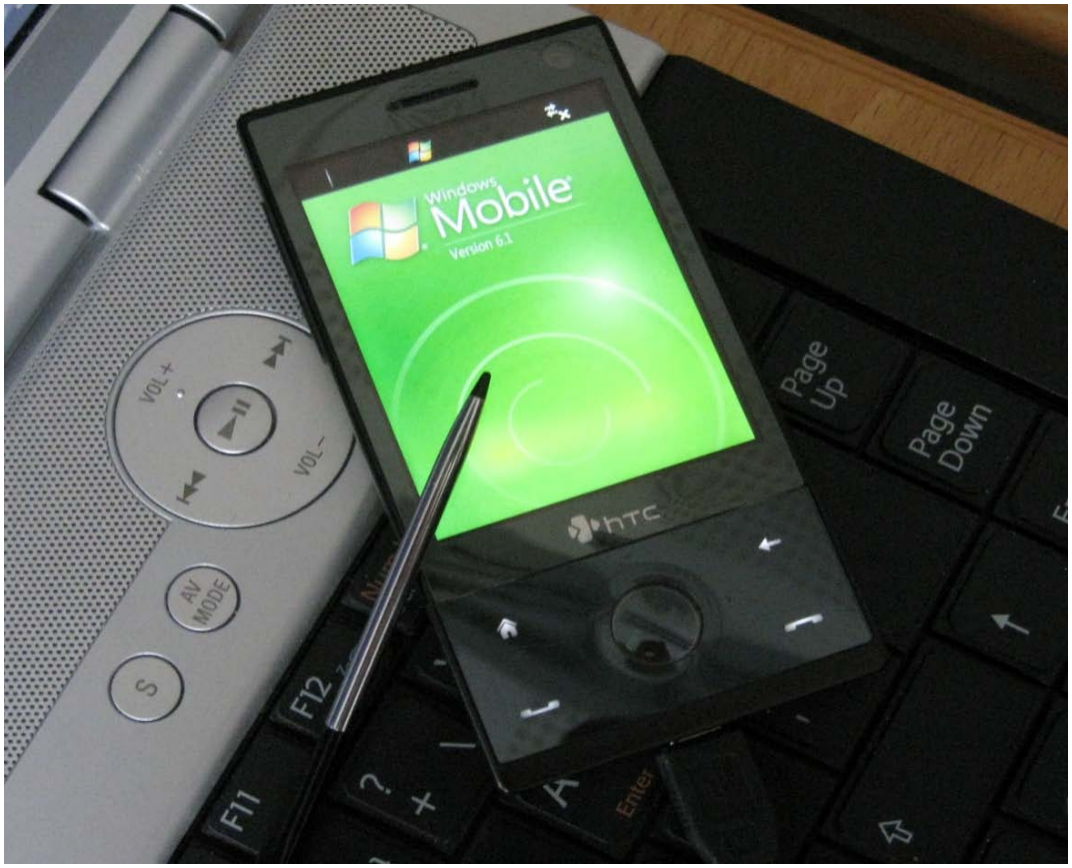
CHALMERS



Security Evaluation of the Windows Mobile Operating System

Master of Science Thesis

Sheikh Mahbub Habib
Syed Zubair

Chalmers University of Technology
Division of Networks and Systems
Department of Computer Science and Engineering
Göteborg, Sweden, July 2009

Security Evaluation of the Windows Mobile OS

SHEIKH M. HABIB, SYED ZUBAIR

Examiner: Dr. TOMAS OLOVSSON

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden July 2009

TO MY WIFE AND PARENTS
- SHEIKH MAHBUB HABUB

TO MY PARENTS AND FRIENDS
- SYED ZUBAIR

ACKNOWLEDGEMENT

We show significant and indescribable gratefulness to our supervisor, Associate Professor Tomas Olovsson for his helpful contribution in giving encouragement, suggestions and guiding in the right direction throughout the research work. During the thesis work, he has been a great source of effective and feasible ideas, profound knowledge and all-time feedback to us.

We are very grateful to Martin G.H. Gustavsson, Product Manager/Senior Advisor, National Security & Public Safety of Ericsson AB (Mölndal, Sweden) for supporting our Master's Project. This thesis could not have been completed without their help.

We are thankful to the Microsoft Corporation for publishing the documentation of Windows Mobile/CE operating system in MSDN. From the documentation, we got a thorough idea about the architecture of mobile/embedded operating systems.

We also want to thank our university for giving all kinds of support throughout our Masters studies.

Finally, we want to give heartiest thanks to our family and friends for inspiring us through out the thesis work.

Security Evaluation of the Windows Mobile OS

ABSTRACT

Smartphones are rapidly replacing the traditional cellular phones because of its PC-like features which allow them to be used as web browsers, editors and media players. These devices run complete operating systems that allow installation of third party applications. As their usage grows, the need for securing the user data stored on these devices and the services that they provide becomes more and more vital. Mobile Operating Systems naturally become the target for security scrutiny as they are responsible for managing all the information and services.

Our report analyzes the Windows Mobile 6.1 operating system and its security features. We have looked into the OS architecture, memory management, process management, file system, device drivers and communication service and network stack architecture. We have analyzed its security model and looked into some of the related research work regarding the operating system's security. Moreover, penetration testing against the Windows Mobile 6.1 operating system has been performed to show the stability and robustness of the network stack in a hostile environment. At last, security issues are discussed in detail from an application's perspective in order to show what type of (security) problems are faced by an application and what improvements are needed to counter those threats. In a whole, this report identifies the strengths and weaknesses of the Windows Mobile operating system. The aim of this report is to create awareness in this area and to encourage further research into mobile operating system security.

Keywords: Smartphones, Mobile Operating Systems, Windows Mobile, Security, Penetration testing.

TABLE OF CONTENTS

Chapter 1
Introduction

Security is a major concern in the area of information and communication technology while dealing with computers. The smartphone is a new addition to the family of mobile devices that combines the functionality of mobile phones and Personal Digital Assistants (PDAs). Quoting Dan Hoffman, CTO of SMobile Systems in Columbus, Ohio:

"Smartphone are becoming the new laptop"

This extends our definition of Smartphone. Smartphones are "smart" because apart from handling voice calls, these devices let the customers browse the web using GSM/3G and Wi-Fi connections, install third party applications for reading documents, reading emails etc. Bluetooth and Infra-red are other network services which are common in almost every smartphone nowadays. Mobile workers are adding smartphones to their daily life because these devices offer extra functionality and services. They are small enough to be carried in pockets, yet still able to link to the enterprise. Linking the enterprise through smartphones not only exposes the device itself to threat but also the company where the mobile worker works. Even in crisis management scenarios, smartphones are used, for example where voice and data communication are the only means to communicate between rescue workers. Smartphones are being used in environments where Confidentiality and Integrity is of utmost importance. All these deployments indicate that these devices are given a lot of responsibility. Since the application of smartphone is rapidly growing it is natural that people start asking "Are Smartphones secure?". In order to be able to examine the security of smartphones we target this question at three important levels:

- **Operating System**: The operating system is the obvious candidate for probing because it runs the smartphone and manages all its resources like files, devices and memory. In this report, we have investigated what level of security a smartphone OS offers, what kind of kernel architecture is used, how is the memory managed, how the processes are organized and what kind of file system is used.
- **Network Stack:** Since strong connectivity is one of the most important features of smartphones, it is important to examine if smartphones have a robust and stable network stack in order to counter different attacks, particularly the ones that have targetted desktop operating systems.
- **Applications:** The ability of installing new applications is one of the hot features of smartphones. We have looked into the threats to applications and vulnerabilities introduced by them.

In this thesis paper, we will provide a detailed discussion and highlight related work to answer these questions.

