 2020-05-27]. Available from: `https://deadcode.me/blog/2016/07/01/UPC-UBEE-EVW3226-WPA2-Reversing.html`.
27. ŠPAČEK, Michal. *UPC Wi-Fi Keys* [online] [visited on 2020-05-27]. Available from: `https://upc.michalspacek.cz/`.
28. *What is a Captive Portal?* [online] [visited on 2020-06-04]. Available from: `https://www.linksys.com/us/r/resource-center/captive-portal/`.

Appendix A

Acronyms

AP	Access point
API	Application Programming Interface
BSSID	Basic Service Set Identifier
DoS	Denial of Service
MCU	Microcontroller Unit
MITM	man-in-the-middle
NIC	Network Interface Controller
PSK	Preshared Key
REST	Representational state transfer
SDK	Software Development Kit
SSID	Service Set Identifier
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access

Appendix B

Contents of enclosed CD

```
  README.md..........................the file with CD contents description
  src..........................................implementation sources
   ├─ ESP Rogue AP Server.........................server implementation
   ├─ ESP Rogue AP Worker.........................worker implementation
   └─ espweb.....................................frontend implementation
       └─ dist........................compiled webassembly and javascript
  thesis.................................the directory with thesis sources
   ├─ thesis.tex.......................the LaTeX source code of the thesis
   ├─ chapters......................the directory with Markdown sources
   └─ images..............................the directory with thesis images
  thesis.pdf..............................the thesis text in PDF format
```