Provisioning VoIP Wireless Networks with Security

ROLAND DUYVENÉ DE WIT

Dissertation submitted in fulfilment of the requirements for the

MASTERS TECHNOLOGIAE:
INFORMATION TECHNOLOGY

in the
School of Information and CommunicationTechnology
of the
Faculty of Engineering, Information and Communication Technology
at the
Central University of Technology, Free State

Supervisor: Professor J. Kinyua

Bloemfontein
December 2008

STATEMENT REGARDING INDEPENDENT WORK

The dissertation is the author's own work and has not been submitted in any form or part thereof to any other University for purposes of a degree. All sources used in this work have been duly acknowledged.

_____

L.R. Duyvené de Wit

ACKNOWLEDGEMENTS

- I would like to thank my study leader, Professor Johnson Kinyua for sharing his immense wisdom and continuous support.

- To the four cardinal points of my emotional compass † Dr. Leopold Jean Duyvené de Wit, Prof. Helene Elma Duyvené de Wit, Jean-Jacques Duyvené de Wit and Iné van Niekerk, thank you for continued love and support.

- Finally, I would like to thank my Saviour for blessing me with strength and the necessary forms of support throughout the duration of this work.

SUMMARY

Telecommunication costs in South Africa are among the highest in the world today [24]. Using VoIP (Voice over Internet Protocol) over wireless networks offers a feasible alternative, but there are numerous security considerations that need to be taken into account when deploying VoIP over wireless networks. This study identifies and addresses some of the issues that arise in the application layer (the VoIP application itself), as well as some concerns with wireless network technologies.

The vulnerabilities are studied in terms of what causes the vulnerability, how it can be exploited and what effect exploitation has. Suggestions to remedy these potential vulnerabilities are also provided where applicable. The vulnerabilities investigated include: buffer overflows, bruteforce attacks, SQL (Structured Query Language) injections, denial of service attacks, encryption on the application layer, and wireless security.

The study has a strong focus on programming principles, various operating systems are used in the study, and networking makes out an important part of the work. Although a working system was developed, the majority of work is divided into several smaller applications that have been developed to aid in researching the mentioned vulnerabilities.

Suggestions towards remedying the vulnerabilities are made where applicable, but it is important to note that vulnerabilities are most often merely oversight errors. Although the vulnerabilities studied in this work identified the most common vulnerabilities, great care should be taken when implementing a secure system. It is crucial to be well aware of what the causes of vulnerability are in order to develop a secure system.

Field work was also done by driving around in a motor car with a notebook, and plotting wireless network nodes along with some security information on a map. Multiple insecure networks have been found which allow for attackers to

further penetrate the systems on the network. Some products do not allow upgrading of encryption technologies, which leave them vulnerable. There are, however, methods to enhance security on the networks without replacing the node; hence suggestions towards securing vulnerable wireless networks are made where applicable.

OPSOMMING

Telekommunikasie kostes in Suid-Afirka is van die hoogste ter wêreld op hede. Om SoIP (Stem oor Internet Protokol) oor koordlose netwerke te gebruik bied ‘n effektiewe oplossing, maar daar is heelwat sekeriteits probleme wat te voorskyn kom wanneer dit so gebruik word. Hierdie studie identifiseer en spreuk sommige van die probleme aan wat te voorskyn kom in die applikasie vlak (die SoIP program self), asook moontelike probleme met die koordlose tegnologie.

Die kwesbaarhede word bestudeer deur te identifiseer wat dit veroorsaak, die impak van uitbuiting en metodes om dit te verlig. Voorstelle om die moontelike kwesbaarhede op te los word gemaak waar dit van toepassing is. Die kwesbaarhede wat bestudeer word sluit in: buffer oorskrywings, brute krag aanvalle, SQL (“Structured Query Language”) inspuitings, ontneming van diens aanvalle, enkodeering in die applikasie vlak en koordlose sekuriteit.

Die studie het ‘n sterk fokus op programmeering metodologie, verskillende beheerstelsels word gebruik, en netwerke maak ‘n belangrike deel van die studie uit. Alhoewel ‘n werkende sisteem ontwikkel is, is die grootste deel van die werk verdeel in kleiner programmetjies om die navorsing van die kwesbaarhede te ondersteun.

Voorstelle om die kwesbaarhede op te los word gemaak, maar dit is belangrik om in gedagte te hou dat kwesbaarhede hul oorsprong vind in oorsig foute. Alhoewel die kwesbaarhede wat in die studie bestudeer word meeste algemene kwesbaarhede identifiseer, moet ontwikkelaars versigtig wees wanneer ‘n veilige sisteem onwikkel word. Dit is belangrik om bewus te wees van die oorsaak van kwesbaarhede alvorens ‘n sisteem ontwikkel word.

Veld werk was ook gedoen deur met ‘n skoot rekenaar te ry en koordlose netwerke te plot op ‘n kaart saam met verdere sekuriteits inligting. Verskeie kwesbare netwerke is opgemerk wat aanvallers sal toelaat om die netwerke

verder te penetreer. Sommige koordlose produkte kan nie opgradeer word nie, wat dit kwesbaar laat. Daar is tog ander maniere om koordlose netwerke verder te beveilig sonder om die eenheid te vervang. Voorstelle sal gemaak word om die probleem om te los.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

VoIP	Voice over Internet Protocol
WEP	Wireless Equivalency Privacy
IPv4	Internet Protocol version 4
NAT	Network Address Translation
PSTN	Public Switched Telephone Network
ADC	Analogue Digital Converter
PCM	Pulse Code Modulation
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
Wi-Fi	Wireless Fidelity
IEEE	Institute of Electrical and Electronics Engineers
MAC	Media Access Control
IP	Internet Protocol
WPA	Wi-Fi Protected Access
NIC	Network Interface Card
RFC	Request For Comments
SIP	Session Initiation Protocol
IETF	Internet Engineering Task Force
DHCP	Dynamic Host Configuration Protocol
CPU	Central Processing Unit
PAN	Personal Area Network
UHF	Ultra High Frequency
VHF	Very High Frequency
SSID	Service Set Identifier
RFMON	Radio Frequency Monitoring
IV	Initiation Vector
SQL	Structured Query Language
DNS	Domain Name Service
FTP	File Transfer Protocol
Suid	Super User ID
IDS	Intrusion Detection System

VAS	Virtual Address Space
LIFO	Last In First Out
SP	Stack Pointer
FP	Frame Pointer
NOP	No Operation
CD	Compact Disk
LAN	Local Area Network
WAN	Wide Area Network
VPN	Virtual Private Network
DoS	Denial of Service
DDoS	Distributed Denial of Service
SSL	Secure Sockets Layer
PSK	Public Switched Key
DBMS	Database Management System
PHP	Personal Home Pages
OSI	Open Systems Interconnect
XOR	Exclusive OR
GPS	Global Positioning System
USB	Universal Serial Bus
AP	Access Point
BSSID	Basic Service Set Identification
ESSID	Extended Service Set ID

1 INTRODUCTION

1.1 Introduction

Many people might think that hacking is a recent phenomenon. The truth is that hacking has a long history dating from the end of the 18th century. Although the term "hacker" was coined later, from its common meaning (gaining unauthorized access to a computer system) it started with the advent of the telephone system. The switchboard workers eavesdropped on calls and misdirected some. The term "hacker" has evolved over time, varying in meaning from someone who knows a great deal about computers to those who use computers to commit crimes. Another meaning of the word is someone who forces a system to behave in a manner that was overlooked or not anticipated by its designer. Today there is a rather large subculture practising hacking, but there are many different perceptions of what hacking is and its origins.

The Hacker's Manifesto [3] was written in 1986 by Lloyd Blankenship, also known as "*The Mentor*", when he was caught for breaking into a system. Here is an extract from the writing:

> This is our world now ... the world of the electron and the switch, the beauty of the baud. We make use of a service already existing without paying for what could be dirt-cheap if it wasn't run by profiteering gluttons, and you call us criminals. We explore... and you call us criminals. We seek after knowledge ... and you call us criminals.
>
> We exist without skin colour, without nationality, without religious bias ... and you call us criminals. You build atomic bombs, you wage wars, you murder, cheat, and lie to us and try to make us believe it's for our own good, yet we're the criminals.
>
> Yes, I am a criminal. My crime is that of curiosity. My crime is that of judging people by what they say and think, not what they look like. My crime is that of outsmarting you, something that you will never forgive me for.
>
> I am a hacker, and this is my manifesto. You may stop this individual, but you can't stop us all ...

When the term “hack” was first coined, it referred to a trick or a way of making something do something else it was not intended to do. Some time ago a cereal named “Captain Crunch” had a whistle inside the box intended as a promotional gift:

Figure 1.1 A Captain Crunch Whistle [1]

John Draper, who later became known as “Cap’n Crunch”, discovered that this whistle can be used to make free phone calls by blowing the correct pitch (2600 Hz) and thus simulating money being put into the payphone. Although this does not involve computers it was referred to as a hack, as something was used in a creative way to make something do something else it was not intended to do.

Because software is written by humans, it is error prone. A software instruction, or piece of code, may seem to do something, and it does, but programmers often overlook the fact that the instruction (under certain conditions) can do something completely different from what the programmer intended.

IPv4 (Internet Protocol version 4) has many great examples of hacks. The Internet has grown beyond the expectations of the original architects of Arpanet resulting in IP (Internet Protocol) addresses depletion and leading to the development of

Network Address Translation (NAT). IP was never intended to be used in such a way, but its architecture allowed it.

From this viewpoint, a hacker is someone who likes technology and has a desire to make things work in ways different from originally intended. In this work, the term *hacker* will be used from this viewpoint. The term *cracker* was coined to refer to people that use their abilities or skills to break the law and often do not really understand what they are doing and use tools written by hackers to break into systems. The term "*script kiddy"* is also used to refer to people who use other people's tools and exploits to accomplish their tasks, often not having much knowledge of what is being done. The meaning of the term *cracker* later evolved into breaking software copy protection so that it can be used free of charge.

Although many hackers prefer not to be categorised, a popular discriminator is according to their intentions. "Black hat" is used to refer to someone with ill intentions such as malice or criminal behaviour. "White hats", on the other hand, are hackers, who immediately upon discovery of vulnerability inform the vendor of the software in an attempt to have the vulnerability fixed. Lastly, grey hats release the exploits they developed, regardless of what it will be used for. Although these terms are commonly used, their meaning is still highly controversial, even in the computer industry.

1.2 VoIP, Wireless VoIP and security

Voice over Internet Protocol (VoIP) has become a feasible alternate solution for telecommunications recently. One of the key advantages of VoIP over the Public Switched Telephone Network (PSTN) is that long distance calls can be made at very low costs. Its power can be clearly seen when it is used to phone someone (who also has Internet access) anywhere in the world free of charge, as opposed to phoning your next door neighbour (although in some countries local calls are also free) at a much higher rate.

The following is a brief description of how VoIP operates. Firstly, VoIP works by having a recording device such as a microphone plugged into a soundcard, which captures the audible analogue signals in the proximity of both, or every user in the case of a conference call. This analogue signal, or sound, is then digitized and stored in a buffer. When the buffer has been filled, it can be manipulated in several ways including encryption and compression. Thereafter the buffer is sent to the other host(s) where it is first deciphered if necessary and then played back to the other user(s) over a playback device such as speakers. Since multiple users can take part in a conversation, and they can speak simultaneously, the system needs to make provision for concurrency – which is typically implemented by threads.

Sound naturally exists as analogue waves. These waves can be represented in a digital form by means of an Analogue Digital Converter (ADC). An ADC uses the electrical voltages generated by the induced current from the microphone to generate a series of numbers. These numbers represent the analogue waves' amplitudes at a specific time. Figure *1.2(a) and figure 1.2(b)* show a sine wave as well as a digital representation of the wave.

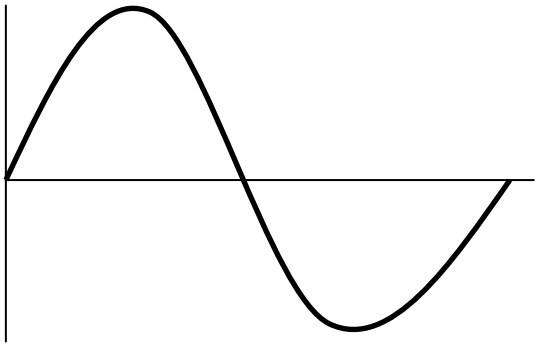

Fig 1.2(a) A Sine wave

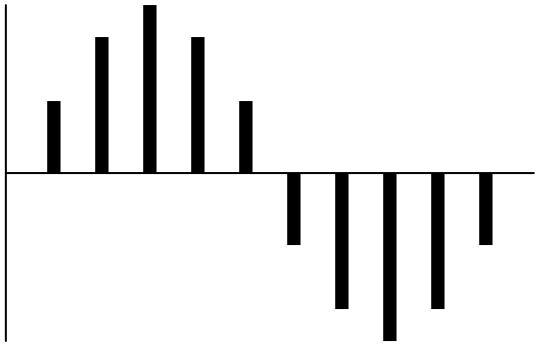

Fig 1.2(b) Digital representation of a sine wave

As the illustration clearly illustrates, the digital representation (*Fig 1.2 (b)*) cannot reproduce the analogue wave exactly. Some of the quality is lost. The digital representation in the above figure allows only seven possible distinct values (the two

extreme peaks, 0, and two values in the middle for positive and negative) on the y axis. A typical 16 bit sample would allow for 65 536 distinct values, which is a much closer representation of the original analogue signal. Another factor influencing the quality of the sound is how often a sample is made over time. In the example above 13 samples where taken. If the period of the wave was one second, the sampling rate would be 13 samples per second, and thus require a bandwidth of 208 bytes per second on monaural and 416 bytes per second for stereo.

Pulse Code Modulation (PCM) is a well-known standard for sampling analogue signals. It samples the signal 8000 times a second and each sample is 8 bits large. This results in a 64 kbps bandwidth requirement. Two particularly widely used techniques are *μ-law* (which is used in North America and Japan) and *A-Law* (used in most other countries).

Various audio compression algorithms have been developed to make transmission of audio over limited bandwidth networks practical. Audio can be compressed in two distinct ways. The first being waveform compression, and the second perceptual encoding. With waveform encoding, the aim is to represent the analogue wave with as few bits as possible. This can be done by reducing the sample size and/or sample rate. The trade-off is of course the quality of the sound versus the size of the encoded data. The other compression technique, perceptual encoding, exploits the manner the auditory system works. The encoded signal, when played back, sounds the same to the human ear, but is in fact something different. It is the same thing to the human ear, as it attempts to only having the elements present that fits the manner in which humans perceive sound, while the rest is omitted. The PCM encoded waveform is passed through an algorithm that filters out elements that are inaudible to the human ear.

For a network to carry VoIP optimally, it requires a great deal of tweaking. Jitter (when the delay or round trip times of successive packets vary) has a devastating effect on VoIP. Streaming applications, including VoIP, rely heavily on a constant

transit time. A constant delay time of 10 milliseconds or 500 milliseconds does not make a large difference in one-way streaming. It only influences the initial delay; the stream continues to flow correctly (in real time) afterwards. A too large delay of VoIP is undesirable, as pauses in communication may become awkward and bring forth interruptions of speakers/users. High jitter, or a large difference in delay, gives an uneven quality to the streamed media and it may become choppy. The following figure illustrates high jitter (*Fig 1.3(a)*) and low jitter (*Fig 1.3(b)*).

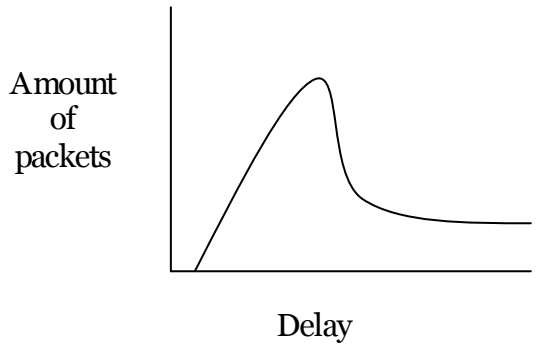

Figure 1.3(a) High Jitter

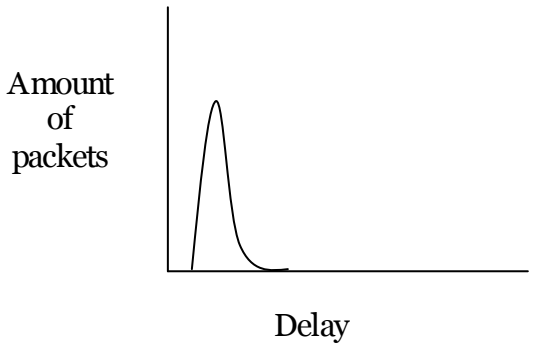

Figure 1.3(b) Low Jitter

TCP (Transmission Control Protocol) can add even further complications on telephony when jitter is high. Since the input is generated in real time (that is through speaking into a microphone), sent across the network, received at the other end, and finally played back in real time (at the same speed recorded), a non-constant delay will increase the delay over time every time the difference in delay fluctuates. The User Datagram Protocol (*UDP)* does not have the same problem. It is an unreliable protocol and some of the packets may be dropped (when received out of order), or may never even be received. On the other hand, when TCP is implemented and the application has been designed not to transmit recorded "periods of silence", every time a period of silence occurs, TCP could be used to implement VoIP, and prevent the delay from continuously growing larger as there are gaps where the delay could be caught up with. VoIP does not require such a large amount of bandwidth (if appropriate compression schemes are used), but other than that, it is a rather expensive and network resource intensive application.

The PSTN has a huge advantage in terms of security because it is much more isolated as opposed to the Internet. Attack vectors are dramatically increased through operating over the Internet, as it is much more accessible as opposed to the PSTN with its wired infrastructure, or proprietary wireless protocols and encryption schemes.

1.3 Statement of the Problem

It is a well known fact that telecommunications costs in South Africa are very high [24]. The discontent of the general public about the charge rate is also visible through articles, blogs, forums and movements such as the open source movement. Since it will be very expensive to lay as many copper lines throughout South Africa as Telkom has done during its approximate fifty years of existence, an alternative means of networking is needed. Wireless Fidelity (Wi-Fi) may offer a feasible solution. Wi-Fi belongs to the 802.11 and 802.11x family of specifications developed by the Institute of Electrical and Electronics Engineers (IEEE) for wireless LAN (Local Area Network) technology that specifies an over-the-air interface between a wireless client and a base station or between two wireless clients.