1.1 Background

The first smartphone Simon [3] which was a joint venture of IBM and Bellsouth in 1993 had some significant features such as touch screen technology, personal information management features (e.g. calendar, address book, world clock, calculator, notepad, e-mail, send and receive fax and games). After that, in the next 15 years the smartphone has been changed significantly in terms of operating system architecture, processing power and services offered in these devices. Users of mobile devices have a craving for new services, they want rich Internet access despite channel constraints. That's why most of the smartphones are using IEEE 802.11 Wi-Fi interfaces to give the user the taste of high speed internet. By using the WLAN feature, the Smartphone becomes a member of an IP-based network like a desktop or notebook and it may be accessible for users all over the Internet. In addition, because of vulnerabilities in the link layer and above, for example when using WEP encryption [1], attackers may be able to spawn link-level attacks via wireless access point. Also, open mobile OS platforms running in these devices let the users install third party applications where the Smartphone is used as man-in-the-middle bridging two networks together, a scenario which was shown in enterprise networks using Blackberry devices [2]. Wireless and Telecom analyst Jeff Kagan (2008) expressed his concern over the above issues:

> *"People are putting everything on them (phones). Surprisingly, we haven't had major security events resulting from these wireless devices yet. But we have to think that they are coming. The people who mount security attacks are bound to focus on smartphones at some point."*

Even though we can see that smartphones are equipped with lots of services similar to desktop/notebook computers, security is still neglected as most of the attention is given to the development and quick release of new functionality and services. This is a major reason why security evaluations of mobile operating systems in terms of applications, services and network stack are needed.

1.2 Problem Statement

For the issue of smartphone security, we would like to focus on the architecture of Windows Mobile operating system that runs on devices like smartphones, PDAs etc. The architecture includes the kernel and memory architecture, file system, device driver management and communication services and the network stack. Moreover, we have explored Windows Mobile network stack vulnerabilities. It is interesting to see how secure an application, deployed/installed in this platform, can be.

Security of mobile operating systems was initially investigated by Jukka Ahonen [4] and Arto Kettula [5] in the year 2001 and 2000 respectively. These studies are based on older platforms but the studies show that most of the mobile operating systems lack of important features like permission based file access control, multi-user support and even memory protection. S. Perelson and R.A. Botha [6] have done a thorough investigation regarding access control of mobile devices in terms of different OS platforms.

Among the recent researchers in the field, Collin R. Mulliner [7] is one of the most active researchers in the area of Smartphone security irrespective of any platforms. In his Masters thesis [8], he has given a detail investigation of security issues in different OS platforms that are running in smartphones in recent times. Also, two security problems were identified related to the increased capabilities of smartphones. The first problem was related to the integration of multiple wireless interfaces into a single device. He showed how cross service attack exploits vulnerability through one interface (WLAN) and then ends up controlling another interface (GSM). Prevention mechanisms of this type of attack have been proposed in this thesis as well. The second problem was regarding software components or applications running on smartphones. The client part of the MMS user agent was compromised through a buffer overflow vulnerability to generate an MMS-based worm. It is known to be the first mobile phone-related code-injection attack. Besides this paper, a practical analysis of Windows Mobile 5.0 and Symbian 9.1 (Nokia, SonyEricsson) operating systems has been done in [9] and [10] respectively. In those papers, focus was given to analyze the network stack's robustness and stability of smartphones while running in real-world scenario.

Most of the papers published earlier are concerned with Windows CE 4.1, 4.2 or 5.0 based mobile OS or earlier versions of the Symbian operating system. In this thesis, our focus is on Windows Mobile 6.1 which is based on modified Windows CE 5.0 architecture. The idea is to describe the OS architecture and its security issues along with some practical analysis and examples.

1.3 Objectives

The major objectives of this thesis are as follows.

- ⇒ To get a thorough idea about the architecture of Windows Mobile operating systems including the application execution on these platforms.
- ⇒ To explore the security model of Windows Mobile platforms.
- ⇒ To study and analyze security issues related to mobile operating systems published by researchers in this area.
- ⇒ To analyze the robustness and stability of the network stack in the Windows Mobile 6.1 OS.
- ⇒ To enumerate security problems from an application's perspective.

1.4 Scope of the problem

In our thesis we have analyzed the architecture of the Windows Mobile OS which is one of the commonly used operating systems by smartphone manufacturers today. A wide range of literature was studied in the area of smartphone security which includes application vulnerabilities through third party applications, malware, multimedia services, and I/O device vulnerabilities through interception of calls while using VOIP applications. Also, the network stack of Windows Mobile 6.1 operating system is tested in order to check the robustness and stability of smartphones while used in open or hostile environments. In our research, third party security products will not be

taken into consideration. Our focus will be on the built in security mechanisms like access control, security policies, crypto libraries, memory protection, certificates and code signing in the Windows Mobile OS.

1.5 Organization of thesis

Chapter 2 describes the architecture of Windows Mobile 6.1 operating systems. The chapter also includes an overview of Windows CE and Windows Mobile OS. Some general security implications related to OS architecture are mentioned here as well.

Chapter 3 presents the security model used by Windows Mobile operating systems. The chapter includes access permissions, certificates, security policies, security roles, security services and users' influence on security.

Chapter 4 illustrates the security issues and some protections mechanisms pointed out by different researchers

Chapter 5 depicts a practical analysis of Windows Mobile 6.1 operating system's network stack.

Chapter 6 discusses the security issues related to a Mobile VoIP application. This chapter also suggests some protective measures to be taken while running this type of application in the mobile operating systems.

Chapter 7 states the conclusive words and plans for the future research.

Chapter 2
Mobile Operating Systems

An operating system is a piece of software that controls the hardware and with which applications interact in order to carry out functions. Mobile OSes are used in different types of mobile devices like PDAs (Personal Digital Assistants), Smartphones, Tablet PCs, Mobile Media Players, Mobile Gaming Devices, Mobile Phones and Industrial Mobile Devices. In this thesis, we will focus on the operating system that runs in smartphones. Although the smartphones are capable of using most of the functionality that is present in desktops, laptops and notebooks, operating systems used in this type of devices are different from those used in personal computers. The main reason for the difference is the energy and space constraint associated with special hardware and special requirements. For example, a mobile OS may require a hardware wake-up function to develop features like appointment reminders which can occur during the "sleep" mode to save energy in the device [8].

Other differences are related to the system applications rather than the OS kernel. The differences in the system software arose from distinct UI (user interface) requirements of such devices, like introducing the thumbwheel, the lack of full keyboard, the small display size and also the sophisticated touch-screen technology. An overview of embedded operating system platforms (Windows CE and Windows Mobile) will be presented to help the reader distinguish between them with respect to their functionality. Also, the OS architecture of Windows Mobile 6.1 is discussed in the next few sections.

2.1
Windows Embedded OS Platforms

Windows Embedded is a collection of operating systems and platform builder tools to build embedded devices with the rich set of componentized technologies. Microsoft has a lot of products under the Windows Embedded umbrella which are Windows Embedded CE [11], Windows Embedded NavReady [12], Windows Embedded Standard [13], Windows NT Embedded [14], Windows Embedded for Point of Service [15] and .NET Micro Framework [16]. In the next two sections, Windows Embedded CE and Windows Mobile platforms will be discussed, as the main objectives of the thesis is to evaluate the security issues concerned with the smartphone OS.

2.1.1
Windows CE

Windows CE (Compact Edition) is an embedded 32-bit operating system that is used to build a broad range of intelligent devices. With Windows CE, it is possible to develop robots, industrial controllers, gas station pumps, communication hubs, point-of-sale terminals, voting machines, medical equipments, digital music players, interactive television, cameras, mobile phones, video games, etc. It is an open, scalable and real-time operating system. From the latest release of Windows CE, Microsoft has changed the previous name of Windows CE 5.0 to Windows Embedded

CE 6.0. The other versions are Windows CE .NET, Windows CE 3.0 and Windows CE 2.1. The WinCE operating system comes with a platform builder which is used to configure, build, deploy and debug a new OS based on selected board support packages (BSPs)[17] and the device type (e.g. Mobile handheld, Gateway, Industrial controller, etc). It is done by using selected components like .NET compact framework or Windows Media and the network services (e.g. LAN or WAN or PAN) needed in the target device. Even, an SDK can be developed using the CE platform builder to further extend the target OS platform. Currently, four types of BSPs or processor architectures are supported which are ARM [18], MIPS [19], SHx [20] and x86 [21]. Based on processor architecture, the platform builder generates the OS and deploys it in the hardware which will have a bootloader itself. In this way, it is possible to build an industrial robot based on Windows CE, may be that it will not have any windows-like interface or any display interface but can have its own components to function properly.

2.1.2 Windows Mobile

The Windows CE platform builder kits are distributed among the different embedded developer audience and to internal teams of Microsoft and includes Windows Mobile (Standard, Classic, Professional), Windows Automotive, MSNTV, etc. Then, the responsible team chooses the components needed for Windows Mobile, adds a Windows-like shell, a bunch of applications (e.g. Office Mobile, IE Mobile, etc.), radio interface layers for connected devices, APIs and platform extensions to generate the SDK (software development kit). This is the point where Windows Mobile differs from Windows CE. WinCE comes as a full OS to the manufacturers but Windows Mobile comes like an SDK. The SDK is built for the Original Equipment Manufacturers (OEMs) who integrate it in their mobile devices, based on the hardware specific BSPs and ship the Windows Mobile based devices.

There are three editions of Windows Mobile 5.0: PocketPC, PocketPC Phone and Smartphone. Windows Mobile 6 also has three editions: Windows Mobile Standard, Classic and Professional.

2.2 Windows Mobile 6.1 OS Architecture

Microsoft Windows Mobile 6 [22], based on Windows CE 5.0 [23], is used in various third party devices such as Smartphones and Personal Digital Assistants (PDAs). It was released on Feb 12, 2007 in three different versions: Windows Mobile 6 Classic (for Pocket PCs without cellular radios), Windows Mobile Standard (for Pocket PCs without touch screens) and Windows Mobile Professional (for Pocket PCs with phone functionality). Windows Mobile 6.1 was a minor upgrade of the existing Windows Mobile 6.0 platform which is based on a modified Windows CE 5.0 kernel not the newer Windows Embedded CE 6.0 [24] kernel.

In the next few sections, the major parts of the OS architecture will be discussed that are relevant for the security evaluation of mobile operating systems. The major parts of the mobile OS architecture are the kernel, memory management, file systems, device drivers and communication services and network stack.

2.2.1
Core OS architecture

Windows Mobile 6.1 is a 32-bit operating system based on a modified Windows CE 5.0 (version 5.2) kernel [25]. Nk.exe is the module which represents Windows CE kernel in all CE based devices. The kernel provides basic OS functionality like process, thread and memory management. Using the kernel process and thread functions, processes and threads are created, terminated and synchronized. They are also used to schedule and suspend a thread. Thread priority levels, priority inversion with inheritance, time and scheduling are all included in the WinCE kernel architecture. In this way, real-time application capabilities are ensured in Windows CE which is very important for time-critical systems.

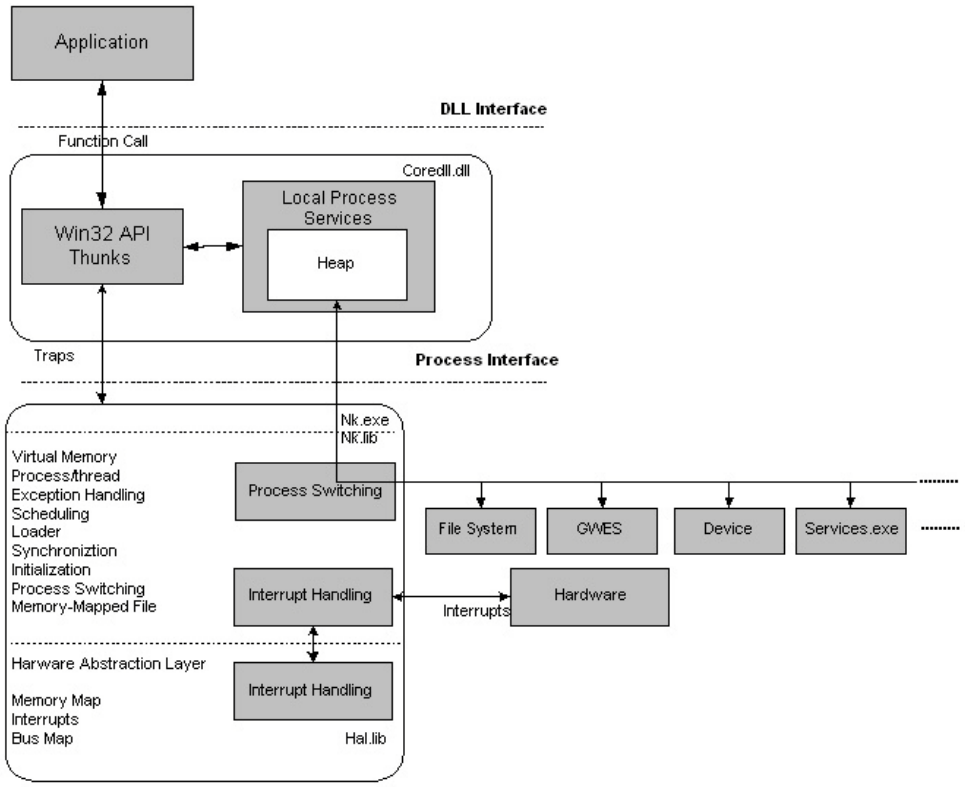


Figure 2.1: General architecture of Core OS emphasizing Kernel.

Interrupt handling and some file management functions are also provided by the kernel. In fig. 2.1, the general architecture of the Core OS is shown together with kernel functionality related to application execution in Windows CE based devices. Here, the core OS can be divided into three parts: the user interface, DLL interface which has Coredll.dll and process interface which consists of kernel handler Nk.exe, interrupt handler and other executable handlers for file management, I/O device interaction and graphical user interface management. Suppose an application makes a function call to Win32 API via the DLL interface to use a particular function. If the application has memory requirements less than 64 KB, the application can use the local heap or create separate heaps. Otherwise, applications request to use virtual memory using kernel memory functions. Kernel memory functions also allocate and deallocate virtual memory, manage memory on the local heap, create separate heap

memory and allocate memory from the stack. User code can use the memory from the static block of data to load an application. Processes can also use memory-mapped objects to share data. If the running application requires system calls, it passes the call to Nk.exe (kernel) by executing a software interrupt or kernel trap. The kernel then calls the proper server process to handle the corresponding system call. Every system call causes an exception that is caught by the kernel. When an application process makes a system call, it is passed to kernel (Nk.exe) via a wrapper function which is included in the DLL interface (Coredll.dll).

The kernel then handles this exception and determines which .exe can fulfill the request or determines the correct process to send the function call request to. The process that owns the function executes it using the same stack and register values that the original thread load in the calling process.

2.2.2 Memory architecture

Typically the devices which run Windows CE based OS, don't have a hard drive. Instead of hard-drive, these devices use ROM (Read-Only Memory) and RAM (Random Access Memory). Flash memory is another integral part of Windows Mobile data storage.

The memory architecture of Windows CE 5.0 [26] will now be discussed in more detail. As mentioned before, Windows Mobile 6.1 is based on Windows CE 5.0 platform which is a 32-bit operating system and it shares a lot of memory management concepts with Windows XP. Memory mapping and slot arrangements were pretty constant from Windows Mobile 2003 to Windows Mobile 6.0. In Windows Mobile 6.1, memory mapping or slot arrangements has changed a little bit which will be discussed later in detail with a separate figure.

A 32-bit OS can address a total of 4GB virtual memory. This 4 GB virtual address space is divided into two parts each of 2 GB space according to Core OS functions. Upper 2GB space from 8000 0000 to FFFF FFFF is Kernel space and can only be accessed by privileged processes running in kernel mode (KMode). Lower 2GB space from 0000 0000 to 7FFF FFFF (the User space) is used by User processes.

In Windows XP, a process can use the entire 2GB space of virtual memory. However, Windows CE it is not similar to its desktop counterpart. Both user space and kernel space is divided into several regions which is shown in Fig. 2.2 (a) and (b). The user space is divided into 64 equal slots each of 32 MB. Slot 0 (0000 0000 to 0200 0000) is reserved for the currently executing process. Slot 1 (0200 0000 to 0400 0000) is used to load all shared and XIP (execute in place) DLLs from ROM. This 32 MB space is shared by all running applications. Slot 2-32 are used to store 31 other processes running in the system such as Filesys.exe, Device.exe, GWES.exe (Graphical, Windowing and Event Subsystem) Services.exe or user applications (e.g. myapp.exe). The Kernel process acts as the 32nd process in virtual memory.