Wireless technologies used with VoIP exposes several security issues. Many vulnerabilities exist that can compromise the integrity, availability, accuracy, authenticity, confidentiality, utility and possession of information. If any of these characteristics of information are compromised, the information can no longer be trusted. Since information is an important asset, it is crucial to take security seriously.

VoIP can be integrated seamlessly on wireless networks, but this introduces several major security issues. This research will deal with the security issues that arise when using VoIP over a wireless network, but will address wireless network security and VoIP security separately.

1.4 Motivation for the Research

For the modern business the value of information depends entirely on the characteristics (integrity, availability, accuracy, authenticity, confidentiality, utility and possession) it possesses. As with timelines (information loses all value if it is delivered too late), good information cannot be left without safeguarding these defining characteristics. Reliable information needs to be stored, processed and transmitted securely.

Since VoIP and wireless networks are potentially insecure technologies, numerous vulnerabilities exist. Elevation hacking (exploiting full privileged processes to gain elevated privileges) can be used to gain full control of a compromised system, which could in turn leave all information on the compromised system untrustworthy.

Wireless technologies encapsulate much more than the Public Switched Telephone Network (PSTN). PSTN provides enhanced security in the form of physical location trace backs (that is each call can be traced back to the physical location of origin because wires are used). To break into the system, one would have to physically connect to the wired network. This is another barrier that does not exist with wireless networks. Conversely, GSM (cellular) networks operate at frequencies for which tampering devices are not easily unattainable.

1.5 Research Goals, Objectives and Methodology

Security is obviously a crucial factor in protecting information. This study focuses on mitigating the security risks involved in operating VoIP over a wireless network. The main objectives of this research are to provide security for both VoIP and the wireless network segments. The specific objectives of this research are to investigate what causes vulnerabilities, how they can be exploited, what effect exploitation has, as well as mechanisms that could mitigate the risks. We do not

seek to implement a system that is free of vulnerabilities, but rather provide guidelines for designing and implementing a secure system.

As for wireless vulnerabilities we evaluate WEP and WPA-PSK (Wi-Fi Protected Access – Public Shared Key). A dedicated host will be set up to detect and attempt to decrypt public keys. Because wireless access points generally filter only legitimate traffic from passing through a network, a network interface card (NIC) will be used to decrypt network keys. The traffic monitoring driver for Windows, *WinPcap*, did not yet support wireless devices at the start of this work. Hence a wireless *NIC* that is supported by Linux was used to detect and decipher wireless network encryption. A problem with wireless NICs is that expensive cabling is required to attach to an antenna and this cabling and connectors introduce signal loss. We will also do field tests by driving around in a motor vehicle and plotting every detected network node on a map.

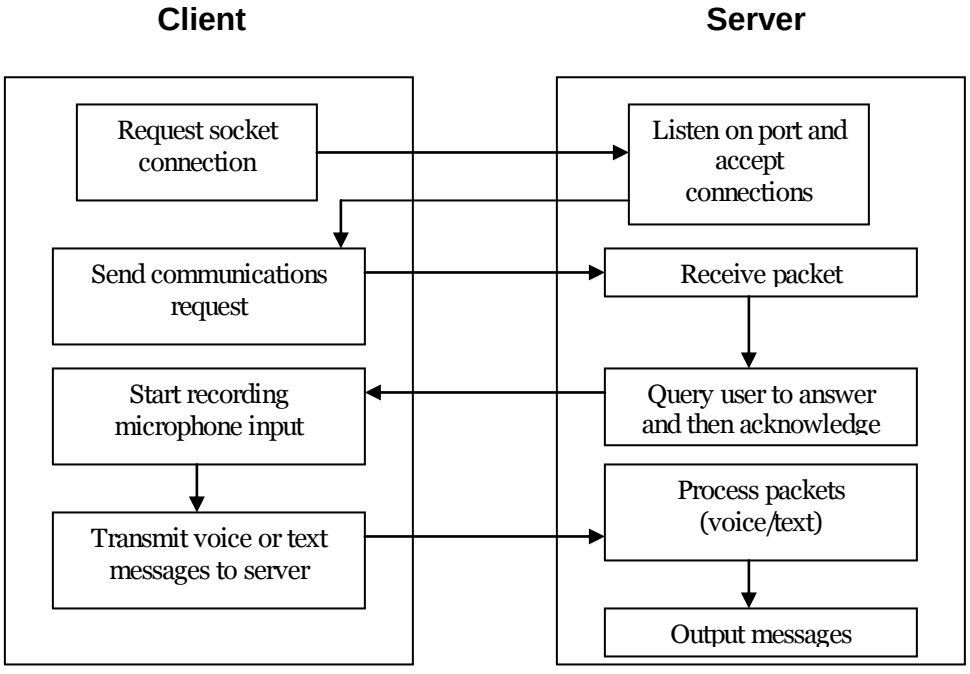

Figure 1.4 A simplified VoIP interaction model

Figure 1.4 shows the simplified version of the methodology with which VoIP works. The client/server architecture will be on each machine using the VoIP software. The

client of one machine will communicate with the server of another, and that client will communicate back to the server of the other. It is therefore necessary to spawn new clients dynamically. This requires a multi-threaded programming environment.

To have practical and usable guidelines and to ensure that VoIP security issues are successfully avoided, mitigated, transferred or accepted, exploit programmes will also be implemented where applicable and used to demonstrate certain issues. They are briefly discussed below:

- Buffer overflows are conditions where the stack is overwritten with user data. We will develop an application and investigate how these conditions arise.

- SQL injections involve special delimiters, which allows user code to be inserted into (or injected into) SQL statements. We will develop a vulnerable application which demonstrates this weakness.

- Denial of service attacks involve a demand of resources which cannot be satisfied. A mechanism of blocking such users will be implemented.

- We will be investigating encryption on two levels. The first is the encryption used to prevent unauthorized access to wireless networks, and the second on a higher level. An application that encrypts and decrypts information sent over the network will be developed to demonstrate the encrypted messages.

1.6 Organisation of the Dissertation

The organisation of the rest of this dissertation is as follows. In Chapter 2, the relevant literature will be reviewed. The anatomy of a network attack is presented in Chapter 3. Security concerns for operating VoIP over wireless networks are discussed in Chapter 4. The implementations and results are presented in Chapter 5. Conclusions and suggestions of future work are discussed in chapter 6. Chapter 7 contains the references and Chapter 8 the appendices.

2 RELATED WORK

2.1 Introduction

The various well-known security vulnerabilities that arise when using VoIP over computer networks are discussed briefly below [17]:

- *Brute Force / Dictionary attacks (on Usernames and Passwords)*

  Brute force attacks involve the application of computer and/or network resources to attempt to try every possible combination of options (characters) of a password. If the field of possible accounts can be narrowed down, the success ratio increases significantly. Dictionary attacks can be seen as a specialized form of a brute force attack. A list of commonly used passwords (known as the dictionary) is used to guess with, instead of every possible combination.

- *Buffer Overflows*

  Buffer overflow exploits have enjoyed popularity recently. These vulnerabilities are created by an application error that pushes instructions on the processing stack when a larger buffer than compensated for is copied onto the buffer. If the overflow section of the buffer is specially crafted, remote code execution is possible, but typically results in denial of service.

- *Unauthorized access – limited by IP address*

  With this type of attack an attacker sends information to another host with an alternate IP address indicating that the message is coming from a trusted host. Usually IP spoofing is used to gain unauthorized access to information or other resources.

- *Packet sniffing*

  A sniffer is a device capable of monitoring information travelling across a network. They are virtually impossible to detect and can theoretically be inserted

anywhere. The only ways to bypass them is to use another protocol or to use encryption with the communications.

- *Timing attacks*

  Timing attacks include a number of different techniques. They include malicious third party cookies, interception of cryptographic elements to determine encryption algorithms and keys as well as monitoring response times on authorizing messages to determine valid usernames.

- *Denial-of-Service attacks*

  A large number of requests are sent to a target which renders the target unable to handle these requests, as well as other legitimate ones. Distributed denial of service attacks involve a number of different systems used together to attack the target(s).

A number of mechanisms have been developed to mitigate networks (including wireless networks) against these vulnerabilities. Most of these are summarised below.

- *Encryption*

  In order to prevent eavesdropping on traffic, encryption plays an important role. Several different encryption techniques and algorithms exist that can be used. With wireless networks, the standard is Wired Equivalency Privacy (WEP), which will be considered in detail in this research. An encryption scheme will also be implemented through a text messaging application.

- *IP Address filtering*

  A security technique known as IP Address filtering can be used to limit network access through gateways by allowing only certain IP addresses access. This is a viable, but not an entirely secure mechanism, as legitimate IP addresses can become available as soon as a network node shuts down. An attacker can then

adopt a legitimate IP address. A simple application that accepts connections from a list of IP addresses will be implemented to demonstrate its efficiency.

- *Media Access Control (MAC) address filtering*

  As with IP address filtering, this is another sensible, but flawed mechanism. MAC addresses are designed to uniquely identify network hardware and therefore create a barrier which disallows unauthorized nodes to access the network. However, with the right tools, a MAC address can be changed on a network device. This means that if an authorized MAC address is known, access can be gained. The mechanism is exactly the same for IP address filtering, but since it cannot be changed in some instances (such as some access points) it serves as a superior means than IP address filtering.

- *Wired Equivalency Privacy (WEP) Key strength*

  WEP uses a key to encrypt and decrypt messages travelling over the network. This key is needed by every networked node to be allowed on the network. The complexity of this key determines the security of the encryption from the point of view of brute force attacks. Obviously (although not always the case), the larger the key, the longer it takes to decipher.

2.2 Wireless Network Security

In [4] wireless security is addressed from the perspective that limited computational power, limited bandwidth and a noisy channel are taken into account and important trade-off decisions are based on that. The proposed security protocol (Point Coordination Function) with its algorithm supports these theories. Limited bandwidth forces a small number of messages to be exchanged to provide security for this model. Limited computational power enforces sophisticated cryptographic methods. The noisy environment requires provision for suitable retransmissions, which is incorporated.

As with all network protocols, a need for standardisation is undeniable in order to support the important issue of compatibility. H.323 [16] was developed in 1996 and revised in 1998 as a recommendation for VoIP, but was much more of an architectural overview than a detailed protocol description. Session Initiation Protocol (SIP), described in Request For Comments (RFC) 3261, was designed by the Internet Engineering Task Force (IETF) to replace H.323. Unlike H.323, SIP only handles the setup of a call through a set of primitives, while H.323 is a complete protocol stack.

The work in [17] discusses VoIP security when deployed in unsafe environments. When deployed in unsafe environments, the possible attack vectors are numerous. Vulnerabilities can be found in the underlying network, the transport protocols, the VoIP devices (for example, servers and gateways), the VoIP application, other related applications (for example, Dynamic Host Configuration Protocol, DHCP), the underlying operating systems and more. The PSTN's security advantage (physical character and operation) is also pointed out. Another concern is that voice conversation flows over many different physical networks with different interception characteristics.

Elementary security principles are discussed within [4]. A variety of attack vectors are explained and focus on security in general is applied.

Many concerns for wireless hotspots are discussed in [18]. Although this work is not aimed at hotspots in particular, an overlap exists with some of the security concerns hotspots face. WEP, although it has its place, is a flawed security measure and should not be fully relied upon.

The work described in [19] is mostly concerned with the security issues of mobility. The considerations encapsulating seamless roaming along and re-establishment of connections are highlighted. Re-establishment problems of Virtual Private Network (VPNs) connections are also discussed.

Re-authentication is also discussed in [4] and two protocols are proposed that provide a clear improvement over Public Key Decryption (PKD). Real time applications are able to function with IEEE 802.11 networks.

An in depth study is made on operating systems in [20]. The system architecture is introduced and various CPU registers along with their functions and usage are explained. As Operating Systems dramatically influence information security, it is an important basis of the study. Knowledge of assembly language is necessary in order to resolve some of the targeted security threats.

In [9] the workings of buffer overflow exploits are explained. This article describes how virtual address space is used, which can be used to achieve remote code execution by writing a larger amount of data into an array than compensated for. This overflow allows an attacker to manipulate execution flow and can bring forth huge bugs.

Novell's SUSE Linux 10 has been chosen as an Operating System on which some sections of this study will be done.

A Linux tool for configuring wireless network interfaces (*iwconfig*) is described in [12]. This utility is helpful in putting a wireless device in the desired mode of operation. Possible modes include **Ad-Hoc** (network composed of only one cell and no Access Point), **Managed** (node connects to a network of many Access Points, with roaming), **Monitor** (the node is not associated with any cell and passively monitors packets on a channel/frequency), **Master** (the node is the synchronization master or acts as an Access Point), **Repeater** (the node forwards packets between other wireless nodes), **Secondary** (the node acts as a backup master/repeater) or **Auto**. In order to decipher WEP keys, it is necessary to have the wireless network interface in monitor mode. This enables the adapter to receive packets from all

networks, not only from the cells it is connected to. When connecting to a network, it generally has to be in Ad-Hoc, Managed or Master mode.

Airsnort is a WEP key cracking tool which exploits the RC4 scheduling weakness as discussed by Fluhrer. In [21] usage and installation of the tool is described. Airsnorts' mechanism of cracking WEP will be evaluated against others with statistical information.

In [2], VNC, a remote desktop administration utility, usage and installation procedures are documented. Because the Linux machine was running on the roof, it was difficult to administer it from there. Instead, the freeware edition of VNC was used to do so over the network. The merit of this application is that it runs cross platform via a web interface and thus easy to use. The remote host can be controlled via the network with a mouse and keyboard as if the user was sitting in front of the remote machine.

Kismet is another WEP key cracking tool with extended functionality. The manual pages in [22] describe its usage.

2.3 Wireless Network Principles

Wireless networks offer an effective means of networking, especially in rural areas where terrestrial networking infrastructure is unavailable. VoIP can be operated over wireless networks at very low costs, but wireless networks are easily penetrable if proper security measures are not used since they are not as isolated as the PSTN. This weakness arises because of the way wireless networks operate – over the air. Wireless signals are difficult to control and often cover a larger range than is often necessary, making them easily accessible to unauthorized use, because there is less restriction on physical location.

Wireless technologies make use of electromagnetically induced waves. An important differentiating characteristic is the frequency of the wave. Different frequencies attribute different properties to the waves, such as absorption. The following table shows the widely used names of the electromagnetic radio waves according to their frequencies. A description of the properties and possible uses of all the other types of waves can be found in [5].

Electromagnetic Radio Wave	**Frequency**
Extremely high frequency (Microwaves)	30 GHz – 300 GHz
Super high frequency (Microwaves)	3 GHz – 30 GHz
Ultra high frequency	300 MHz – 3 GHz
Very high frequency	30 MHz – 300 MHz
High frequency	3 MHz – 30 MHz
Medium frequency	300 kHz – 3 MHz
Low frequency	30 kHz – 300 kHz
Very low frequency	300 Hz - 3 kHz
Voice frequency	30 Hz – 300 Hz
Extremely low frequency	3 Hz – 30 Hz

Table 2.1 Electromagnetic waves

Many implementations of wireless networks exist and examples include: Bluetooth, Wi-Fi and Infrared. Bluetooth and Infrared networks are typically arranged in Personal Area Networks (PANs), which cover very limited topographical areas. The 802.11 standard (Developed by the Institute of Electrical and Electronics Engineers - IEEE) has three distinguishable subsets: a, b and g. Both the 802.11b and the 802.11g standards operate at the 2.4 GHz frequency band, while the 802.11a standard operates at the 5 GHz frequency. The 802.11b allows up to 11Mbps, while the 802.11g standard can handle up to 108 Mbps mainly owing to the differences in their physical layer's protocol designs.

A noteworthy characteristic of electromagnetic waves is that the higher the frequency becomes, the higher the level of absorption, and the greater the need for line of sight between the antennae. Microwave frequencies, where Wi-Fi operates, are higher frequencies and require a clear line of sight if the network spans over considerable distances. Since 802.11a operates at 5 GHz, which is in the Super High Frequency (SHF) band, as opposed to the 2.4GHz 802.11b/g at the Ultra High Frequency (UHF) band, while the 802.11a is much more dependent on a clear line of sight.

Another important difference between the 802.11a and 802.11b/g standards is their channel allocations. The 802.11a standard has 16 non-overlapping channels ranging from 5170 MHz to 5805 MHz, while the 802.11b/g standards have 14 channels with an overlap to adjacent channels ranging from 2412 MHz to 2482 MHz. Since some countries have different regulations on which parts of the frequency spectrum may be used as well as different regulations on their gain, the manufacturers of the equipment tend to manufacture for the most widely used specifications. In most countries the 2.4 GHz band is unregulated and can be used without a license. Unfortunately, South Africa is an exception. The intentional broadcasting of signals over public areas, such as roads, is illegal. Of course, signals are hard to control and isolate from spanning across a road. For example, the South African law prohibits two devices from communicating when the signal spans across multiple properties or public property.

2.4 Wireless Network Security Schemes

The well known proverb that a chain is only as strong as its weakest link is especially valid for networks with integrated wireless components. Probably the largest problem is that whenever a wireless device becomes part of (or associates with) a network, it leaks out data over the air, which can easily be intercepted. If a highly confidential VoIP conversation, using only Pulse Code Modulation (PCM) was to be operated over a wireless network only the wireless encryption would need to

be broken for the whole conversation to be played back in its entirety. It is therefore much more secure to add more layers of encryption, for example, encrypting the digitised sound at the application layer as well as the wireless network's communications layer. The digitized voice could for example be encrypted using public key cryptography, as it needs to be deciphered at the receiving end as well. There are currently two widely used encryption schemes for the 802.11x standard, namely Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA).

Since wireless networks operate over the air and it is difficult to restrict their coverage to a single physical location, it is necessary to encrypt all the data that is sent. The first solution was Wired Equivalency Privacy (WEP) and was supported by most of the first generation wireless WEP devices, but this scheme has already been compromised. The problem with WEP is that the encryption key used is static. (A WEP key as well as a Network Name or Service Set Identifier (SSID) is required to connect to a WEP encrypted wireless network). WEP uses a special type of prefix (a three-byte array) that is appended to almost every packet. This prefix is called an Initiation Vector (IV) and if enough packets are captured, it is easy to determine the key through statistical mathematics.