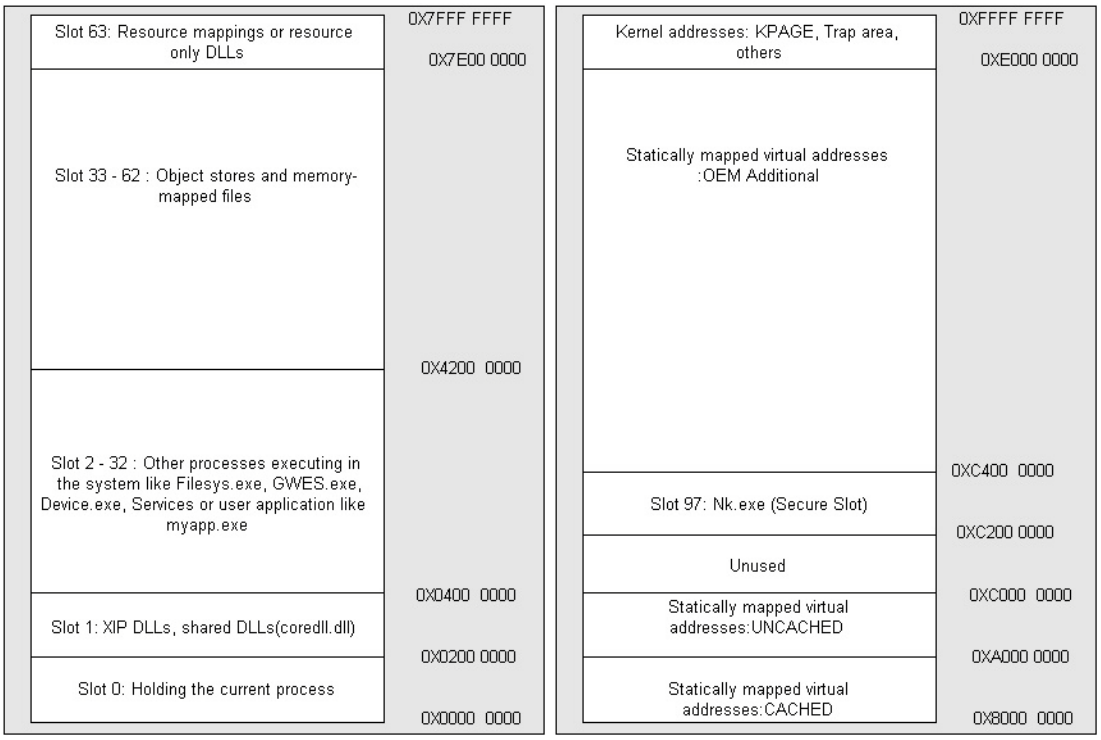


Figure 2.2: a) User space and b) Kernel space in Windows CE 5.0

When a process becomes active, it is cloned to slot 0. Slot 33-62(4200 0000 to 7E00 0000) is reserved for object stores and memory-mapped files. At the top of the large memory area, the 63rd slot is used for resource-only DLLs, the DLLs that have no code but only resource data such as strings and fonts. An expanded look of the lower 64 MB (Slot0-1) are shown in Fig. 2.3.

When a process initializes, the OS maps the following DLLs and memory components in the lower 2 slots (0000 0000 to 0400 0000).

- Shared DLLs like Coredll.dll.
- Some execute-in-place (XIP) DLLs.
- Some read/write sections of other XIP DLLs.
- All non-XIP DLLs.
- Stack.
- Heap.
- Data section for each process in the slot assigned to the process.

For every new thread a new stack is created. The typical size of the stack is 64KB with 2KB reserved for overflow error control. Exceeding the limit of stack by an application causes system access violation and it is shutdown immediately. If an application needs to allocate extra memory, the operating system reserves a block of memory in the heap on a per-byte or a per-4-byte basis. An application starts with a local or default heap. A single application can create as many heaps as needed. These will have the same set of properties as local heap but are managed through a set of heap functions. A block of memory that contains string, buffers and other static values, is loaded by WinCE with every application is called static data block. Variables in the static data block are referenced by an application throughout the

execution time. Read/Write blocks and Read blocks are two types of static data blocks in Windows CE.

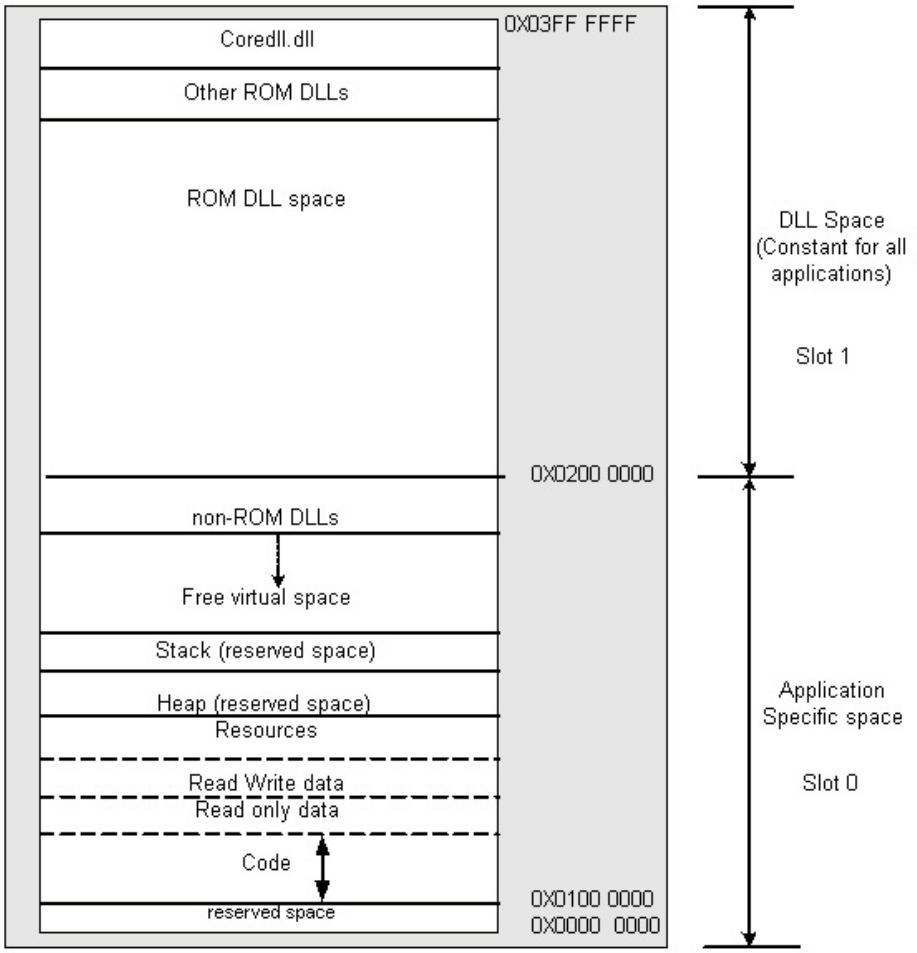


Figure 2.3: Expanded look of Slot 0 & Slot 1 of User space.

As seen in fig. 2.2 (a), slots 33-62 are reserved for memory-mapped files and object stores. Object stores will be discussed in the File System section. **Memory-mapped files** need to be discussed here briefly. If two or more processes need to share a resource or a file, it is done by means of memory-mapped files. A portion of space between slots 33-62 in virtual memory is associated with the contents of the file. Inter-process communication is achieved by means of these memory mapped files. There can be two types of memory mapped files. One can be readable and the other can be writeable. Readable files in the memory can be read by all processes in the system. So, no confidential information should be kept in the memory-map file. Memory of a writeable file can be used by any process to write in that particular file. An application must check the data read from such regions to ensure integrity before using it in any kind of processing.

Douglas Boling of Boling Consulting has discussed DLL loading problem and significant changes made in Windows Mobile 6.1 memory slot's arrangement in his personal blog [27]. In Windows Mobile, there has always been problems with memory regarding a number of DLLs within the system. Usually, DLLs are loaded from the top of slot 1 to bottom. As there are so many DLLs in the system, sometimes they fill up slot 1 and approach slot 0. This results in reducing the already limited

address space for the currently active processes. This can also be a problem in the device manager process space where lots of device drivers, tends to fill up most of the virtual memory, each with their own interrupt service thread.