Wireless and wired sniffing are largely the same, interception being easier with wireless networks. Sniffing is referred to as a passive attack, as the attacking host does not (need to) communicate with the remote host. The remote host does not know of the sniffer's intention or presence. A few tools have already been written to crack wireless networks. Some of the most popular ones are: *Airsnort*, *Kismet* and *Aircrack*. Although *Kismet* can crack WEP, it is more often used as a detection and/or site surveying tool. Wireless sniffing is very hardware dependent. The wireless Network Interface Card (NIC) needs to support RFMON (Radio Frequency Monitoring) mode, or monitor mode. It can be a lengthy process to crack WEP as several hundred thousand weak keys or IVs need to be captured before the key can be deciphered. A tool named *Aireplay* has been developed to speed up this process. It captures packets in the air that created weak IVs (typically an

association) and resubmits them again, as if it was legitimately sent by the original host. This results in faster IV collection and ultimately faster cracking of the key. Depending on the type of encryption and some other factors, it can take as little as five minutes to crack a WEP key with this type of attack.

The cracking of WEP keys is not an exact science. Sometimes, using *Aircrack*, with sufficient keys can break the key within one second. In some cases, with additional IVs, it can take even longer. Any estimate is therefore a rough estimate of how long it should take to decipher the key. Below is a table that indicates widely used key lengths for WEP encryption.

WEP Encryption Name	**Hexadecimal**	**Characters Data Byte Length**
64 bit (40 bits)	10	5
128 bit (104 bits)	26	13
176 bit (152 bits)	38	19
256 bit (232 bits)	58	29

Table 2.2 WEP key sizes

As a general rule the larger the encryption key, the more secure the traffic. This is often not the case when other factors come into play such as key strength. Humans often want to use easily remembered keys in order to easily enter them on other hosts requiring the keys. Another possible problem is found when the key is randomly generated by the computer and carried around on a flash stick. The weakness lies within limitations of randomness on the generation algorithm. If such limitations are exposed one can decrease the possible combinations and thus improve chances of successfully brute forcing the key.

History has proven that security should not be supported through obscurity. Secret information tends to leak out. It is therefore best practice, with cryptography, to make the encryption algorithm publicly available, but to protect the private key. Both WEP and WPA-PSK are vulnerable to their keys being discovered. The details on what

makes them vulnerable will be discussed in detail later, but for now it will suffice to know that cracking the key involves capturing packets. Initiation Vectors (IVs) are special packets that are communicated amongst nodes of a wireless network in order to associate with each other as an authentication requisite. These packets are necessary to break the encryption and a large number of them are required. Since wireless networks can operate on several channels, which are at different frequencies, many prewritten cracking utilities offer collection of packets from all the networks across the channels. Many of them also offer locking to a single channel. This increases the collection process for a single network at a single channel, as some weak IV might pass by undetected, while the NIC monitors at other frequencies.

As soon as an attacker has access to the network, (if the WEP key was deciphered, for example) there are several noteworthy attack vectors that will be discussed in the next chapter.

3 THE ANATOMY OF A NETWORK ATTACK

A basic layout of the methodology an attacker may follow, which is commonly referred to as the anatomy of an attack, is required before we move on to the systems design. This is intended as a guide to aid the understanding of attack vectors and should not be used under any circumstances to break into any system without authority. It needs to be emphasized that the process is not unrivalled, but is only a rough outline with some finer details where applicable. There are many additional techniques that can be used for each of the steps, but only a few will be discussed to the discretionary of the author.

3.1 Information gathering

During information gathering as much information as possible is collected. At first the information is gathered without regard as to how it is going to be used or how it should be structured, but as the attacker gains experience, a feel for what kind of information is important is developed. This information includes everything related to the target. If the target was a company, for example, information ranges from employee details, to the domain name of the company, to the external service providers they use. Potentially knowledge about the target could be useful, thus one step can often be iterated after reaching a dead end in another step.

3.2 Network Probing

If the target is going to be attacked over the Internet (or over any network), probing can reveal much of additional useful information. IP scanning, port scanning, WHOIS, DNS lookups, Operating System fingerprinting, retrieving information from corporate web sites, sniffing, and service identification are only some of the most popular methods. A quick discussion of some of these methods, along with a practical example, follows.

There are many free port and IP scanners available on the Internet. The basic function of an IP scanner is to identify additional hosts on the network. Many of these IP scanners follow different techniques to discover hosts. The main function of Port scanners is to identify what network services are running, and not blocked by a firewall or other device or measure, on each of the identified hosts. *Nmap* [6] is probably the most popular IP and port scanner utility available for UNIX-like systems designed specifically to be effective in scanning large networks, but it works fine on single hosts as well. *Nmap*, which is free and open source, can also be used for operating system fingerprinting. It makes an educated guess on the basis of how packets are treated and the reaction of the remote host' to certain types of packets.

The following command will launch a *SYN* port scan on the 192.168.0 subnet and even show what type of services were found, as well as the version of the programme (if it is detectable and known).

```
$ nmap –sS –T4 –F 192.168.0.0/24
```

The parameters' meaning can be found in *nmap*'s documentation (unlike many open source applications *nmap* is well documented). The /24 annotation merely means that all addresses (1 to 254) will be scanned in the 192.168.0 network with a subnet of 255.255.255.0. This annotation is derived from the number of bits. As IP addresses are composed of four blocks of eight bit values, the /24 annotation means that the first 3 blocks of the subnet mask have values of 255 (8 x 3 = 24).

Netcat and *telnet* are two other noteworthy tools. It should be noted that we are discussing the *telnet* client, not the protocol. *Netcat* is well known as the "hacker's Swiss army knife", for its power and diversity of usage. It has been observed that *netcat* is to networks as *cat* is to files, with some extra functionality added. *Telnet* and *netcat* can be used to manually communicate with a service.

The protocol's commands can be typed in (if the protocol is text based) through the utility. Arguably, the most important and powerful feature for attackers, of *netcat* is its ability to spawn a shell for remote control of the host. *Telnet* is actually a largely deprecated service (because it had plaintext authentication) intended for remote control and replaced with the more secure *ssh*, but its client is often included with Operating Systems to aid troubleshooting. The following *netcat* command listens for a connection on port 23 and gives a shell to the connecting host, which means that one can work on a remote machine as if sitting in front of it by *telnetting* to the socket.

```
$ nc –e "/bin/sh" –l –p 23
```

DNS lookups are used to identify more target hosts related to the target. WHOIS [7] is a tool that queries a publicly accessible database of registered domain names. It can reveal information such as the organization's location, its domain name, contact persons along with their contact details, and the subnet of the organization. Although this might sound somewhat useless at first, this can be very helpful to attackers who find themselves at a dead-end with usernames and passwords.

Alternatively, there are some security auditing tools such as *Nessus* and *GFI Languard* that combine this step along with the following step (Vulnerability Assessment). It then creates a report of all vulnerabilities found and suggests solutions on how to fix them. Almost needless to say, the trade-off between manual scanning and these assessment tools are convenience versus thoroughness (if the auditor is competent in doing so).

3.3 Vulnerability Assessment

Once the attacker has sufficient information about the target and probed its network, he can start identifying weaknesses, or possible entry points. This depends on what the attacker wants to accomplish. If, for instance the attacker

wants to deface the web site, it would probably be best to first look for vulnerabilities within the web service. If an administrator login page is available along with an SQL injection vulnerability (SQL injections are explained in chapter 4), the task is straightforward. One can even use search engines for this purpose. When searching for “Admin” “Login” on Google, many hits are found and it is shocking how many of these web pages are not protected against SQL injections. On the other hand, if the attacker is after sensitive data, he might want to try and guess, or use brute force to attack the username and password of an FTP account.

This is where the information obtained during information gathering comes into use. Some employees may use their own names as usernames and some other phrase as their password, such as their date of birth, or their child’s first name. This practice is quite common, even though password strength is repeatedly emphasized in at companies with sensitive data. Creative attackers may even go through the garbage of a company in order to obtain some useful information. If the attacker is not after sensitive data, or defacing the company home page, he might want to obtain complete control over the system, and maybe escalate the attack to other systems on the network as well. This is usually done by looking for additional weaknesses in the services running on the target hosts. After identifying the services, an attacker can further identify the service version, and check whether there are known vulnerabilities of the service. Alternatively, many programmes are open source, which means that the source code is available for inspection by the attacker. The attacker then needs to obtain a copy of the code, read it, find a vulnerable piece of code, and exploit it. After the vulnerabilities are known, the attacker can try to penetrate the vulnerable systems.

3.4 Launching the Attack (Penetration)

It will be more efficient to first check on the Internet whether there are already existing exploits that the attacker may use to accomplish his task. Two of the best databases are the *metasploit* framework and *milworm*. The *metasploit*

framework is a framework designed to aid the research of exploits. Contributors make their exploits available to others in the hope that they will be educational. *Millworm* is another such database, but in contrast to *metasplot*, it is not a framework, and more of a host for source code of exploits written in natural programming or scripting languages.

The main difference between the two is that *metasploit* has its own 'language' and 'modules'. Often, attackers may find that there is no full disclosure or proof of concept available on the Internet for the sought vulnerability. It is often here that the difference between the hacker and the script novice becomes apparent. In order to be dubbed an "elite hacker", one needs to have extraordinary programming skills additional to the preceding knowledge. In Chapter 4, we discuss buffer overflows in detail where it will become clear that quite extensive knowledge is needed in order to successfully exploit this type of vulnerability.

3.5 Privilege Escalation

It has become common over recent years for operating systems to support several user accounts all of which have their own predefined privileges. When a service has been exploited, and typically this involves having remote access to a command prompt, the privileges of the attacker are that of the user account running the service. It has therefore become a common security practice to run services with limited privileges. This does not necessarily prevent the attacker from gaining full privileges, but it creates an additional barrier and is therefore justifiable. Gaining elevated privileges is not limited to what is discussed here, for example, two of the most common methods are to try and recover, or break usernames and passwords, and to exploit another weakness. An example of the latter in UNIX-like systems would be to overflow the buffer of a programme that takes a parameter value with the SUID (Super User ID) flag set. These programmes are common in the UNIX environment. In Windows XP, a possible command would be:

```
C:\> at 13:00  /interactive cmd
```

This spawns an interactive command terminal with *System* privileges and a title of *svchost.exe*, the highest privilege in the Windows environment, even higher than *Administrator*, at the specified time. It is therefore wise to specify the time one minute into the future of the compromised machine. Since the "*at*" command is owned by the *System* user the newly launched scheduled task still belongs to *System*. Again, a simple side-effect was overlooked. If a remote desktop service is available and logged into, this can be used in to run the graphical mode with *System* privileges as well simply by opening the *Task Manager*, ending the *explorer.exe* process, and running *explorer.exe* again within the *svchost.exe* shell.

After successful privilege escalation, the attacker is free to do with the system as he pleases. If the attacker is malicious, this could severely impact the victim, from disclosure of sensitive data, sabotage, to losing all the data on the compromised system.

3.6 Maintaining Access

Good security principles not only suggest, but demand that systems must be kept up to date by applying appropriate patches and updates on a regular basis. If an attacker needs to stay in control of a compromised system, they usually set up a *backdoor* or make use of a *rootkit* to maintain access to the host. These programmes can also be present with phenomenal stealth, so much so that the user or antivirus software may be completely unaware of its presence. It is thus wise to manually check from time to time for the presence of such software. The methodology to do so however is beyond the scope of this research. Useful information can be found in [8] and [23].

3.7 Stealth mode - Covering Tracks – (editing logs)

Since the attacker needs to connect to the compromised system, and probably generates much traffic, especially in the case of a brute force attack, chances are that his activities might have been logged in several different ways. One of these methods might be an intrusion detection system (IDS). Another method might be as close as the command prompt. The Linux bash shell logs all the commands issued. This is useful for many reasons, including pressing the up arrow to traverse through previously executed commands. As usual there are ways around this for the attacker, such as editing his log entries. Linux commonly stores bash's logs in /root/.bash_history as plain text. It is thus easy for the attacker to simply delete the lines of the commands he issued when he is done, leaving it exactly as it was for the user. Specialized log cleaners also exist to automate these tasks. When network traffic is logged, there are also a few other work arounds, such as making use of an anonymous proxy, which hides the IP address of the attacker from the logging systems of the victim.

3.8 Radio Network Intrusion

The problems with wireless security have already been discussed, but it has not yet been properly put into perspective in terms of where it fits in. How to retrieve a WEP or WPA-PSK key has already been discussed. This step could actually have been added to the top, for getting on the corporate network, but it is discussed here as it is not the only method to gain access to the network, and not all corporations employ (vulnerable) networks. When an attacker successfully compromises the wireless network's encryption, there may again be many additional fortification layers. Probably one of the largest benefits for an attacker being able to break into a system from the wireless network is the added anonymity. If the attacker faked the MAC address of his NIC prior to breaking into the system, the logs generated by his actions do not uniquely identify the attacker, or the hardware that was used. In order to identify the culprit, the attacker needs to be either recorded by means of surveillance or physically caught while he is in the act. The attacker may be sitting in a building across the

street, or more than a block away, which almost completely rules out both methods. It is thus extremely difficult to obtain incriminating evidence to an attack from this vector.

Table 3.1 depicts some of the common exploitable software vulnerabilities with a short description of each.

Weakness	Description
Buffer Overflow	Buffer overflows have recently become very popular. Buffer overflows are caused by writing a larger piece of data into an array or buffer than was provided for, leading to memory corruption. The main problem with preventing buffer overflows from happening is that array bounds need to be checked at runtime, which is a performance drawback.
Format Strings	Very similar to SQL injections, this vulnerability is produced through unexpected special characters input by a user. A typical C programme may read in a string to a character array from the user and print it out to standard output. If the user was to type in some special characters, the programme will treat them as the tokens they represent, for example if "%s" was input, another string will be expected as parameter in the *printf* statement.
Race Conditions	A case where the order of instructions or events, and in the majority of cases as a side-effect of improper file locks, can be exploited to behave in a non-anticipated way. A common exploitation example (with file locks) is to gain access to other user's accounts through the *passwd* programme in UNIX-systems by using symbolic links. Semaphores address this problem produced by the introduction of multitasking in Operating System design by implementing a critical region.
Integer Overflow	Although integer overflows are much more difficult to successfully exploit and almost requires a special case to be useful, it is a noteworthy vulnerability found in popular language implementations. The main concern is what value is assigned to a variable when the value to be assigned is larger than allocated for an integer whether it is 16 bit, 32 bit or 64 bit, or whatever size. An unsigned 16 bit integer can hold no

	larger value than 65535 for example.
Cross-site scripting	Typically found in web applications, attackers exploit a web server that does not limit pages' elements to the local server. This remote page, which might include malicious code, is then viewed by users visiting the site. The intent is usually phishing that is, pretending to be someone else, and using the gathered data input from the user (which might be a username and password of a user's bank details) malignantly.
SQL injection	SQL injections involve a special character (the single quote) being input by an attacker to confuse the parsing of arbitrary code as SQL statements. SQL injections are most commonly used to gain unauthorized access to administrative consoles of websites. SQL injections as well as format string attacks can be categorized under code injection exploits.

Table 3.1 Common exploitable software vulnerabilities

The explanation of the anatomy of an attack is important because it makes it easier to explore the different countermeasures that may be devised to mitigate against attacks in the next chapter. It is important to note that security measures should be seen as trade-offs between cost and the level of security. Risk analysis should therefore be conducted in order to assure that the security level matches the company profile. It is senseless to secure a system with the latest technology when that level of security is not necessary at that cost, and vice versa.

4 SECURITY CONCERNS OF THE WIRELESS VOIP SYSTEM

4.1 Introduction

In this chapter we discuss some of the common vulnerabilities that are possible in our system that is presented in Figure 4.1. It is crucial to first fully understand the cause of the vulnerabilities; the effect exploitation has as well as the solutions to them, because as was stated earlier, vulnerabilities are often oversight errors or unanticipated special cases. The attack vectors and security measures investigated include buffer overflows, brute force attacks, SQL injections, denial of service attacks, sniffing and encryption techniques. With wireless encryption, two different encryption techniques that are widely used in practice, namely WEP and WPA are investigated. Demonstrational exploitable applications were developed in order to demonstrate the different types of vulnerabilities (they are available on the disk attached to the back cover of this dissertation). The design flaws of these vulnerable programmes, or code snippets, are discussed in this chapter and the results are presented in the next chapter by showing their exploitation.

4.2 The Wireless VoIP System

Figure 4.1 depicts the workings of each node of the VoIP System. Although the system is medium independent and can be operated over both wired and wireless networks, a wireless network, which could be penetrated, is assumed.

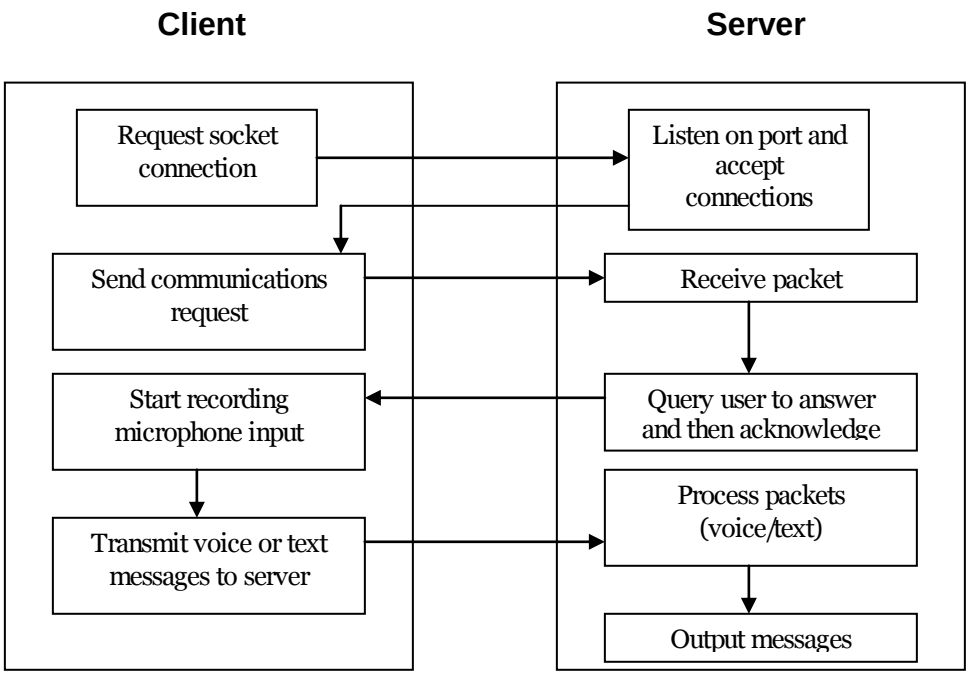

Figure 4.1 The VoIP model

The system has both a client and a server, which connects to the other hosts with which a conversation is in progress. Figure 4.1 (included again for readability) best describes a conversation to the local host. The server section (which runs in its own thread to allow concurrency and multiple calls) listens on a port, awaiting a client to connect to it.

When the server detects that a client has connected to the network socket, it accepts the connection and network communication may commence. The client detects that the connection is established and sends a request to the server to initiate a voice call. Upon receiving this request, the server prompts the user to take the call.

If the user declines the call, the client is notified and the connection is terminated to free up resources. If, however, the user decides to take the call, the server spawns a new client that will record microphone input and send the speech to the server on the other node, and acknowledge to the client that the call may

proceed. When the newly spawned client connects to the server of the remote machine it detects that it is already in conversation with the other host and this prevents it from spawning a new client again and causes an infinite loop. The server merely accepts the connection (without prompting the user again to accept the connection).

Both clients thereafter start recording and buffering microphone input. The encoded voice is then sent to the remote server which decodes it and plays back the speech over the speakers.

4.3 Vulnerabilities

4.3.1 Buffer Overflows

Buffer overflows, although limited to the C and C++ programming languages, are important to consider as current and future VoIP implementations may be written in this popular language and buffer overflows are probably the most exploited vulnerability. It is important to note how a process, or an application, is organised in memory, before we delve into the details of buffer overflows.

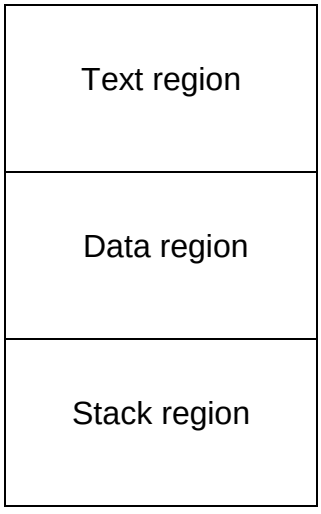

Figure 4.2 The constituents of a process

There are three major distinguishable parts of a process map as shown in Figure 4.2: the text region, the data region, and the stack region. We describe the first two parts and a description of the last part is given later. The text region contains the instructions compiled by the compiler. This region is fixed, and any attempt to overwrite it will interrupt execution and raise a segmentation fault. The data region on the other hand contains static variables, which are writeable. Details of memory management and the Virtual Address Space (VAS) are described in [9]. A stack is a very popular data structure used in programmes supporting several operations, including PUSH, POP and PEEK. Stacks follow the “Last In First Out” (LIFO) scheme. The stack expands and collapses at runtime as execution of the programme continues. Most high-level programming languages have support for functions, which temporarily alter the flow of control by executing a different set of instructions located at another memory address, and thereafter returns control to the instruction following the function call. This flow of execution is implemented at a low level by means of the stack, although it is not the only function of the stack.

The processor contains registers, which is an important aspect of this discussion. Registers vary from manufacturer to manufacturer as well as model to model. For this work a 32bit Intel-x86 processor is assumed. Processors contain user-accessible registers, which are addressable during application programming, and programme inaccessible registers, which cannot be addressed directly, but may be addressed indirectly through system programming. Some registers are general-purpose while others are used for special purposes. 8-bit, 16-bit, and 32-bit registers are allowed to be referenced. The 8-bit ones are AL, AH, BL, BH, CL, CH, DL and DH (The L indicates the “Low” and the H the “High” order bits. A through to D are sequential numbers to distinguish between the registers). The 16-bit registers are AX, BX, CX, DX, SP, BP, DI, SI, IP, FLAGS, CS, DS, ES, SS, FS, and GS. Finally, the extended 32-bit registers are EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI, EIP and EFLAGS (the “E” prefix indicates that it is a 32-bit register). The noteworthy registers are the general purpose registers (EAX through to EDX), EBP, which are

often used to point to a memory location for data transfers, and EDI, which is often used to point to the destination address of an operation. The important special purpose registers are EIP, which is used to point to the next instruction (within the instruction segment) to be executed, and ESP which is used to point to the stack.

There are two important registers we need to consider with respect to stack operations: the Stack Pointer (SP), which points to the top of the stack, and the Frame Pointer (FP) (EBP on 32-bit Intel architectures), which points to a fixed location in a frame. The FP is used by many compilers to keep track local variables' and function parameters' memory addresses, as the relative distance between FP and the variable is not influenced by PUSH and POP stack operations.. The base of the stack (where the remaining firstly pushed element is located) is at a fixed memory address. Depending on the implementation, the stack expands (as elements are pushed onto the stack), and therefore changes the value of SP, either downwards (towards lower addresses) or upwards. On the Intel processor, the stack expands downwards, therefore the demonstrations in this work are presented in such a way. Multiple instructions are executed both during a function call, and during its return. A simple stack example follows in Figure 4.3, which is similar to the self-exploiting demonstration "bof.c" included on the disc.

```
void calledFunction(string param) {
  char arrString[3];
  strcpy(param, arrString);
}


void Main() {
  string x = "xyz";
  calledFunction(x);
  printf("x = %s", x);}
}
```

Figure 4.3 A Buffer Overflow Vulnerability Example Code

The assembly code for the above listed programme is then compiled. Details of the assembly code will not be discussed here, but it is important to note that when calling the function *(calledFunction)* that the value of the FP (EBP) is PUSHed onto the stack. Thereafter, the value of the SP is copied into the FP, rendering the current top of the stack as the FP value. Additional space for local variables is then allocated by first determining the required size. This is done by taking their size and subtracting (for downward expansion of the stack, added for upward expansion) it from the SP.

It is important to note that the word size plays a crucial role in addressing memory, as it can only be addressed in multiples of the word size. A 32 bit or 4 byte architecture was used for this work. Since the 32 bit (four byte) architecture was used, the buffer sizes will be in multiples of 4 bytes. For example, a 16 byte buffer will take up 16 bytes of memory (4 x 4 bytes), a 32 byte variable 32 bytes and so on. It is important to note, however, that a 5 byte buffer (or a 5 element array) will actually take up 8 bytes in memory, when using a 4 byte (32 bit) architecture since this is the smallest possible buffer size in a 32 bit (or 4 byte) architecture that can accommodate 5 bytes. The following allocated variable will start at the next multiple of the 4 byte offset. A sketch of the example code is shown in Figure 4.4.

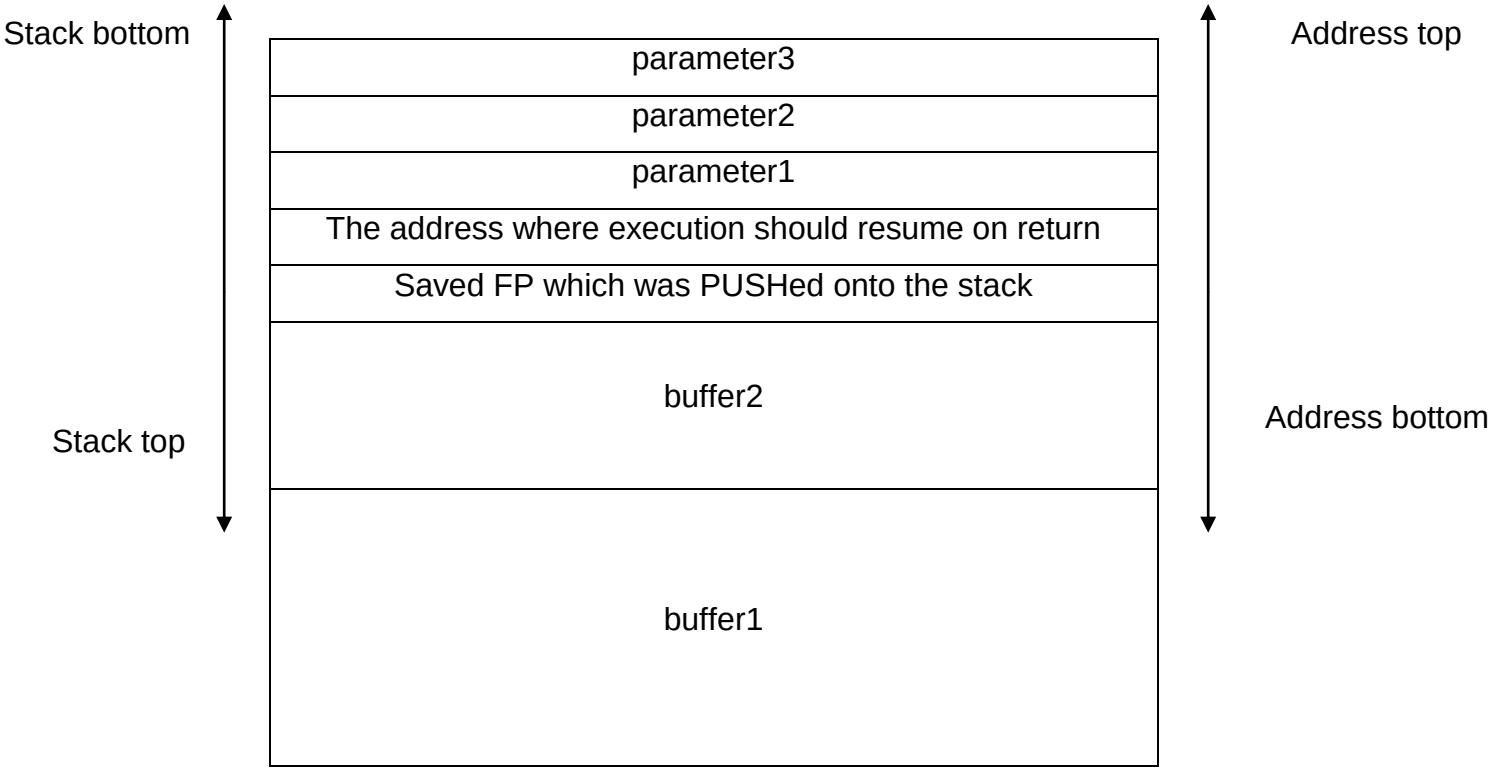

Figure 4.4 Memory allocation at runtime

A question that one may ask is: “Why not simply send a large amount of junk that will overflow the buffer?” This is indeed a valid question. If the buffer is simply over-flown without taking care what is overwritten, the programme will most probably crash and make a core dump. On the other hand, if a buffer overflow is done in a very precise and calculated way, it becomes possible to execute arbitrary code, which is not even inside the programme. What an attacker usually does is to craft a special payload, also known as *shellcode*, which is sent to the application (possibly over the network) and which overflows the buffer. Since the targeted buffer is of a certain size, and also a specific number of bits away from the return address, which is the target to be overwritten, the payload needs to be extremely precise.

It should be noted that the payload will stretch from the overflowing buffer’s address up to the return address, and that this block should be viewed as a contiguous

memory section, although it is actually allocated to several variables. In this case the *shellcode* will consist of a continuous series of NOP (no operation) instructions (also referred to as a NOP slide), the instructions to be executed, as well as a guess at the wanted return address (which should point somewhere inside the NOP slide). The NOP slide is a series of no-operation instructions deliberately added to the *shellcode* in order to make it easy to guess the correct return address. The overwritten return address can point to any region from within the first NOP instruction, up to the last one, just before the first arbitrary instruction. The content of the section in between the last instruction, which is usually an exit call, does not really matter, only the size matters, so that the exact address of the return address can be overwritten effectively, pointing it towards the arbitrary instructions.

Although *shellcode* is a very important part of exploiting a buffer overflow, it will not be discussed further, since information regarding *shellcode* is abundantly available on the web. Some even include pre-compiled *shellcode* which can be used on various platforms. A tutorial on writing shellcode can be found in [9].

We developed a sample programme with a buffer overflow vulnerability that exploits itself and is included as “bof.c” on the disk. Consider the code in Appendix A for the following discussion of this programme. The global variable *shellcode* is used to store the instructions to spawn a shell (*$/bin/sh*), which was obtained from *metasploit*. The *main()* function is invoked as soon as the programme starts executing. Thereafter the function named *function()* is called. As was mentioned earlier, the important thing to note is that at a lower level, a return address is pushed onto the stack telling the processor where to return execution flow when the function is completed (which is at the line below the function call and, in this case, the end of the programme). As execution is passed to the called function, two character arrays are declared as well as an integer pointer named **ret*. This pointer will be used to get a handle to the return address and ultimately change its value. We get the return address’s location in memory by some simple math. Because we have addressable locations in memory (the arrays’ addresses), and we know that the return address is

stored in a location within a certain offset from the function's local variables, we can assign the pointer to point to the address of the return address and ultimately change its value to whatever we want. In our example we simply set its value to the address of the *shellcode* variable, so that execution is passed to that address when the function terminates. This spawns the shell by issuing the command contained within the shellcode variable ($*/bin/sh*).

Most other languages avoid this issue by bounds checking memory transactions. Whenever a read/write attempt to an illegal memory location is detected the application's execution is terminated. As this needs to be done at runtime, it is of course a performance trade-off.

4.3.2 Brute Force

Although we did not devise and implement strategies to mitigate against brute force attacks, it is a commonly used attack and therefore deserves to be investigated what it is and how attacks are expected to be launched. Brute force attacks are mainly concerned with guessing correct usernames and passwords. An attacker would most likely write a small programme that understands the authentication section of the relevant protocol. The programme will automatically cycle through username and password combinations until the correct one is guessed and then log in using this correctly guessed username and password. There is a tool named *Brutus* [10], which supports a large variety of protocols that can be used to make a brute force attack against supported protocols. These attacks often take on specialised forms such as wordlists or dictionary attacks where the attacker cycles through popular username and password combinations rather than trying every possible key combination. This drastically scales down the number of attempts and therefore the time it will take to break the authentication (assuming the username and password combination is on the list).

Strong passwords are an alternative solution attempting to solve the problem by leaving the burden of remembering a difficult password with the user. Although it might seem like a reasonable solution, users who are told not to base their usernames and passwords on English words and that they should contain numerals as well as alphabetical characters, usually use some name (typically their own or their spouse's) and a concatenation of numerals such as "1234". This can indeed result in strong passwords, if the attacker is using an English dictionary, for example, to crack the username and password.

Mutnick's book "The art of deception", which has a strong focus on social engineering suggests that the user is the weakest link in the security of most systems [11]. They are easily deceived into entrusting an unauthorised person with classified information such as a username and password. It would therefore be best for a user to have as little as possible to do with authentication. This in turn can be solved through storing this information on the user's computer, but it is unfortunately at the cost of mobility. It is therefore desirable to suggest, or perhaps even enforce, a sensible password policy by forcing the user to specify a password based on a secure regular expression. Care should be taken with such an expression, as a single wildcard out of place can drastically influence the possible number of combinations or produce other detrimental effects that compromise security. Since the SIP standard is implemented, and no central server has been implemented, a temporary workaround which will store user credentials has been made for purposes of completeness of our VoIP application. It is completely unacceptable practice to store others' usernames and passwords on potentially distrusted sources (that is on an unencrypted local file on the client's PC).

4.3.3 SQL Injections

To illustrate how common SQL injection vulnerabilities are, one can search for the string: 'inurl:.co.za "Administrator" "Login"', excluding the single quotes, with Google and it will return all the hits of Administrative login consoles with a ".co.za" extension

indexed by Google. The results returned will contain a listing of all South African business domains' web login consoles. One can then try to just insert a single quote into either the username or password fields, and when this generates an error relating to a SQL query, the chances are that the site is vulnerable to an SQL injection. This is one of the most common methodologies for defacing websites. It is surprising how many vulnerable sites there are on the Internet, as a company's homepage contributes largely to its public image.

Imagine a web site development company whose website was defaced. One would not entrust such an organization with your own website if they are unable to uphold their own. It is far from uncommon to see defaced web design websites, which again points to oversight errors. This may be one of the greatest reasons companies tend to be secretive about their security status. Many reports have shown that organizations deny that their websites were defaced, as they are only interested in resuming business rather than bringing the culprits to justice.

Consider the pseudo code in Figure 4.5.

```
recordSet result;
result = SQLobj.execute("SELECT * FROM USERS WHERE USERNAME = '" +
user_supplied_username + "';");
If (!result.isEmpty()) {
    call LoggedIn();
}
else {
    call accessDenied();
}
```

Figure 4.5 SQL injection illustrative sample code

Firstly, an SQL query is executed against a database and the record set of all the records are stored in the *recordset* variable. Since the “USERNAME” field in the “USERS” table in our example is the primary key of the table, the record set will either contain one entry or be empty. After execution of the SQL statement, the result is evaluated for being empty. Thereafter the appropriate function is called to either notify the user that he is logged in, or that access is denied. In this example “*user_supplied_username*” is a string which was typed in by a user wishing to log into the system. If the user supplied a string such as

anything’ OR ’1 ’ = ’1

for the username, the SQL query’s semantics drastically changes. The query will read as follows: (note that the “user_supplied_username” was replaced by its value. It is highlighted below to clarify.)

“SELECT * FROM USERS WHERE USERNAME = ‘ ” + “anything’ OR ’1’ = ’1” + “’;”

And now the statement is given as it is passed to the database management system:

“SELECT * FROM USERS WHERE USERNAME = ’anything’ OR ’1 ’ = ’1’;”

What happened was that the SQL code was injected into the statement – hence the name SQL injection. Since the OR operator needs only one of its operands to evaluate to “true” in order to execute its command (selecting a username, which is done in a loop), the query successfully executes, listing every “USERNAME” in the “USER” table, because 1 is always equal to 1. Furthermore, the evaluation of the *recordset* being empty thus takes a different route. Rather than being empty – for an incorrect username supplied it holds a whole set of records. It thus satisfies the evaluation of not being empty and passes execution onwards to the successful login.

Apart from an attacker using SQL injections to log into a system, other arbitrary code can be executed in conjunction. If another semicolon was injected to delimit the end of one statement and the beginning of another, a whole different dimension of attacks against the database opens. A malicious attacker could inject a drop table, or drop schema statement, or replace important data with garbage, corrupting the database.

A sample programme named “SQL.exe”, written in Visual Basic 6, was included on the CD to demonstrate the effect of SQL injections. We refer to the code in Appendix B in reading the following discussion of the code. The sample depicts a typical user login example where the user is presented with a prompt to supply a username and password to log into the system. The problem with the programme is that it fails to check for and take appropriate action against a delimiter. When a single quote is given somewhere in either the username or password fields, the input afterwards is treated as SQL code. This will most likely lead to a malformed SQL statement, but if care is taken to inject the correct code, a positive login is possible without a valid username and password. We can remedy this issue by searching for and handling special delimiters such as single quotes before concatenating user-input to SQL statements passed to the DBMS.