The slot arrangement in user space has remained constant from Windows Mobile 2003 to Windows Mobile 6.0. However, with the release of Windows Mobile 6.1, a new slot arrangement is introduced to reduce the DLL pressure and to solve the device manager process space. The stacks for the device manager are no more reserved in the processes' slot. The operating system uses slot 59, which is on top of the large memory area (memory mapped files), to store the thread stacks for the device manager. Moreover, changes were made where the DLLs are loaded. XIP DLLs are still loaded in Slot 1. The non-XIP DLLs larger than 64KB are loaded in slot 60. Slot 61 is used in case slot 60 is full. This introduction of additional Large DLL slots helps reducing the DLL strain on the slot architecture. The new memory mapping looks like:

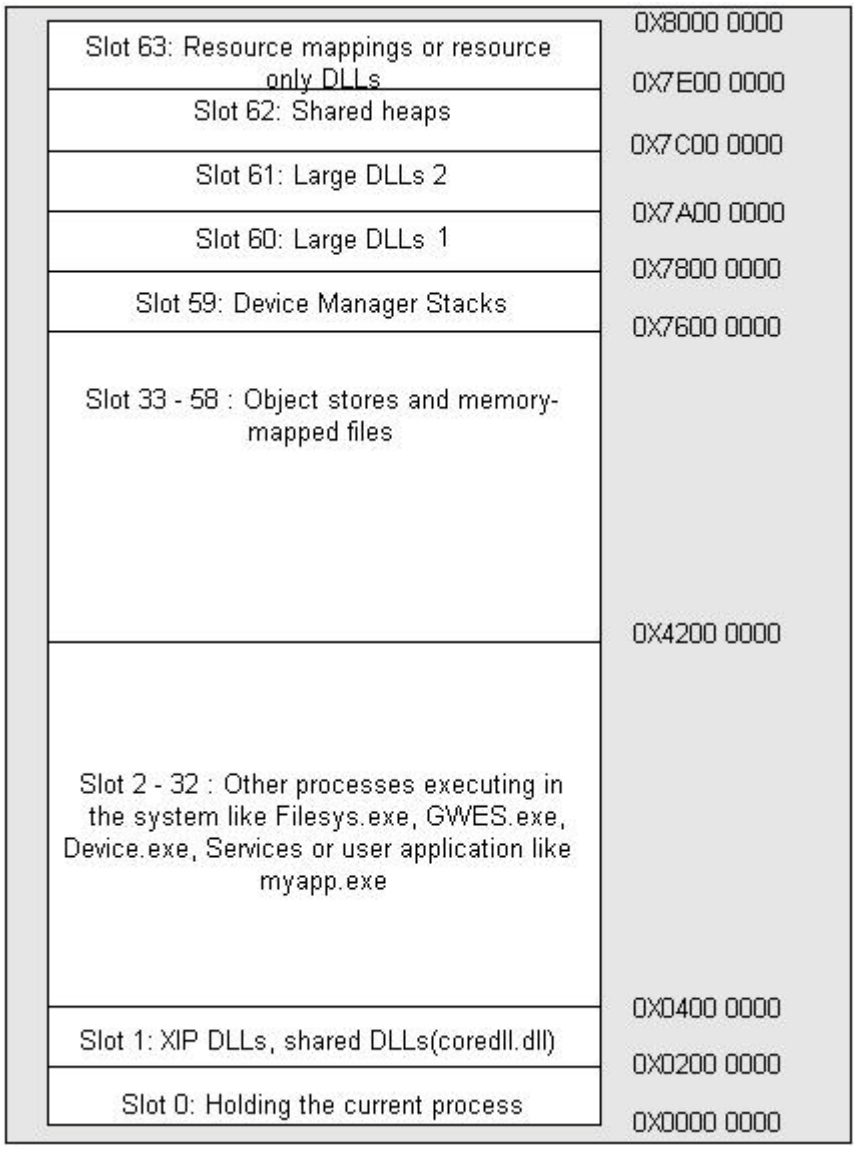


Figure 2.4: New slot arrangements of User Space in Windows Mobile 6.1.

In fig. 2.4, it seems that space for object stores and memory mapped files has been significantly shrunk but it is not the real scenario. The Large memory area (object stores and memory-map files) size, which is close to 1 GB is reduced only by 128 MB.

2.2.3 File System Architecture

The file system organizes the data stored on a storage device into files. Every file system tries to do this organization in such a way that makes finding and accessing information easy. Due to increasing awareness about security, most file systems come equipped with mechanisms like file permissions and fault tolerance in order to keep sensitive data safe and secure. Since the file system manages all the data and provides access to it, it will be a good idea to look into the file systems used by Windows Mobile. First, we will introduce the internal file systems used by Windows CE then we will look into the main components of a file system in windows CE. Finally, we will see how a file system is loaded.

The **Windows CE Internal File System** [28] controls access to the ROM. It also provides access to the RAM resident object store. Windows CE offers two choices for internal file systems.

- **RAM and ROM file system** provides access to ROM and Object store. Applications can store there data in the object store which resides in battery backed RAM. The Object store is mounted as the root of the file system. All the files (except the external file systems visible as folders under the root) visible under the root are stored in the object store. ROM data is accessible through "\Windows". This setting was used by Windows Mobile before the introduction of "persistent memory".
- **ROM only file system** does not provide access to the object store. The applications can store their data in ROM or external storage like memory cards. This setting allows you to mount external file systems as root. ROM data is accessible through the same path and other file systems (external) are mounted as folders under the root. WM5 onwards use this setting. In this way RAM is used much like in Desktop PCs while ROM is used as Hard disk or secondary memory.

Windows CE File System and its components

Windows CE allows custom file systems. A process called Filesys.exe manages the file systems and file system filters. Since Windows CE supports a variety of Block (storage) devices that can have partitions and each partition in turn can have a different file system. A Windows Mobile device can implement more than one file system at a same time which are unified under a single root "\" by Filesys.exe. In Windows mobile, file paths are specified in the same way as in its Desktop counterparts, the only difference is that new drives are mounted as folders under the root i.e. a storage card will be shown as "\Storage Card".

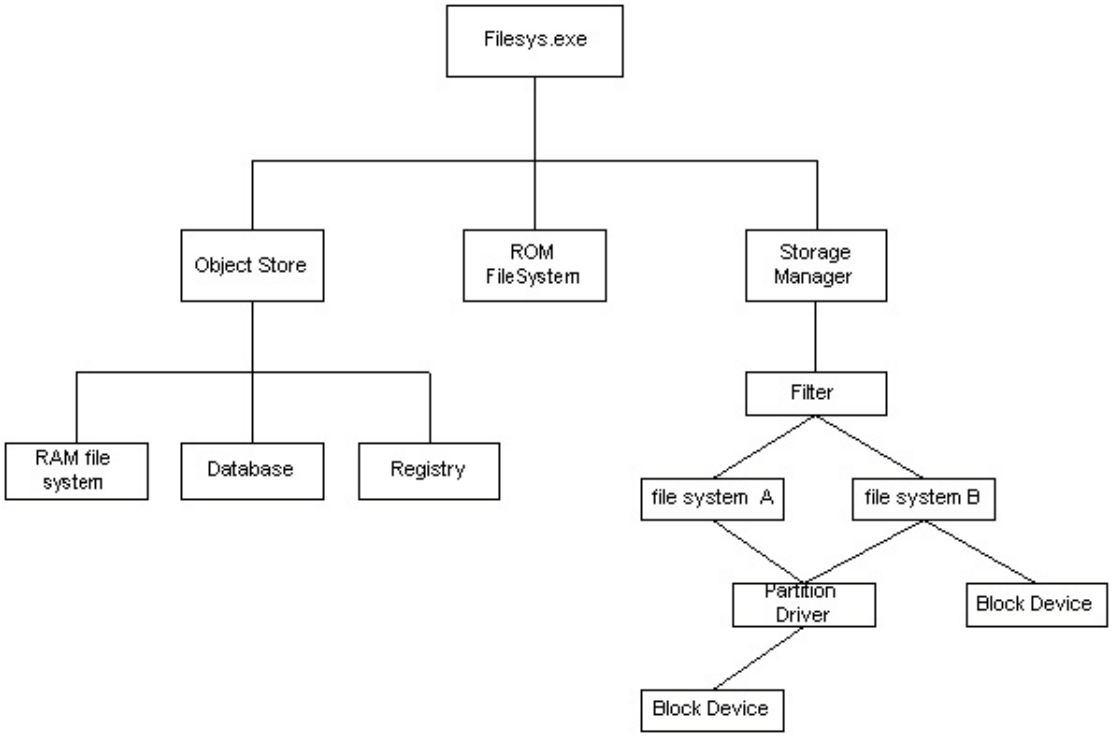


Figure 2.5: Windows CE file system layout

Filesys.exe has three components:

- Object Store
- Storage Manager
- ROM Files system

The **Object Store** provides non volatile storage for programs and user in RAM and for this to work the RAM has to be powered on all the time even when the device is switched off. Object store consists of three optional components- RAM file system, RAM system registry and property databases. The RAM file system is hooked up to the unified root i.e. a file "\file.c" is located in the root of the RAM system and the unified root as well. The ROM file system is hooked up to "\windows" folder. The files in ROM system are accessible as read-only.

The **Storage Manager** manages all the external storage. It deals with four main things:

- **Storage Drivers** or Block Drivers are drivers for Block devices. They read or write to the block devices in fixed sized blocks. Block devices are used as ROM (NAND/NOR) or external memory card in mobile devices.
- **Partition Drivers** manage partitions on a storage device. Windows CE allows multiple partitions on a device with each formatted for a different or same file system. "mspart.dll" is an example of partition driver provided by Microsoft.
- **File System Drivers** organise data into the traditional files and folders on a storage device. Windows CE Includes support for different file systems, including FAT, TFAT, UDFS/CDFS [29].

- **File System Filters** stay on top of Installed file systems (file systems for ATA devices, flash memory and SRAM cards) and process calls for file systems before they are processed by the file system. File system filters are used by different types (compression, encryption, anti-virus) of software in order to pre process the data before it is handed over to the file system. Windows CE does not provide filtering mechanism for access to ROM and RAM file systems.

Let's have a look at how all these components come into play. Filesys.exe resides in the ROM file system. When the OS boots, NK.exe loads filesys.exe, which in turn initializes the registry from the default registry in the ROM file system. Various applications are started by Filesys.exe by reading from the registry. One of those applications is the Device Manager (Device.exe) which loads the needed device drivers by reading from the registry.

The Storage Manager registers with the Device Manager's notification system in order to be notified about the block drivers being loaded or unloaded. The storage manager then queries the block drivers about information like the partition driver used and default file system for a device. All this information is stored in the registry under the PROFILE key where each block device has an associated entry. After acquiring this information, the Storage Manager loads the appropriate partition driver. The Storage Manager then asks the partition driver about the partitions on a disk and the file system for each partition. The partition driver takes this information from the Master Boot Record (MBR) and forwards it to the Storage Manager which then loads the required file system drivers for each partition and mounts the file system in the root.

Since the registry is used extensively in the process stated above and on occasions other than this therefore, we will look at what purpose the registry serves and how windows CE manages it.

Windows CE uses the **Registry** in the same way as desktop Windows uses it. The registry stores data about drivers, applications, user settings, preferences and other configuration data. Windows CE supports two types of registry

- **RAM-Based Registry** resides in the object store. This means that this type of registry is suitable for devices with battery-backed RAM. Registry must be backed up if a device is to be switched off and it does not have battery-backed RAM. Even if the RAM is kept 'alive' while switched off, the battery can run out of power thus resulting in the loss of Object store therefore backing up will be a good idea in any case.
- **Hive-Based Registry** stores data inside hives (files). These files can be kept in any file system. This type exempts the system from backing up and restoring the registry on every switch on/off thus speeding up the boot process. There are two types of hives, the **System hive** which contains system data stored in a file usually named "system.hv" in "\Documents and Settings" while the **User hive** contains data related to a user and is named "user.hv". Both name and path of hives is subject to change depending on OEM. Another hive called the **Boot hive** contains settings that are applied during the boot

process. This hive is in the ROM. This is the hive used in starting the file systems and drivers. Once the file system is started System hive can be mounted thus the boot hive is no more needed and it is discarded. Windows Mobile 5 onwards uses Hive-Based registry

RAM and ROM

Since we are talking about file system and data storing, a brief introduction of the memory that mobile phones use nowadays is important. Mobile phones use flash memory as ROM or Hard Disk. Now the ROM in mobile phones is a bit different from ROM in Desktops. ROM is used to store Operating System, programs and user data. The OS region of ROM is unwritable except by ROM updates, while the user region (where user's programs and data lie) is accessible and writable through normal file read write mechanism. RAM is used similar to Desktops. It is faster than ROM and it takes a lot of power to be kept alive.

Flash ROM

Flash ROM is widely used as ROM and can be either Internal (non-removable flash card) or external memory (removable flash cards). There are two types of flash: the NAND Flash and the NOR Flash. **NAND flash** is faster to write and slower to read while **NOR flash** is faster to read and slower to write. Therefore NOR flash is used for XIPing that is, programs are executed in ROM without loading them into RAM thus saving valuable RAM space. Now this is all OEM-dependent but an ideal system would have a chunk of NOR flash as well as a chunk of NAND flash where NOR flash will hold the programs and NAND flash will hold user data as it needs frequent writing which is relatively faster in NAND.

Flash memory needs special care because of its slow erase and wearing away property. A file in flash memory cannot be overwritten simply; it must be erased and then written. Flash memory is arranged in blocks which are rather large (often in Kilobytes) in size. The memory can be erased and reprogrammed in units of blocks. Blocks have a life span limited to a maximum number of erase cycles after which they start wearing away. Therefore, a file system that is specifically designed for flash memory is needed. A file system that can spread writes over the memory in order to use all blocks equally (wear leveling) and deals with the slow erase time of flash by writing a copy of an updated or edited data to a new block and erasing the old data when it has free time. In Windows CE, FMD (Flash media driver) is linked with FAL (Flash Abstraction Layer) to make a block driver that is then used by file systems like FAT to manage Flash memory.

2.2.4 Device drivers

Device drivers abstract the functionality of a physical or virtual device and manage the operation of these devices with the applications or operating systems. Examples of physical devices are network adapters, audio/video adapters, timers and universal asynchronous receiver-transmitters (UARTs). The File system is an example of a virtual device.

There are three main processes in Windows CE that load device drivers [30]. The first process loaded by the kernel is filesys.se which is a file system process. This particular process loads the file system drivers which must correspond to the file

system driver model. Device.exe (device manager process) and the registry are loaded after the file system. The device manager process is responsible for loading the majority of the device drivers in the system and they all expose the stream driver interface [31]. Lastly, the graphics, windowing, and events subsystem (gwes.exe) loads display and keyboard drivers.

Windows CE-based device drivers [32] run in one of the system process address spaces. In Windows CE based devices, all processes share a single virtual memory address space with the kernel and user space of the OS. Each process is loaded in its own 32 MB slot and slots are protected from each other and also from the kernel. Usually, the device drivers run in user mode and use specialized functions instead of kernel mode addresses while mapping hardware registers. Even the device drivers are not loaded in kernel mode and drivers does not directly use kernel mode addresses, the operating system executes the drivers' entry points in kernel mode because they implement system calls.

2.2.5 Communication Services and Network Stack Architecture

Windows CE provides communication and networking capabilities that enables CE-based devices to connect and communicate with other devices and people over both wireless and wired networks. As Windows Mobile is a part of the Windows CE family, it includes all the major services and network stacks needed to support the advanced mobile phones such as smartphones. The communication services and network stack architecture of Windows Mobile 6.1 can be divided in to three main parts which are Networking (Core), Networking (Remote) and Networking (Wireless).

2.2.5.1 Networking (Core)

Networking (core) consists of EAP (Extensible Authentication Protocol), TCP/IP, Internet Protocol Version 6 (IPv6), Windows Networking API/Redirector and Windows Sockets. In the next few sections, each of the networking (core) elements will be discussed.

Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is an Internet Engineering Task Force (IETF) standard [33] that provides a framework for network access clients and authentication servers to host plug-in modules for different types of authentication methods. The EAP framework in Windows Mobile 6.1 supports four protocols like Windows XP and Windows Server 2003 does, which are Challenge-Handshake Authentication Protocol (CHAP) [34], Transport Layer Security (TLS) [35], Protected Extensible Authentication Protocol (PEAP) [36] and Microsoft Challenge-Handshake Authentication Protocol 2.0 (MS CHAP V2) [37].

TCP/IP Architecture

The TCP/IP suite in Windows Mobile 6.1 has similar characteristics as the desktop versions (e.g. Windows XP, Windows Server 2003). Windows Mobile 6 based smartphones have a TCP/IP stack with the dual stack architecture support and wide range of protocol elements, services and interfaces for easier communication between

peers of different types. The core protocols that are available in the TCP/IP suite are the Internet Protocol (IP), the Internet Control Message Protocol (ICMP), the Internet Group Membership Protocol (IGMP) and the Address Resolution Protocol (ARP). TCP/IP and other networking protocols in Windows Mobile 6.1, uses the Network Driver Interface Specification (NDIS) [38] interface to communicate with network card drivers. Much of the Open Systems Interconnection (OSI) model link layer functionality is implemented in the NDIS interface. Development of network card drivers gets much simpler for the NDIS interface. Moreover, Winsock API acts as a service provider interface between application layer and transport layer. Netbios and Windows Networking API/CIFS redirector plays the role as presentation layer and session layer respectively. Figure 2.5 shows how TCP/IP fits into the architecture that Windows Mobile uses for communications

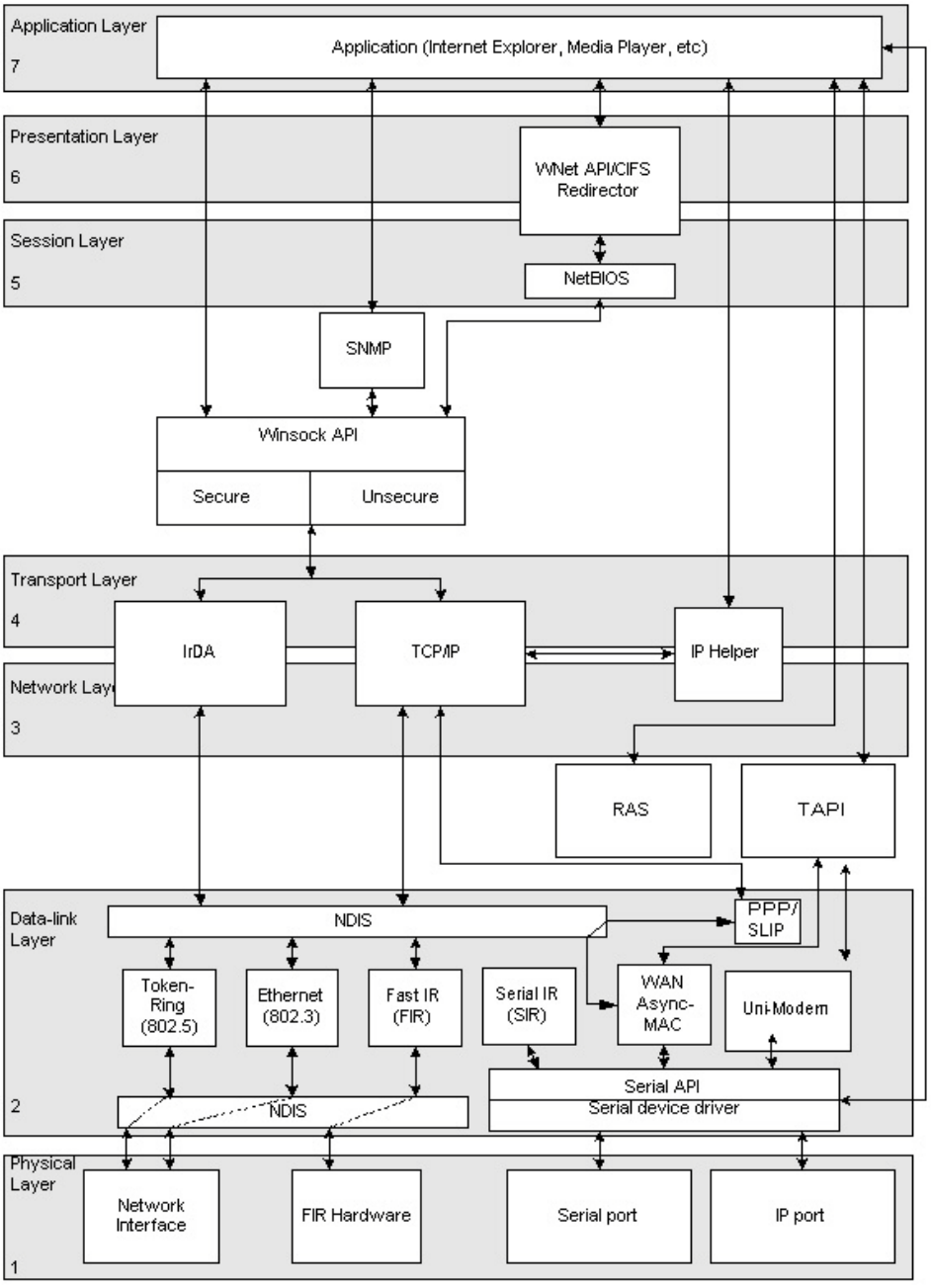


Figure 2.5: Communication stack for Windows CE.

TCP/IPv4 and TCP/IPv6 technology is implemented in two specialized functions (Tcpstk.dll and Tcpip6.dll). IPv4 function is accessible through the Winsock 1.1 and 2.2 interfaces On the other hand, IPv6 function is accessible through the Winsock 2.2 interface only. Moreover, the tunneling mechanisms such as 6to4 [39] and Intrasite Automatic Tunnel Addressing Protocol (ISATAP) [40] are available in Windows Mobile 6.1 to ensure the communication between IPv6 nodes and IPv4 nodes over an IPv4 network. The core protocol stack of IPv6 includes Internet Control Message Protocol version 6 (ICMPv6), Multicast Listener Discovery (MLD), and Neighbor Discovery (ND) and Dynamic Host Configuration Protocol version 6 (DHCPv6).

Windows Networking API (WNetAPI/CIFS redirector)
Windows Networking API/Redirector (SMB/CIFS) implementation in Windows Mobile based devices help to establish and terminate network connections. Also, it provides functions to access files on servers based on Common Internet File Systems (CIFS). WNet API communicates through the CIFS which is also called Server Message Block (SMB) redirector to the remote host.

Applications can use the Universal Naming Convention (UNC) which is a system for naming files on a network so that a file on a computer has the same path when accessed from any other computer. WNet API in Windows CE based devices is almost similar to WNet for Windows-based desktop operating systems except some minor changes. Windows CE based devices do not support drive letter like desktops. Also, Microsoft Windows Network is the only network provider in Windows CE-based devices.

Winsock Architecture
Windows Sockets (Winsock) specifies a service provider interface between the application programming interface (API) and the underlying protocol stacks. Winsock is based on the familiar socket interface from the University of California at Berkley. Windows CE 4.1 onwards uses the Winsock 2.2 which provides easier access to multiple transport protocol stacks. Winsock client and server applications are used as endpoints for network applications.

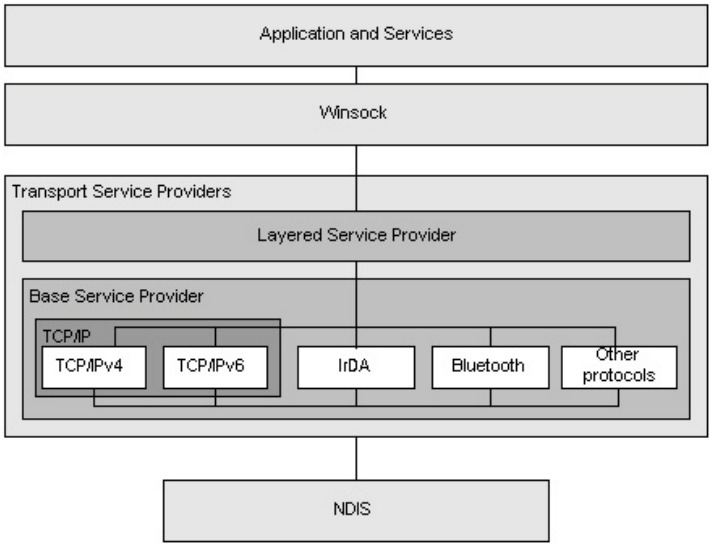


Figure 2.6: Winsock Architecture of Windows CE.