4.3.4 Denial of Service and Distributed Denial of Service (DoS and DDoS)

As with bruteforce attacks, no measure is implemented as a part of this work, but we attempt to broadly outline the concerns faced with DoS and DDos attacks. Denial of service attacks encompasses a large number of requests sent to a target which renders the target unable to handle these requests, as well as other legitimate ones. An attacker executing this type of attack is generally interested in having the remote server deny service to everyone else, including legitimate traffic, and may use a wide variety of techniques to keep the server application/thread too busy to respond to any other queries. These methodologies may range from making a connection

and immediately dropping it in a loop, sending large amounts of junk that must be processed, or even keeping the user busy with senseless conversation so that the user is unable to take any other calls (which is more of a social engineering and denial of service combination). In order to identify a denial of service attack, one needs to monitor some variable being repetitively changed, while some other criteria that make it associated with the same attack remains the same.

Devising defence mechanisms for this type of attack poses major challenges. For example, limiting the number of connections in a specific time frame from a certain IP address is a possibility, but unfortunately only as long as the attacker does not know that it is his IP address that has been blocked. An attacker could create a script that opens a connection, closes it and then changes his IP address, all in an infinite loop, rendering this solution ineffective. The same principle is applicable to MAC addresses and any other variable the attacker can alter. It is therefore problematic to link several connections to a single attacker and properly block the attack attempts.

The situation outlined above is hypothetical, though everything mentioned is possible when operating over a LAN. Changing an IP address on the Internet is much more troublesome (depending on the method by which one is connected to the Internet). Virtual private networking (VPN) is another possible solution, which excludes unauthorized parties from gaining access to the application layer. If the attacker is not “on the same network as the target”, he cannot launch an attack. Also, the connection speed (especially over the Internet in South Africa) may be very low, ruling out the possibility of the processor becoming too busy, but the bandwidth resource will still be in jeopardy from such an attack.

Access controls, from the perspective of this research does not fall in the application layer of the TCP/IP reference model and is thus beyond the scope of this work. The problem with access controls, specifically in the case of VoIP, is that one would like to be reachable by anyone on the Internet (the public). VPNs and other access

controls serve their purpose well for business applications stretching across large geographical areas where only some parties should have access. Although filtering, or rather temporarily blacklisting, MAC addresses is not a sufficient solution for VoIP, it could easily, and efficiently be implemented as the connection could be dropped with very little resources going to waste. A combination of some criteria such as MAC address, IP address, etcetera, would offer a much more secure solution, or an even better one - that randomly changes the combination of filtering criteria. One should, however, take great care not to deny service to legitimate users as a side effect of mitigating against these kinds of attacks. Such a setting would in itself produce a denial of service situation.

4.3.5 Encryption

4.3.5.1 Sniffing and encryption

A *sniffer* is a device, or piece of software capable of monitoring information travelling across a network. They are virtually impossible to detect and can theoretically be inserted anywhere. The only two methods to prevent information from being intercepted is to use a protocol the sniffer does not understand; using IPX instead of TCP, for example, or to make use of encryption. Packet sniffers may also be used by attackers on a compromised machine. Its purpose could be to collect sensitive data that is sent out over the network, or retrieve usernames and passwords of all incoming logins to a server.

Many authentication protocols make use of a process where the plaintext is hashed by the client, sent over the network, and each entry in the database is retrieved by the server and also hashed. The two hashes are then compared one-by-one with each other to validate authenticity as un-hashing would take far too long. VoIP's digitised data transmission can unfortunately not rely on the digitized voice being compared with a table, as the data can have far too many combinations to be efficient; it is not feasible to store this data in a table. It is therefore necessary to be

able to decipher the encrypted audio when it is received by the remote host. VoIP therefore requires that the only secret is the key. The algorithm may be known, but each session needs to have a secret key, which if compromised would unfortunately compromise the secrecy of the conversation.

Sniffers are especially effective with protocols that use clear text such as the File Transfer Protocol's (FTP) since the username and password can be read off the sniffer's logs.

A sample application has been written in C# which allows two users to connect to each other and send text messages to each other. The user has an option to encrypt the data that is sent over the network or not. If the data is chosen to be encrypted, the user needs to specify a key with which the data is to be encrypted and decrypted. If the users are not using the same key, all the received messages are illegible. This is also true for someone using a sniffer. If the person using the sniffer does not know the algorithm and the key, he will not be able to understand the messages.

As far as the encryption scheme's key generator is concerned, another well known weakness exists with random number generation. If for instance a key of 40-bits was to be generated by a random number generator, the randomness is often limited, and thus less secure than the length of the key itself suggests. 40 bits is supposed to generate 1099511627775 possible combinations for a key.

The generator may be flawed in such a way that it actually is only capable of generating say only 100000000000 unique numbers. With this hypothetical case there are only 37 bits of randomness and the key strength is limited to that level of encryption strength. An estimate was that the average 128-bit session key for Secure Sockets Layer (SSL) contained only 47-bits of randomness. As the strength of the encryption is exponentially dependant on the (effective) key size, the status quo of decryption methodologies poses a severe threat to such flaws. For example,

an 11-bit key is twice as hard to break as a 10-bit key and a 20-bit key around one thousand times as difficult to break.

A sample application was written in Java that selects a random number between 1 and 10. The application generates 1000 random numbers and calculates how many times each number (between 1 and 10) was selected. In theory, each number should be selected 100 times (1000 / 10), but as the generator is random, one will have different results every time, but the closer the number is to 100, the better.

4.3.5.2 Wireless encryption - WEP and WPA

In [14] the vulnerabilities of commonly used wireless encryption technologies are discussed. The WPA-PSK (public switched key) protocol was found to be “crackable” on November 4, 2003. The method, as opposed to cracking WEP, is active rather than passive and is more easily detected.

Different perceptions exist on how wireless devices should function. Two popular schools of thought are that wireless systems, especially access points, are devices with very limited computational power, (temporary) storage and bandwidth, and the contrary. These different perceptions’ implementations will need different designs in order to perform optimally. This is especially true from a security viewpoint and the encryption strength.

Tanenbaum noted in [13] that in the race between computation and networking, networking is far in the lead. The bottle neck is when the signals are converted (computation) from light to electricity and vice versa. Although this was said with fibre optics in mind, laser is also used in wireless applications with a transmission of the speed of light as well.

5 IMPLEMENTATION AND RESULTS

Microsoft Access was used as a Database Management System (DBMS) for storing and querying all types of necessary data, as it is a flat file and not resource intensive. Java was used as the programming language, as it has built in support for the JET engine required to connect with the Microsoft Access database, and Java is designed with security and stability in mind. Buffer overflows are automatically ruled out when using Java, as runtime bounds checking is implemented into Java. Java is also widely supported across platforms. All the required libraries and data structures, such as linked list objects, are natively available in Java, except for a proper means of implementing a tray icon. JDIC (a binary distribution of a tray icon interface) [15] is freely available and is based on Swing, which is also native Java for the purpose of the tray icon.

In the previous chapter, we investigated what causes all the vulnerabilities in the scope of this work, how they can be exploited, and what effect exploitation has. In this chapter we investigate possible measures which can be used to mitigate the risks regarding these vulnerabilities and we present the results of the wireless protocols investigated. It is important to emphasize that it is an absurd statement to advocate that a piece of software as completely secure as vulnerabilities derive from oversights, which is simply a flaw that was not anticipated. The software developed in this work is therefore designed with security in mind, but can under no circumstances be labelled as completely secure.

5.1 Buffer Overflows

Although Java has built-in support for preventing buffer overflows from occurring, it is crucial to point out what operations enable the overflowing of buffers in the C language, as buffer overflows are one of the most exploited vulnerabilities in practice. Several library functions are known to ignore bounds checking arrays (primarily because it is a performance trade-off) such as *strcpy()* and *sprintf()*. Additionally to avoiding the use of these functions, one should be careful as to

what operations are executed on buffers (as well as regular variables along with the usage of pointers) within the application. The following code snippet copies one array onto another, character by character, without bound checking the target array (it only checks for a delimiter in the source string which is used by the C language to terminate strings), which results in the buffer overflow vulnerability.

```
while(*ptrBuff1 != '\0')
{
        *ptrBuff2 = *ptrBuff1;
        ptrBuff1++;
        ptrBuff2++;
}
```

Figure 5.1 Buffer overflow vulnerable code

Both the *client and server* sockets of the previously implemented VoIP system have been tested for handling abruptly large buffers with the intent to overrun them, but because Java is interpreted, and does bounds checking at runtime, it did not disrupt the programme from normal execution. Upon receiving the large buffer, an error was raised, the overflow of the buffer was ignored and normal execution resumed. The following figures show the results obtained.

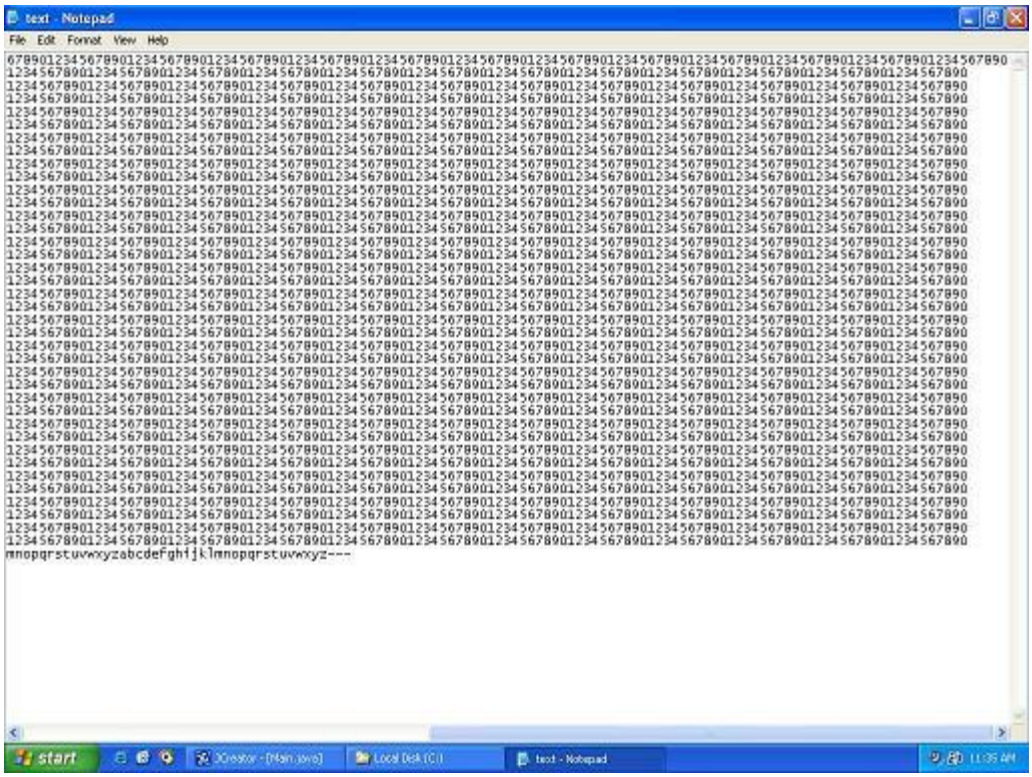

Figure 5.2 Text file used as payload

Firstly, the above text file was created to overflow the buffer. This file starts with the string “/dlg”, which is interpreted by the VoIP application as a text message that should be displayed in a popup window and the end of the text file is delimited with “xyz---”, which will be used as proof that the payload is indeed larger than the buffer, and that it should overflow if the vulnerability exists. We used the Windows port of *netcat* (nc.exe) to pipe the contents of the text file to the socket of the VoIP application by issuing the command in Figure 5.3 below:

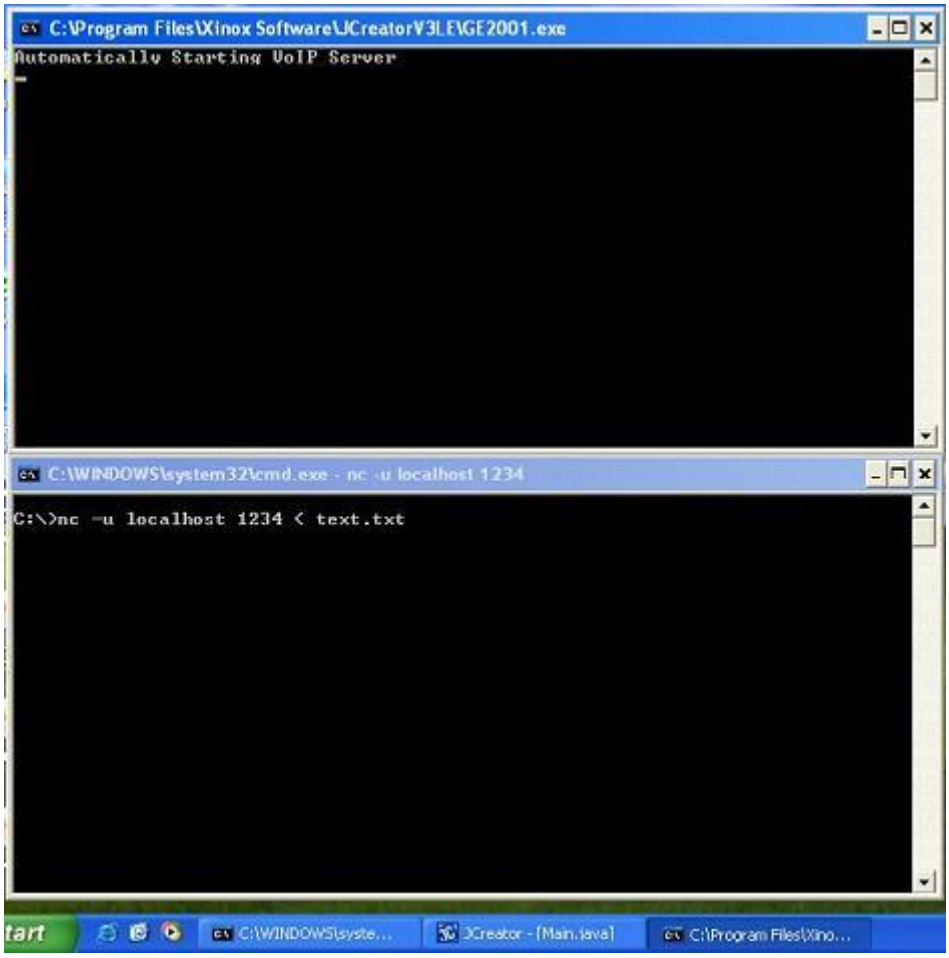

Figure 5.3 Using netcat to attempt overflowing the buffer

As can be seen in Figure 5.3, no segmentation fault occurred when the payload was sent to the application. Fig 5.4 below shows the popup window that was output by sending the payload, which proves that a larger payload was sent to the application, as the terminating string from the text file ended in “xyz---”, but the popup message ended in “xyz-”. The last two characters of the payload (“--”) were simply truncated without causing a segmentation violation.

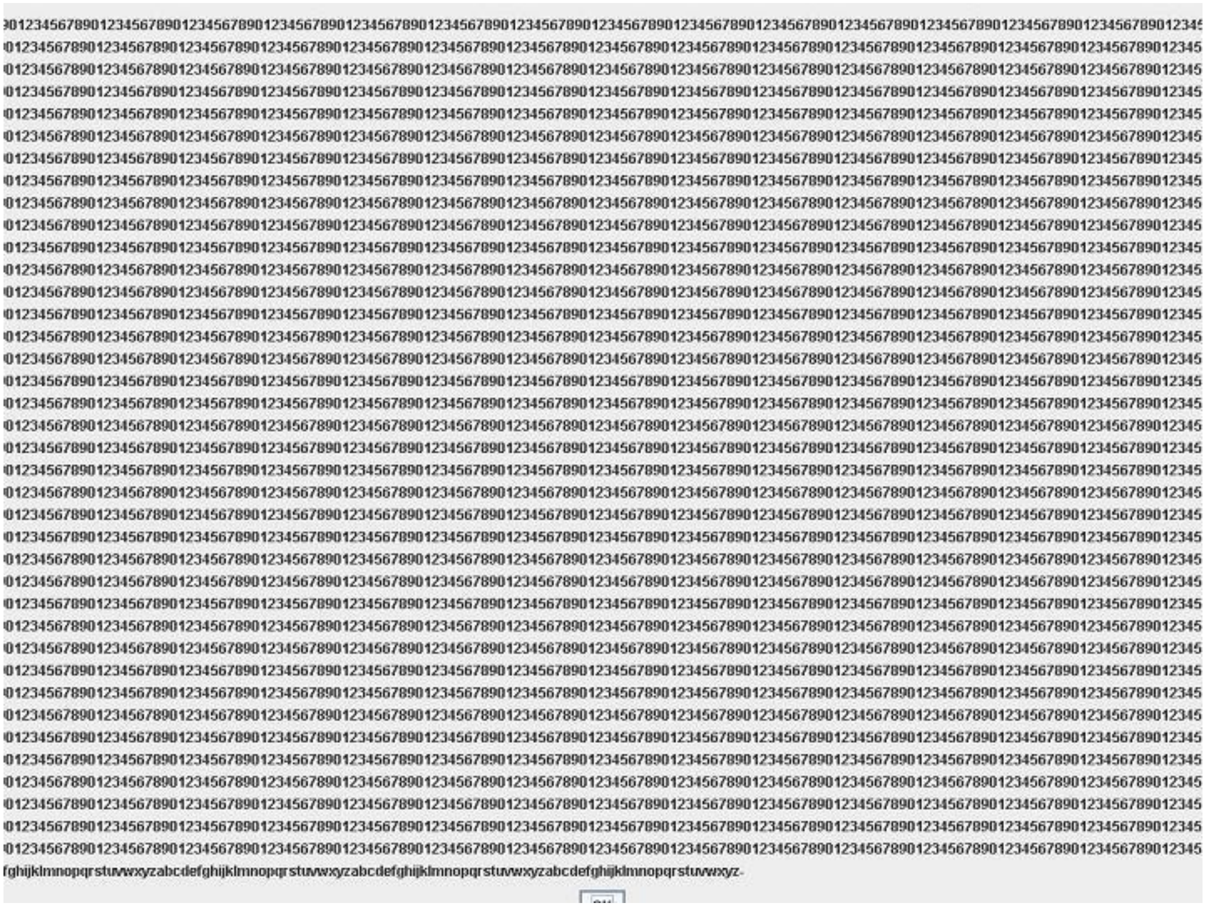

Figure 5.4 The received popup message

When receiving a larger packet than compensated for, the default mechanism of the *DatagramPacket* class constructor is to truncate everything after the offset specified by its length parameter.

5.2 Brute Force

An effective and a widely used solution could be to set timeouts on usernames. If a user wishes to log in, for example, but specifies an incorrect password, the user's account is temporarily disabled. Obviously, the longer the period of the disabling, the longer an attacker would take to successfully brute force the username and password, but one should take into consideration that humans are error prone and might enter their usernames and passwords incorrectly, and lock out their own accounts. Alternatively, a malicious user could enter wrong passwords for a known username on purpose to have the account locked out.

This is why the time which an account is locked out should be decided on very carefully.

Let us consider a scenario where a malignant attacker knows the username of someone, and wants to brute force his account. Let us say that the attacker knows that the password consists of a single alphabetical character (a through to z), as the hypothetical password policy is enforced this way. Knowing this, the attacker may narrow down the number of possible passwords to 26. If the scenario (processor speed and network resource availability) allowed him to attempt one password per second, he would be able to crack any password under half a minute. If, however, the system decides to disable the account for one minute after each incorrect attempt, the attacker would in the worst case be able to crack it just under half an hour. This effect is exponentially more visible as the time which the account is disabled increases as well as better password policies are enforced. This solution could render brute force attacks completely unfeasible.

An alternative mechanism for dealing with brute force attacks was implemented by tweaking the SQL injection demonstration programme. Figure 5.5 shows a screenshot of the example.

Figure 5.5 User interface of the brute force demonstration

The system keeps track of every unsuccessful attempt made to guess the password of the given username ("Username"). When the number of unsuccessful attempts is more than a preset number (in this case 3), logging into the specific user's account is disabled until the value representing the number attempts is changed to a number lower than 3. As was stated earlier some field is monitored to be repetitive, while another changes. In this case the account is locked on the grounds that a specific account (based on a username) has numerous password attempts.

5.3 SQL Injections

As for code injections, and particularly SQL injections, all possible input from the user, including, but not limited to all information arriving over the network, should

be scrutinised for special delimiters such as single quotes for SQL injections and appropriate countermeasures should be taken. Since these special characters may appear naturally and validly in some cases, especially in the digitized audio as well as conversational text messages, the policy should be refined to fit the purpose of the input. There is no sense in delimiting special characters that are not evaluated in databases or which may not form part of an SQL query (such as usernames and passwords) to the database.

An application was written in Visual Basic 6, to demonstrate SQL injection vulnerability. The application presents a typical login screen with a username and password field. The application fails to remove, or delimit, special characters such as the single quote (‘). It is possible to successfully log into the application without knowing the username and password.

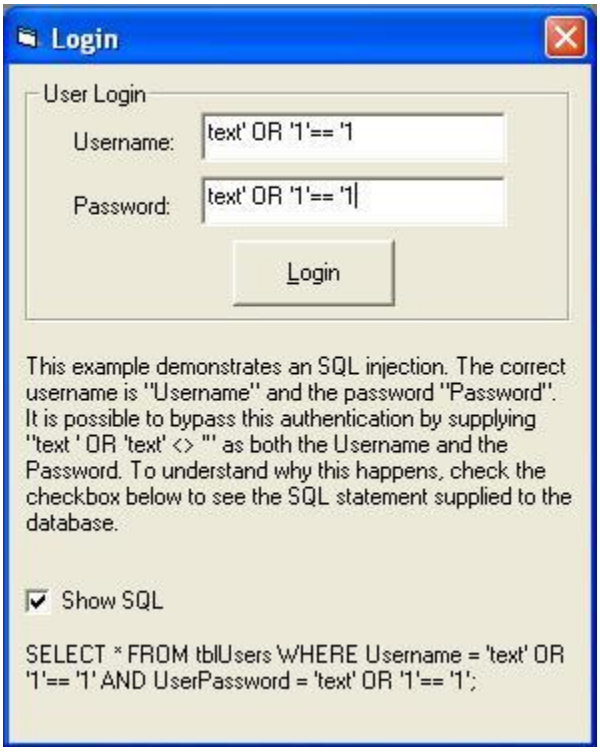

Figure 5.6 The SQL injection vulnerable application

The following pseudo-function is suggested to make the application invulnerable to SQL injections:

```
void Addslashes(string var)
{
        foreach (char character in var)
        if (character == "' ")
                character = "\' ";
}
```

Figure 5.7 Addslashes function

Whenever a single quote is identified, the character is delimited by a "\". SQL injections were thus eliminated at the obvious level. As most vulnerabilities are unintended errors, and overlooked, we cannot say that the implemented system is completely immune against SQL injections. This is even more important when other vulnerabilities are discovered and exploited in a combination to successfully accomplish the desired exploitation result. The suggested method is a common methodology to handle SQL injections in web applications and particularly PHP (although there is no official full name for PHP, a common one is "Professional Home Pages") applications before adding the string to an executed SQL statement. This effectively includes the single quote in the data, without having it be parsed as a string separation token.

5.4 Denial of Service and Distributed Denial of Service (DoS and DDoS)

Preventing or recovering from a distributed denial of service attack is a complex issue, as it is difficult to distinguish between the perpetrators and legitimate users, which may require manual intervention. Best practice suggests tearing down the connection and freeing up resources as soon as it has been determined that a malignant attack is taking place. New TCP/UDP connections could be rejected, even before the three way handshake has taken place to ensure that as few as possible resources are devoted to the culprits. A serious

problem with this approach is that the grounds on which the user is blacklisted (as in username, IP address, MAC address, etcetera) may prevent legitimate users from making use of the service.

Denial of service as well as distributed denial of service attacks are difficult to automatically detect and is often best solved by manual intervention [25]. An administrator would typically search for a pattern (such as every computer in the distributed denial of service attack is downloading the same file, which makes it relatively safe to assume that everyone downloading the file are involved) which is often difficult to anticipate. Anticipation of certain patterns is an insufficient measure, as security should not be enforced through obscurity, which leaves the attacker free to search for any unanticipated pattern. In the event of a distributed denial of service attack, an administrator may typically prevent all IPs involved from routing to the source of contention. Such a solution does not lie in a single layer of the OSI model, and cannot be handled most efficiently in the application layer, as lower layers would save more resources.

To demonstrate an IP address filtering scheme, the DoS.exe programme was developed. The results are presented in Figure 5.8 and 5.9.

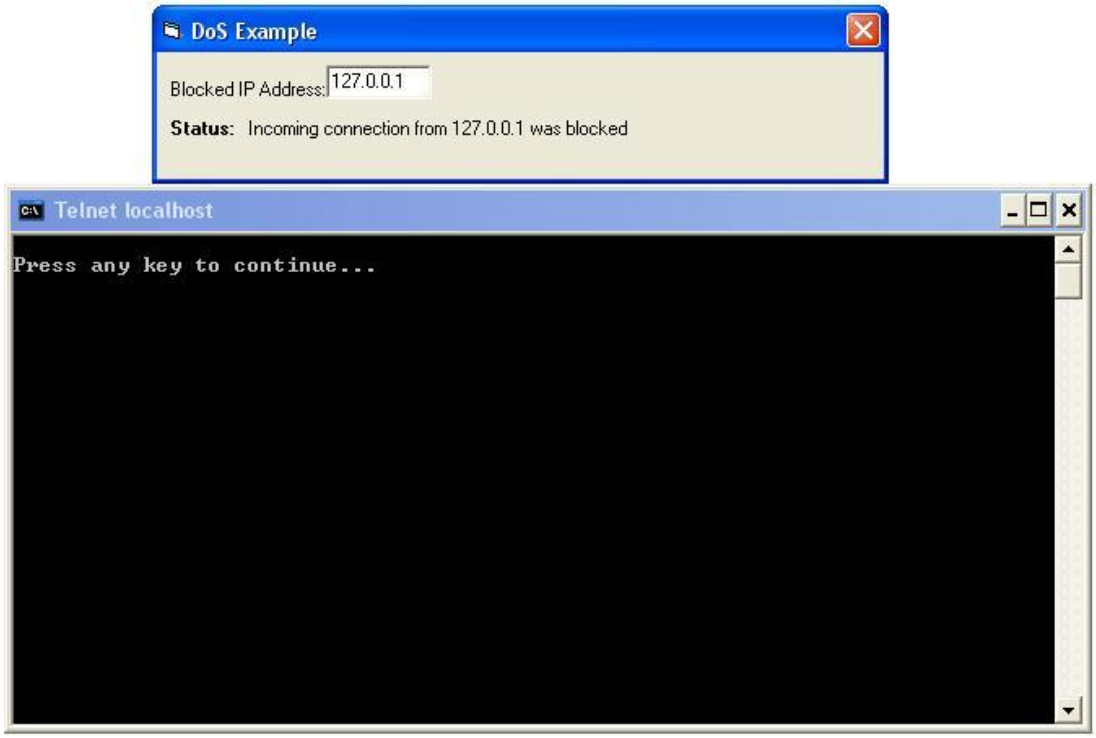

Figure 5.8 A blocked connection attempt

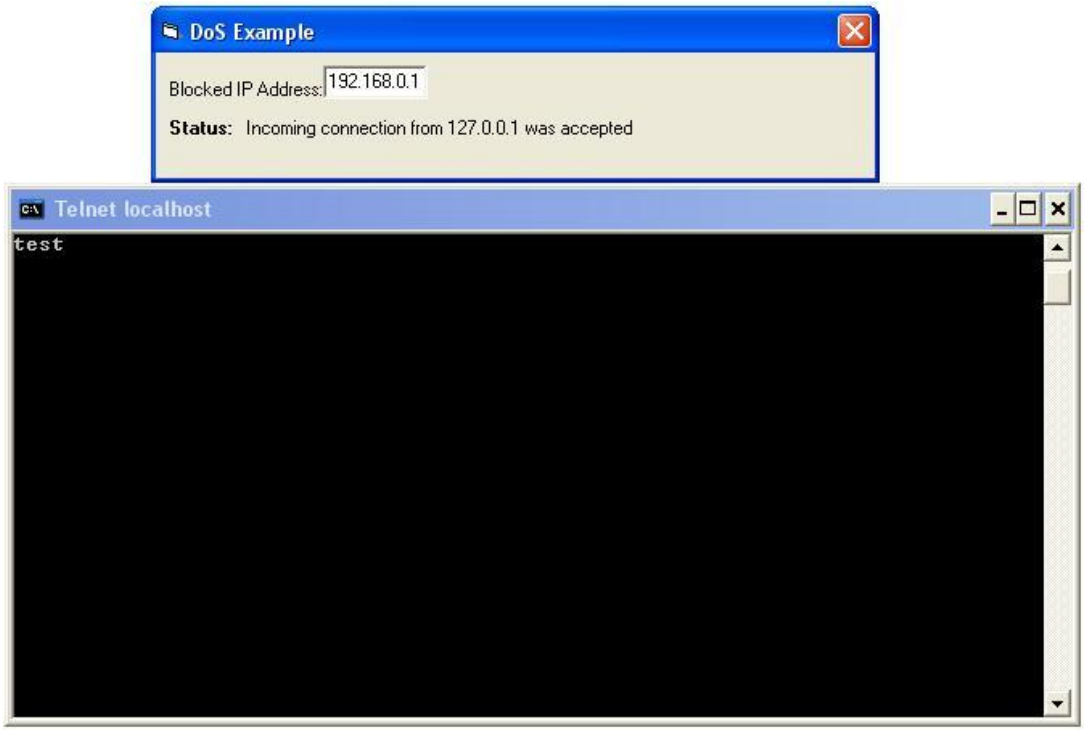

Figure 5.9 An accepted connection attempt

The first screenshot shows a blocked connection attempt. The server (DoS.exe) was listening on port 1000 and 127.0.0.1 (which is the IP address for localhost) was blocked. When a telnet connection was made to the server socket, it was immediately disconnected, as the server detected that the connection request was coming from the IP address being blocked.

In Figure 5.9, however, the connection attempt was accepted, as the IP address being blocked by the server was not equal to the source of the connection request (127.0.0.1).

5.5 Encryption

5.5.1 Sniffing and encryption strength

Encryption was not implemented into the VoIP application, and was separated from this as the encryption demonstration was implemented as a separate stand alone C# application. Here is an intercepted message, as well as its plaintext counterpart:

Intercepted data (string representation)	Plaintext Message
?,!t0&i.>a6<'*8&.l%2=&t40:<1&6j	This is an example text message.

Figure 5.10 Demonstrating encrypted data

This application uses a simple method of encryption. The XOR operator is simply used on the payload along with a key. Although unintelligible, there are much stronger algorithms for encryption that should be used. From this data it is clear that when encryption is enabled, the messages are unintelligible. The digitized audio should also be encrypted, although it is difficult to illustrate it as text, as they are byte arrays. Figure 5.11 demonstrates the user interface of the encryption example:

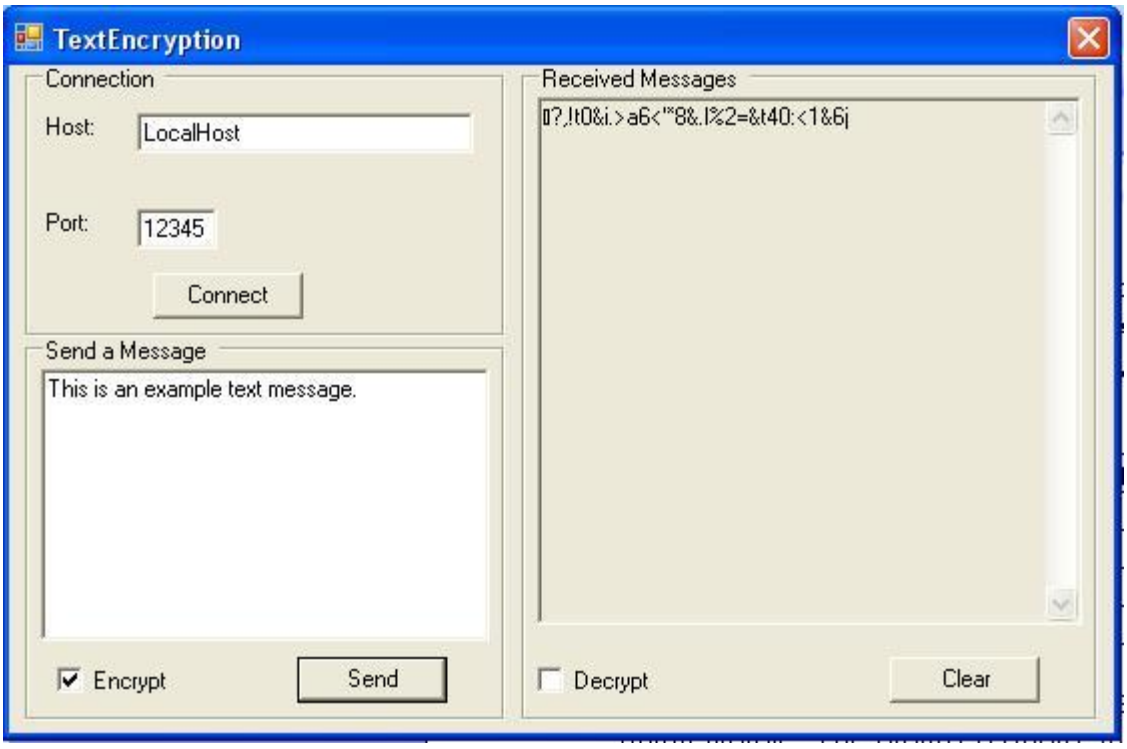

Figure 5.11 Encryption example

Keys are usually based on random number generators, which as was said earlier may influence the key strength. The randomness of the random number generator has been tested (which will be discussed below) and is given in Fig 5.12:

Value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Frequency	959	1013	989	999	1073	934	1082	1023	980	948

Figure 5.12 Random number generator

The data in Figure 5.12 was gathered from having chosen a random number between 1 and 10 for 10 000 iterations. The frequency denotes how many times the specific number (between 1 and 10) was chosen. It is evident from the data above that the numbers are fairly random. Although not precisely divided into each value, the frequency a number was randomly generated is roughly equal. Take note that the data generated differs when the application is executed again,

or a new set of data is generated. The range of possible values (1-10) should be increased radically, desirably to fit the size of the key that will be used.

5.5.2 Wireless Security

The wireless protocol's security has been evaluated by driving around in the cities of Bloemfontein, Johannesburg and Kimberley with a Senao 802.11 b/g PCMCIA with a built in Antenna, probing for networks as well as their attributes and plotting them on a map. The security levels of WEP and WPA have already been found to be insufficient [26]. Although security can be enforced (more effectively) at other layers, it is better to apply secure practice in as many layers as possible. By taking into account the small amount of administrative work involved in applying encryption to a wireless network, it is well worth it, even if it can be broken into within five minutes.

Many of the discovered nodes have default settings applied to them. Some manufacturers attempted to lighten the burden of setting up their equipment, in particular access points, for users by having defaults that work out of the box. From a support point of view, this is valuable, but from a security viewpoint, it is very bad practice. Users tend to accept defaults, and happily accept their circumstances when the device is working. Malicious attackers knowing this could easily compromise these devices. Some of the SSIDS one should look out for are the ones that are named after the manufacturer's name such as Linksys, Netgear, Gigabyte and 3com. Marconi is another frequently detected SSID in South Africa. They are (by default) routers bundled by Telkom with ADSL Internet subscriptions. Apart from most often being on default settings, they are vulnerable to other attacks. These routers are even accessible from the Internet, which means that an attacker could easily get access to many of these if an attack was launched against Telkom's whole IP range. Apparently, after some abuse, Telkom fixed the problem with their new routers and also prevented administrative access to the router from the wireless side. Although this

weakness was mostly exploited to gain additional bandwidth as Internet expenses are very high in South Africa, the problem could have been much worse as these since prime locations to insert data loggers as all traffic passes through the routers. Sensitive information such as banking details could have been acquired in the masses.

The table below depicts the total amount of wireless nodes found, the number using no encryption at all, the number which uses WEP, the number that uses WPA and the number that use WPA2 in Bloemfontein, Kimberley and Pretoria. The total number of wireless nodes found in each city does not reflect the entire population. Kimberley and Pretoria, especially have fewer nodes, as much less time was spent scanning for nodes.

	Total	**Open**	**WEP**	**WPA**	**WPA2**
Bloemfontein	326	161	127	4	34
Kimberley	105	57	43	0	5
Pretoria	141	69	58	1	13

Figure 5.13 Encryption methodologies used in the gathered data

The method and further details used to gather the above information is given below:

As mentioned, the wireless card used was a *Senao High Speed PCMCIA Adapter - b/g - 108Mbps*, which is based on the *Atheros* chipset and has good Linux device driver support (manual installation is however required for free software such as *openSuSe* as a result of licensing issues). Although the 108 Mbps is overrated, as it can only be accomplished with two cards joined together as one. Omni directional antennae generally work better than directional antennae; unfortunately the card with the built in antenna was all that was available for the tests. The November release of the *Back Track 2 Public Beta Live CD* (a CD bootable Linux distribution designed for penetration testing and

based on *Slackware Linux*) was attempted to be used on a Sony Vaio VGN-FE38GP laptop. Unfortunately, there was an issue with the built in wireless network card, and the operating system did not boot up. In order to determine the geographic location of the wireless networks and ultimately plot them on a map, a Garmin GPSmap 60 Global Positioning System (GPS) device was used. The GPSmap60 was used primarily for its ease of use and support of the NMEA protocol, which is a prerequisite for most GPS devices to communicate properly with *gpsd* (a daemon interfacing with GPS devices and making the communications available through a network socket). Unfortunately, only after the acquisition of the device did it become apparent that the NMEA protocol only operates over the device's serial cable interface, not over Universal Serial Bus (USB). Since the notebook in use neither has a serial adapter nor does the GPS device come with its serial cable packaged, much time was spent getting the device to work in conjunction with *gpsd* over the USB cable. The only solution that worked was to use *SuSe Linux 10.0* along with *kismet* and *gpsd*. Apparently it is a not-so-well-known bug that *gpsd* does not work too well with the combination of USB devices and newer kernels (the exact same packages of *gpsd* and *kisme*t where used with open *SuSe 10.2* with no success). It is said that the problem lies in the way USB devices are handled with newer kernels (/dev/ttyUSB).

Kismet is an application that has the capability of monitoring wireless networks by logging many of the wireless networks' properties and attributes such as signal strength, location, SSID, Number of clients, etc. See the *kismet* homepage (www.kismet-wireless.org) for more information. By having the wireless NIC in monitor mode, *Kismet* collects information of all the wireless networks in its presence, such as ESSID, BSSID, the channel used, whether it is cloaked or not, the type of encryption used, whether it has been decrypted yet, and some GPS information to describe each node's location. This information, along with some additional information, which will be described in detail later, is then stored in log files. These logs can be converted to a *Google Earth* compatible format by using

KNSgem. Thereafter, the nodes are displayed on the satellite photos of *Google Earth*. Unfortunately, some sections of *Google Earth's* maps are obtained from different organisations. This, along with some sort of flaw with the pattern matching software *Google Earth* uses, makes sections of the map inaccurate. For example, the southern centre landmark can be seen twice on the map. The Google Earth compatible files (*.kml*) are included on the disk. One can, after installing Google Earth, open Google Earth and simply double click the "*_Knsgem_Master.kml*" file. *Google Earth* will "*fly to*" for an overview of the Bloemfontein area and a yellow dot will be located more or less towards the centre of the screen. Thereafter, one can right-click on the dot (see Figure 5.15) and select the log (sorted by date) to be opened. The screenshot below shows the yellow dot on the map:

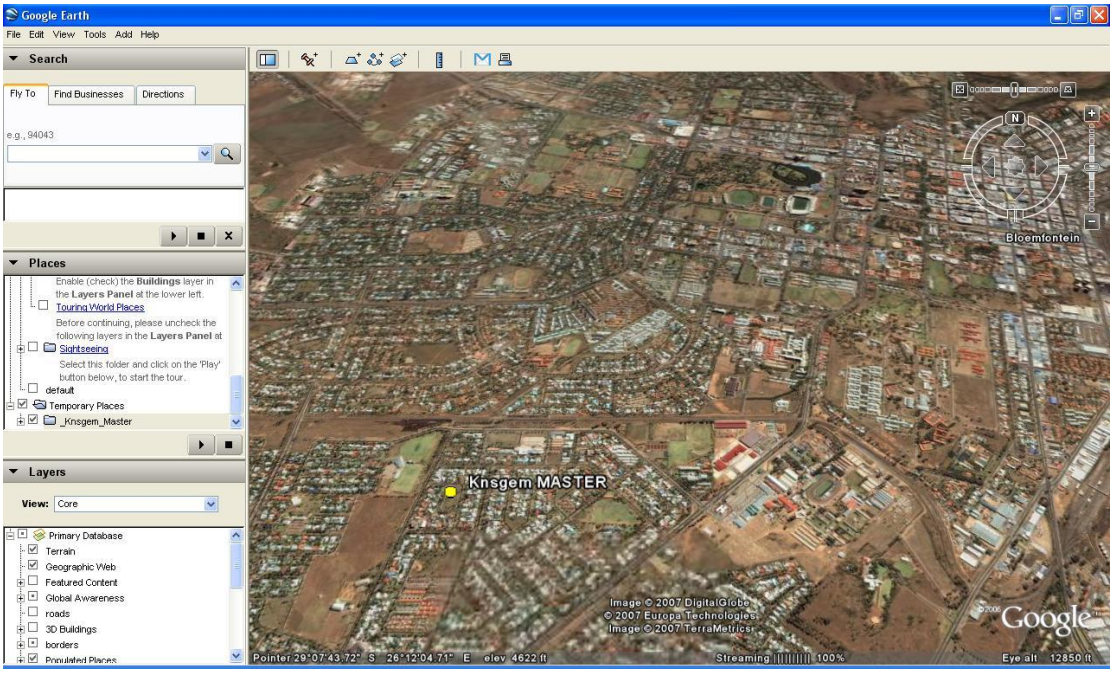

Figure 5.15 Google Earth

The image below (Figure 5.16) shows a section of the Westdene area which is rich in small businesses with some of the plotted nodes:

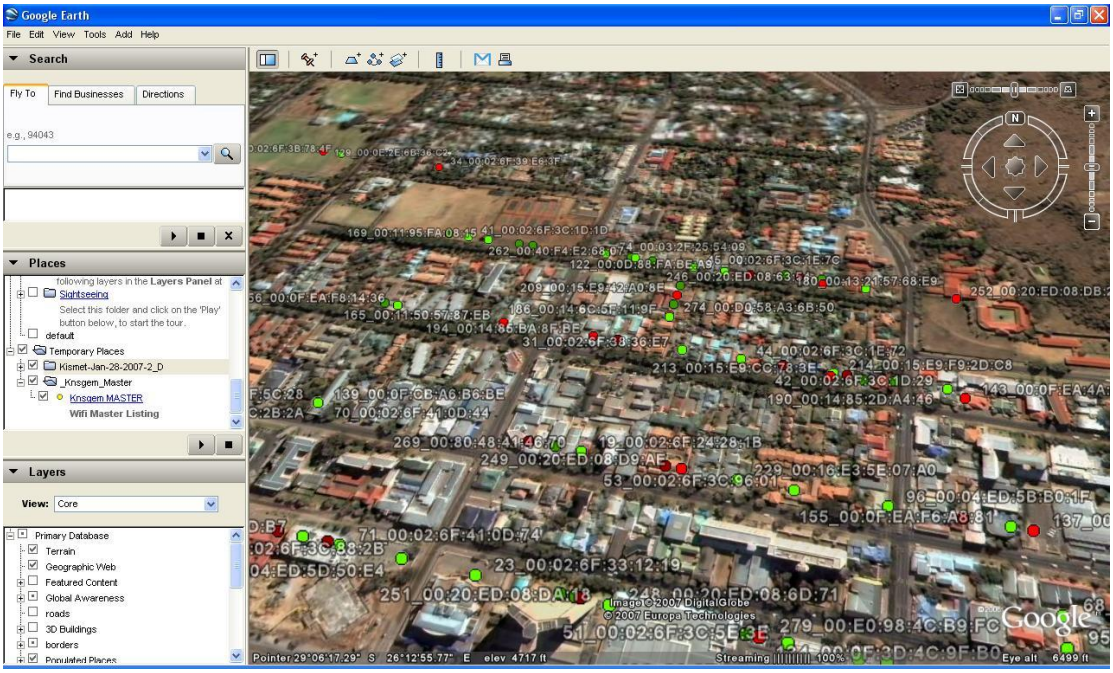

Figure 5.16 Google Earth Westdene area

It is obvious when viewing the map that there are inaccuracies. For example, some nodes are pictured within the water at Loch Logan Waterfront, and the Western curve Paul Kruger makes in Universitas is reflected by the detected nodes, but an incorrect offset (in accordance to the satellite photographs) is present. What percentages of the accuracy error are owing to the incorrect mapping or owing to flaws in the method (obstacles may influence the plotting accuracy) still remains unresolved.

The data gathered in Kimberley was done using *Network Stumbler*. *Network Stumbler* unfortunately, at this stage, does not have built in support for GPS devices over USB. Therefore it is not plotted as with the other data.

The commands are listed below:

 # iwconfig ath0 mode monitor Puts the NIC in RF-Monitor mode

 # kismet

Some APs contain NICs. They are typically composed of PC board with an open slot (in some cases several slots) for PCMCIA cards. The ones with external antenna slots have these connectors on the PCMCIA card. The operating systems of the AP as well as data settings are stored on a chip on the board. It therefore makes sense that these cards can be put in the otherwise rather strange modes such as AP mode with a tool such as *iwconfig*, which is the wireless version of *ifconfig* – a common Linux console tool. See the manual pages of *iwconfig* for more details.

The weakness of the WEP protocol lies in IV leaking out and can be deciphered by statistical mathematics. *Kismet*, *Airsnort* and *Aircrack* are the most popular tools that accomplish this. The WPA protocol can be deciphered with *coWPAtty*.

5.6 The Applications' user interfaces

The VoIP application

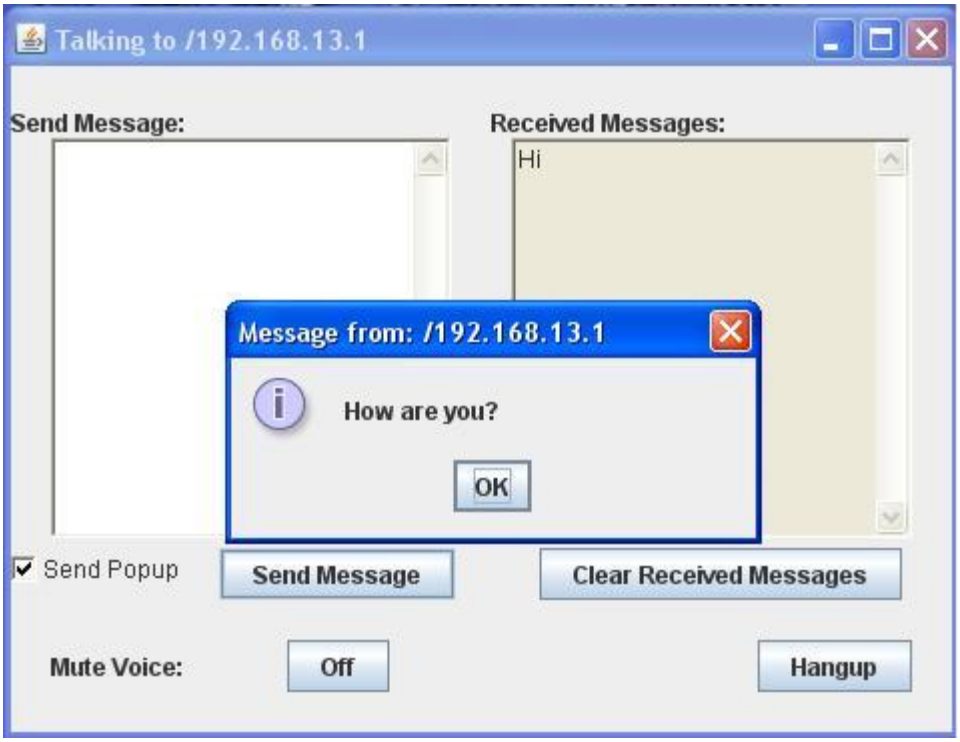

Figure 5.17 The VoIP application

Rootkit

A rootkit was also developed in C, which spawns a shell which can be used to execute commands on a remote machine. The protocol is mimicking FTP in order not to draw suspicion.

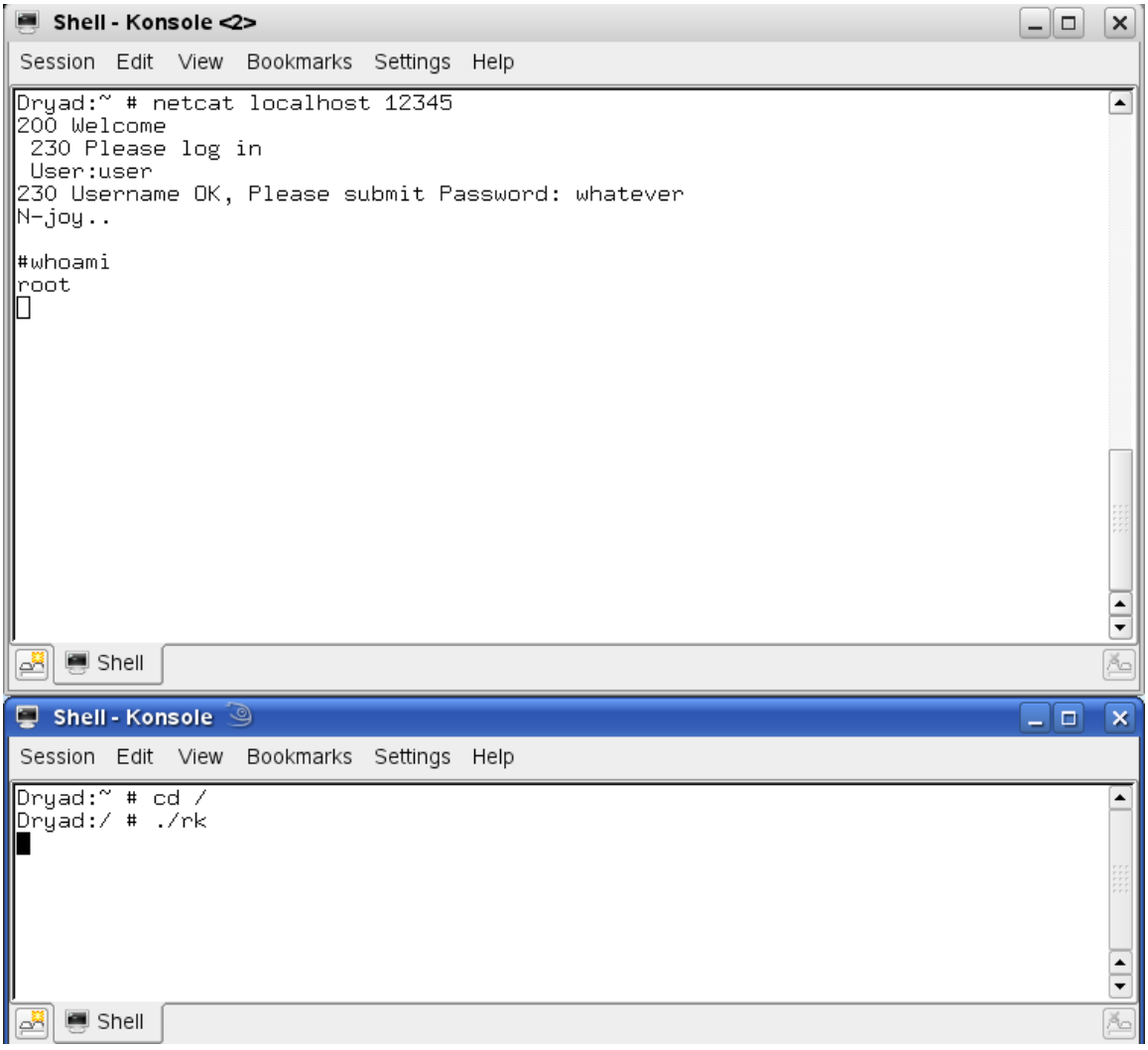

Figure 5.18 Rootkit

Trojan

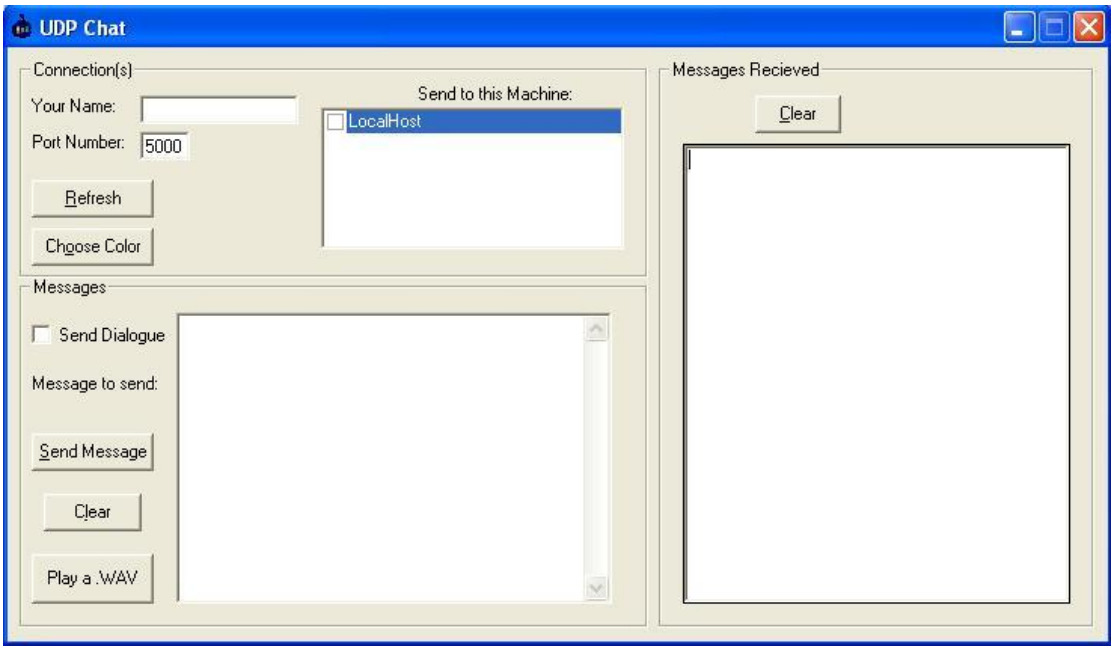

Figure 5.18 the Trojan GUI without the hidden functionality

This Trojan is a good example of how innocent an application may look to a user. The true purpose of the application can only be seen once the network traffic is analysed or the source code is reviewed (with Windows binaries this is a rather daunting task as decompiling and interpreting the assembly code is trivial).

With this application a user can strike the F12 key which brings up the following dialogue box:

Figure 5.19 The Trojan password prompt

If the correct password is entered (in this case “Password”), the user is presented with much more functionality, as can be seen on the following screenshot.

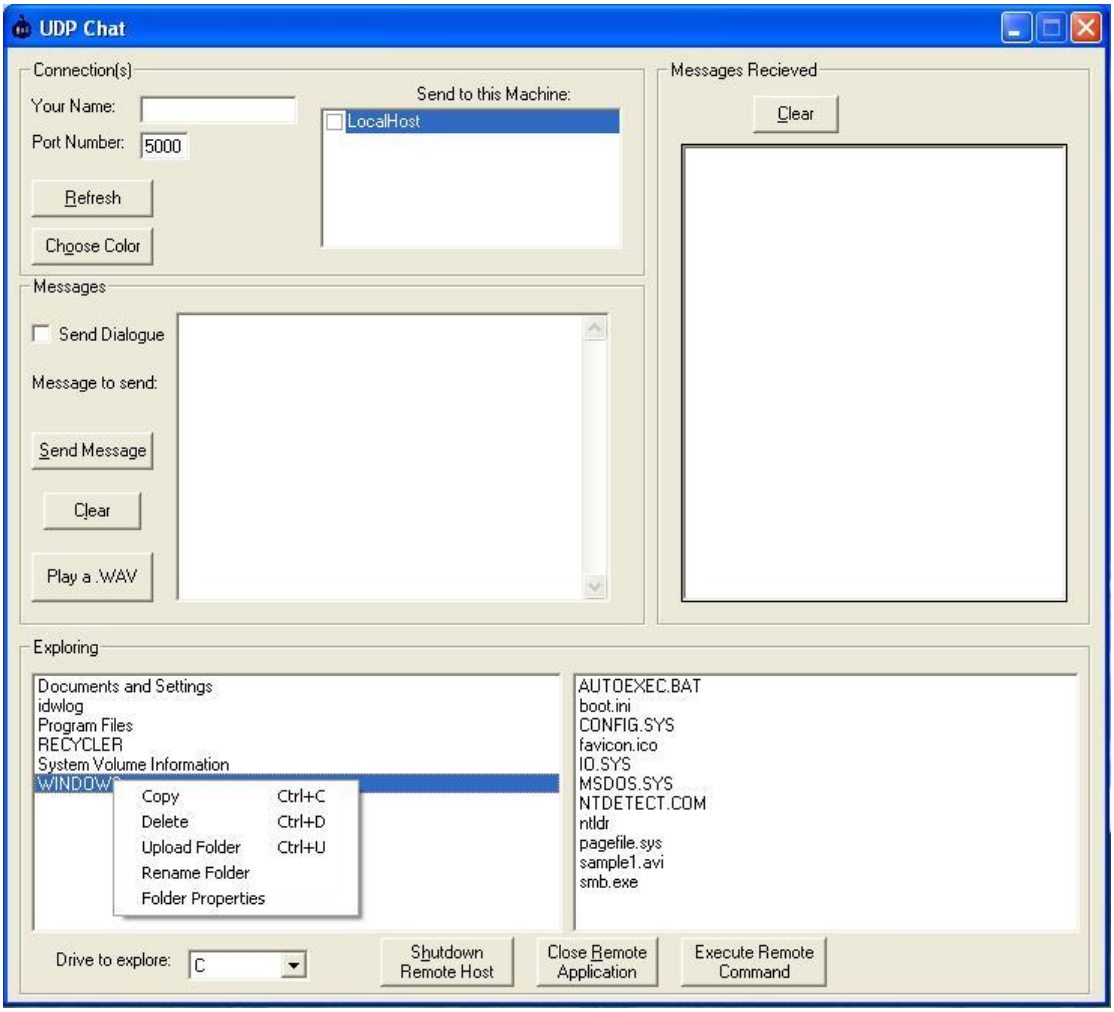

Figure 5.20 The Trojan GUI with hidden functionality

5.7 Summary

To summarise, we present the following code to serve as an example of the multitude of attack vectors that can be implemented against a few flawed lines of code. Consider the following pseudo code which is basically a possible method of FTP authentication:

```
if (left$(packetReceived,8) == "220 User") {
        if (fetchUsernameFromDB(right$(packetReceived, 8)) == true)
                correctuser = true;
        if (correctUser == true)
                sendPasswordRequired();
        else
                sendIncorrectLogin();
}
elseif (left$(packetReceived, 4) == "Pass") {
if (correctUser == true && fetchPasswordFromDB(right$(packetReceived, 4))
                == true)
        userAuthenticated();
else
        sendIncorrectLogin();
}
```

Figure 5.21 Multiple vulnerabilities example code

The preceding code examines a received packet to determine what was requested by the client. It firstly checks whether the packet was intended to identify the client with a username; if so, the database is queried with the supplied username, and the client is informed whether a correct username was supplied. If the packet was not a username submission, execution flow continues to check whether a password has been submitted instead, in which case the password is checked for in the database, and confirms if the previously supplied username was in fact a correct username. Again, the client is notified accordingly. After careful examination of the code, one might conclude that the code is vulnerable to an SQL injection (depending on how the usernames and passwords are handled in function calls). Besides that, the code introduces bad practice on username and password notification. Most modern applications are aware of this 'vulnerability' and respond with a "You supplied a valid username, please supply a password" response regardless of whether the username was

correct or not. This dramatically increases the number of tries or guesses the attacker has to make in order to receive notification that something is correct – both the username and password need to be correct in order to be sure a username is in fact registered in the system (valid).

An obvious vulnerability that was missed with the code above as well is that the username and password were not treated as a combination. The logic is flawed since any correct username, as well as any correct password will result in successful authentication. The username does not require its own matching password. This also drastically decreases the number of possibilities (especially in the case where there are many usernames and passwords in the database) to the benefit of a brute force attack.

If the code was adjusted to reply positively to the client on a username submission, regardless of whether it is in fact valid, closer examination of the code reveals yet another possible weakness. An attacker may code an application that understands the authentication process of the protocol along with some added functionality. Additional to authentication the application maintains a timer and a log of measured response times from the server. Every time the attacker guesses an incorrect password the response time may vary minimally. Conversely, if the attacker guesses a correct username, the amount of time again may vary, but on average (of a few attempts) correct usernames are responded to in a longer time segment. This happens because an extra instruction is executed (line 3), along with passing some database parameters and so forth. Resource availability may drastically influence and most probably distort results of the attacker, but on the other hand the attacker might be aware that the system load is very low at a specific time of day, for example, and launch his attack accordingly.

From the vulnerabilities discussed above, it is apparent that a vast number of things could go wrong. Developers need to be fully aware of what their code really does, and forget what they think it does, or what they want it to do.

6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Both VoIP and wireless networking are relatively new technologies. All new technologies tend to take time before they are implemented to satisfactory standards. Furthermore, security's worst enemy is additional features. These two factors have drastically detrimental trade-offs. Probably the largest problem with wireless networking is the feature that it should be able to support re-association of a client and a different access point in the network. The problems exponentially escalate as soon as a work-around is devised. It therefore makes sense that different standards are implemented, that is 802.11 and 802.16, as different problems are attempted to be solved. The 802.11 standard has a stronger focus on mobility (disassociation and re-association) whereas 802.16 does not attempt this and therefore accommodates a stronger focus on security.

The most common security flaws for VoIP over wireless networks were examined along with viable solutions. The objectives of this work were mainly to describe the details of what cause the vulnerabilities, and how they can be solved, as awareness is the key to developing a secure system. The vulnerabilities within the scope of this work were fully explained by means of what the causes are, what effect exploitation has, as well as how they can be mitigated.

A difficulty that was experienced is that all of these vulnerabilities may appear in almost any section of the code of the entire programme. Since this opens a myriad of new conditions that should be compensated for, the code may grow more than two-fold for a working, secure VoIP system. As larger pieces of code opens even more possibilities of oversight errors, securing VoIP systems seems an unattainable feat.

Security is often approached on the basis of the systems development life cycle. The reason for this is that security is never achieved; it rather is a continuous process of risk analysis (which constantly changes along with the company's internal and external environment), mitigation and acceptance. It is both senseless and unprofitable for a business to mitigate a risk which realistically would never come to pass. On the other hand financial and military institutions need excellent security as they are much more prone to attacks.

The worst problem with security is that the vulnerabilities arise from oversight errors. This is the main reason why we addressed each problem separately, in order to completely explain and ensure in depth knowledge of some of the problems that security professionals face. Implicitly, this approach also revealed some weaknesses. The in depth study of the vulnerabilities imposed time restrictions on developing a complete, secure system on which tests could be done. There are also many other types of vulnerabilities that have not been studied. Developers should be encouraged to take considerable conditions into account.

In 2003 when WEP was proven beyond doubt to be "crackable" the Internet Engineering Task Force made it known that the next standard will be much more secure, and that WEP was not intended to be invincible. As the name states: Wired Equivalent Privacy, wired networks do not have much privacy, except for physical access restriction. A large factor influencing the successful maintenance of security is training. As Kevin Mutnick said, the user is the weakest link in security [11].

6.2 Future Work

Future work may include studying the other common vulnerabilities, field tests on new wireless technologies, such as WiMAX and UMTS/WCDMA. A working system could be developed and tested for vulnerabilities. There are also code scanners available that scan code for known vulnerabilities before compiling, which could be used to aid in developing the system.

Furthermore, a system could be developed that allows for calls to be made over a NAT enabled network. This would require a server on the network that is reachable by all hosts and directs all calls. However, great care should be taken to guard against DoS attacks in such a setup and in-depth research should be conducted on the matter.

6 Gevolgtrekking en verdere navorsing

6.1 Gevolgtrekking

Beide VoIP en koordlose netwerke is relatief nuwe tegnologieë. Alle nuwe tegnologieë neig om tyd op te neem alvorens dit volgens bevredigend standaarde geïmplementeer kan word. Verdermeer, is sekuriteit se grootste vyand is addisionele funksionaliteite. Hierdie twee faktore beïnvloed mekaar drasties. Die probleem eskaleer eksponensieel sodra omleidings ontwikkel word. Dit is gevolglik sinvol dat verskillende standaarde geïmplementeer word byvoorbeeld 802.11 en 802.16, in ’n poging om verskillende probleme op te los. Die 802.11 standaard het ‘n sterker fokus op mobiliteit dissosiasie en herassosiasie terwyl 802.16 nie so sterk op mobiliteit afgestem is nie. Gevolglik bied dit beter sekuriteit.

Die mees algemene sekuriteitsleemtes vir VoIP via koordlose netwerke is geëvalueer met lewensvatbare oplossings. Gesien in die lig dat waaksaamheid die sleutel is tot die ontwikkling van ’n sekure sisteem, was dit die doelwit van hierdie studie om in hoofsaak die eienskappe te beskryf wat verantwoordelik is vir kwesbaarheid en hoe om dit op te los. Die kwesbaarhede binne die reikwydte van hierdie studie was volledig verduidelik: Die oorsake, die impak van uitbuiting en metodes om dit te verlig.

‘n Dilemma wat ondervind was, is die feit dat al die genoemde kwesbaarhede in bykans enige afdeling van die kode van die algehele program kan voorkom. Aangesien dit ‘n veelheid van nuwe geleenthede meebring waarvoor daar gekompenseer moet word, vergroot die kode meer as dubbelvoudig vir die werking van ’n sekure VoIP sisteem. Aangesien bonkige kodes selfs meer moontlikhede vir oorsig-foute meebring, blyk die opskerping van sekuriteit vir VoIP sisteme onhaalbaar.

Sekuriteit word dikwels gebaseer op die sisteme lewenssiklus. Die rede hiervoor is dat sekuriteit nie ’n afgekapselde eindpunt bereik nie. Dit is eerder ’n konstante proses van risiko-analise (vanweë die konstante veradering van die maatskapy se interne en eksterne milieu). Dit is beide sinloos en ook onekonomies vir ’n besigheid om risiko’s wat realisties gesproke nooit sal manifesteer nie, te probeer bekamp. Aan die anderkant benodig finansiële en militêre instansies wel uitstekende sekuriteit gesien in die lig dat hulle groter teikens vir infiltrasie is.

Die grootste probleem met sekuriteit is dat swakplekke as gevolg van oorsig-foute ontstaan. Dit is die hoof-rede waarom hierdie studie elke probleem afsonderlik aanspreek, ten einde elk breedvoerig te verduidelik en indiepte kennis te verseker van sommige van die probleme waaraan sekuriteitslui aandag moet skenk. Implisiet het hierdie benadering ook sommige leemtes uitgewys. Die indiepte studie van die swakhede het ’n tydsbeperking geplaas op die ontwikkeling van ’n volledige sekure sisteem, waarop toetse gedoen kon word. Daar is ook baie ander tipes kwesbaarhede wat nie bestudeer is nie. Ontwikkellaars moet aangemoedig word om ‘n substansiële hoeveelheid omstandighede in berekening te bring.

In 2003 is bo alle twyfel bewys dat WEP indringbaar is. Internet ingenieurs het bekend gemaak dat die volgende standaard baie meer sekuur sal wees, en dat WEP nie daarop toegespits was om ondeurdringbaar te wees nie. Soos die term “Wireless Equivalency Privacy” aandui, het bedrade netwerke nie veel konfidensialiteit nie behalwe vir die fisiese beperking wat daar op toegang gestel word.

‘n Belangrike faktor wat die suksekvolle instandhouding van sekuriteit beïnvloed, is opleiding. Kevin Mutnick het gesê dat die gebruiker die swakste skakel in sekuriteit is. [11]

6.2 Toekomstige Navorsing

Toekomsnavorsing kan gedoen word oor ander algemene kwesbaarhede en veld-navorsing oor nuwe koordlose tegnologieë, soos WiMax en UMTS/WCDMA. ‘n Werkende sisteem kan ontwikkel en getoets word vir kwesbaarhede. Daar is ook kode-skandeerders beskikbaar vir die voorafskandering vir kwesbaarhede alvorens inbruikneming van ‘n sisteem.

‘n Sisteem kan ook ontwikkel word wat oproepe oor ’n NAT netwerk kan maak. Dit vra om ‘n bediener op die netwerk wat vir alle kliënte bereikbaar is en alle oproepe kan herlei / kanaliseer. Dit moet egter streng gewaak word teen DoS aanvalle in sodanige opstelling en indiepte-navorsing moet in die verband onderneem word.

7 References

1. http://www.jetcityorange.com/CapnCrunchWhistle/ (Last accessed 22 September 2008).
2. http://www.realvnc.com (Last Accessed 22 September 2008)
3. http://records.viu.ca/~soules/media112/hacker.htm (Last accessed 22 September 2008).
4. S.H. Park, A. Gins, Z. Gins, "Robust Re-authentication and Key Exchange Protocol for IEEE 802.11 Wireless LANs", IEEE Military Communications Conference, Boston, MA, October 1998.
5. R. A. Serway, "Physics for scientists and engineers with modern physics", Fourth Edition, 1996.
6. http://insecure.org/nmap (Last accessed 10 August 2007).
7. http://www.whois.net (Last accessed 08 August 2007).
8. http://www.invisiblethings.org/papers/chameleon_concepts.pdf (Last accessed 2 March 2007).
9. http://en.wikipedia.org/wiki/Virtual_address_space. (Last accessed 07 March 2007).
10. http://www.hoobie.net/brutus/. (Last accessed 07 August 2007).
11. Kevin D. Mitnick, William L. Simon, "The Art of Deception", John Wiley And Sons Ltd, 2002.
12. http://www.linuxcommand.org/man_pages/iwconfig8.html (Last accessed 22 September 2008).
13. Andrew S. Tanenbaum, "Computer Networks", Fourth Edition, 2004.
14. Arbaugh, W. A., Shankar, N., and Justin Wan, Y.C., "Your 802.11 Wireless Network has No Clothes" Available on http://www.drizzle.com/%7Eaboba/IEEE/wireless.pdf (Last accessed 3rd March 2007).
15. http://jdic.dev.java.net (Last accessed 12 March 2007).
16. RFC 4123 http://www.rfc-archive.org/getrfc.php?rfc=4123 (Last accessed 22 September 2008).

17. Ari Takanen, TO Codenomicon Ltd., “VoIP Security Threats:Rebuilding Trust in Communication Networks”, 2006

18. Anand Balachandran, “Wireless Hotspots: Current Challenges and Future Directions”, 2003.

19. Ashutosh Dutta, Tao Zhang, Sunil Madhani, “Secure Universal Mobility for Wireless Internet”, 2004.

20. William Stallings, “Operating Systems - Internals and Design Principles”, Fifth Edition, 2005.

21. http://airsnort.shmoo.com/ (Last accessed 22 September 2008).

22. http://linux.die.net/man/1/aircrack-ng (Last accessed 22 September 2008).

23. http://secguru.com/link/how_write_rootkit. (Last accessed 08 August 2007).

24. http://www.lightreading.com/document.asp?doc_id=139772&page_number=9 (Last Accessed 22 September 2008).

25. http://anml.iu.edu/ddos/types.html (Last Accessed 22 September 2008).

26. http://www.drizzle.com/~aboba/IEEE/ (Last Accessed 22 September 2008).

8 Appendices

Appendix A

```
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

void function() {
    char buffer1[5];
    char buffer2[10];
    int *ret;

    ret = &buffer1[5];
    ret += 2;
    (*ret) = (int)&shellcode;
    printf("buffer1 = %x \nret = %x\nbuffer1[4] = %x\n", buffer1, ret,&buffer1[4]);

}

void main() {
    function();
}
```

Appendix B

```
Private Sub chkSQL_Click()
    If chkSQL.Value Then
            lblSQL.Visible = True
            Call calcSQL
    Else
            lblSQL.Visible = False
    End If

End Sub

Private Sub cmdLogin_Click()
    On Error Resume Next
    Dim strSQL As String
    Dim conDB  As ADODB.Connection
    Dim rsUser As New ADODB.Recordset

    Set conDB = New ADODB.Connection
    Set rsUser = New ADODB.Recordset

    conDB.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\UsersDB.mdb;Persist Security Info=False"
    conDB.Open
    strSQL = "SELECT * FROM tblUsers WHERE Username = '" & txtUser.Text & _
    "' AND UserPassword = '" & txtPassword.Text & "';"

    With rsUser
        .CursorLocation = adUseClient
        .Open strSQL, conDB, adOpenDynamic, adLockOptimistic ', adCmdText
```

```
        If .EOF And .BOF Then
            MsgBox "Wrong username & password"
        Else
            MsgBox "Correct"
        End If
    End With

End Sub

Private Sub txtPassword_Change()
    Call calcSQL
End Sub

Private Sub txtUser_Change()
    Call calcSQL
End Sub

Private Sub calcSQL()
    If chkSQL.Value Then
        lblSQL.Caption = "SELECT * FROM tblUsers WHERE Username = '" &
txtUser.Text & _
    "' AND UserPassword = '" & txtPassword.Text & "';"
    End If
End Sub
```