Figure 2.6 illustrates the architecture in Winsock 2.2. If we focus on the architecture, we can see that application and services work on the top of Winsock and Transport service providers' layer. Transport service providers use the NDIS (Network Device Interface Specification) to make easier communication with the network drivers. A brief overview of the architecture components according to the figure 2.6 are given below.

i) Applications and Services Layer
Applications and services created by Independent software vendors (ISVs), operators and enterprises belong to this layer.

ii) Winsock Layer
It is programming interface that provides enhanced capabilities over Winsock 1.1, including installable service providers for additional third-party protocols. Moreover, Winsock 2.2 provides protocol independent APIs that concurrently support IPv4 and IPv6 in applications.

iii) Transport Service Providers Layer
Two service providers are supported in Winsock 2.2:

a) Base Service Provider: It implements a complete protocol. For example, TCPv4 is fully implemented by the WSPM provider that comes with Windows CE. Also, DNS support is implemented by NSPM, default Domain Name Service (DNS) namespace provider in Windows CE-based devices.
b) Layered Service Provider: It is used to modify the transport service provider and therefore the protocol that it implements, to expand, restrict or redirect its capabilities. For example, the Secure Sockets Layers (SSL) LSP is in the layer above the WSPM provider which will encrypts and decrypts data before it calls into WSPM, WSPM then sends and receives data using TCP. Here, SSL LSP as layered service provider above the base service provider WSPM is called the provider chain [41].

iv) NDIS (Network Driver Interface Specification) Layer
It is the standard network driver architecture for all Windows operating systems. NDIS implements the link layer functionality to communicate with network drivers in physical layer.

2.2.5.2
Networking (Remote)
…….....
Networking (remote) in Windows Mobile based devices mainly consists of Remote API (RAPI) and Mobile Virtual private Networking (VPN). As RAPI is used by activesync services, a brief discussion is given regarding activesync here. Moreover, telephony API (TAPI), connection manager and voice-over IP (VOIP) are need to be discussed in detail as they are responsible for voice call management through phone or IP network and network connections through different interfaces. In the following sections, networking (remote) will be discussed in detail.

Remote API (RAPI)
The Remote API library enable applications on desktop to perform actions on remote Windows CE-based devices such as Windows Mobile-based smartphones. By using RAPI, file system manipulation, database manipulation and even query and modification of registry keys is possible. Activesync is a service which uses RAPI to establish connections between desktops to enable above mentioned tasks. RAPI2 is the latest version which is the replacement of the previous RAPI library. It has two significant improvements over RAPI:

- The most significant change in RAPI2 is that it supports multithreaded information transfers between the desktop/laptop and the remote mobile device. Multithreaded means that more than one application can transfer data at a time so that applications using RAPI2 do not have to wait for other applications to finish before beginning their operations.
- Another enhancement includes support for multiple remote devices to be connected with the desktop/laptop. Enhanced versions of Activesync might take advantage of this enhancement. Activesync 4.5 is limited to one device at a time.

ActiveSync
ActiveSync software provides support for synchronizing data between desktops/laptops and remote devices. Time stamps and user preferences are used by the synchronization process to track the changes on both devices and transfers the appropriate data so that both machines has the most recent versions. Activesync supports serial, USB, infrared, Bluetooth, modem and Ethernet connections. It uses the desktop pass-through (DTPT) [42] technology that enables Windows Mobile or Windows CE-based devices to transparently access external networks (e.g. the Internet), through the desktop/laptop computer to which it is connected. DTPT is enabled by default when the device is connected to the desktop/laptop running ActiveSync.

ActiveSync is built on a client/server architecture that consists of a service manager (the server) and a service provider (the client). The service manager is a synchronization engine which is built into ActiveSync and is available on both the desktop computer and Windows CE-based device. It performs the synchronization tasks such as- establishing a connection, detecting data changes, resolving conflicts, mapping and transferring data objects. The service provider determines what data is tracked for changes by the service manager. The desktop provider in the desktop and the device provider on the target device act as the service provider that performs the synchronization tasks specific to user data.

Telephony API (TAPI)
The telephony API implementation is a subset of the Microsoft Telephony Application Programming Interface (TAPI) 2.0 and some parts of TAPI 2.1. TAPI simplifies and abstracts the details of making telephony connections between two or more devices. Windows Mobile-based devices cannot use TAPI directly to make cellular connections, they should be made via the Connection Manager.

TAPI abstracts call-control functionality to allow different and incompatible communication protocols to expose a common interface to applications through its support of telephony service providers (TSPs). Windows CE-based devices come with default TSP, the Unimodem service provider, which supports AT-command based modems. Windows CE as well as Windows Mobile-based devices support installable service providers, which enable independent software vendors (ISVs), original equipment manufacturers (OEMs) and independent hardware vendors (IHVs) to add additional TSPs under TAPI. VoIP such as H.323 and session initiation protocol (SIP) are examples of such providers.

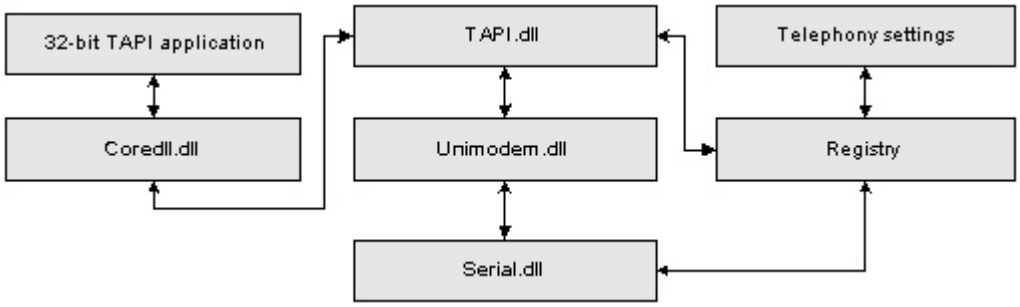


Figure 2.7: Windows CE TAPI Architecture.

Figure 2.7 shows the general architecture of TAPI 2.0 for Windows CE-based devices. In the TAPI 2.0 architecture, the TAPI.dll library is loaded in the protected library Device.exe. When an application calls a TAPI function, it first links to the Coredll.dll which thunks the call to Tapi.dll in the Device.exe process context. The Tapi.dll library validates and arranges function parameters and forwards them to the specific service provider when an application calls a TAPI function. A service provider can provide different levels of telephone service provider interface (TSPI) which are basic, supplementary and extended. For example, a basic service provider might support basic telephony service, such as outbound calls, through a Hayes compatible modem. On the other hand, an extended service provider which has been developed by third-party vendor can provide inbound and outbound call management. Beyond the TSPI layer, the service provider can use any system functions or other parts which are necessary to work with kernel-mode services desgined by OEMs, as well as standard device such as serial and parallel ports, to control external devices.

Connection Manager

The Connection Manager is used in Windows Mobile-based or CellCore [43] enabled devices to manage network connections regardless of the service provider used for establishing the connection. It provides a fast and transparent way of making a connection. The Connection Manager handles many different types of connections, including Remote Access Service (RAS) connections using the Point-to-Point Protocol (PPP), Virtual Private Network (VPN) connections, General Packet Radio Services (GPRS) connections, Proxy Server Connections, Wi-Fi (IEEE 802.11) connections, Wired Ethernet (802.3) connections and Desktop Pass Through (DTPT) connections. Windows Mobile-based devices can access many types of data networks, such as the Internet or a corporate network. A device can connect to each of these networks through multiple connection paths. Cost, security level and specific network considerations for the client application are the basis of choosing path for a particular connection. The Connection Manager tracks the connections whether they are in use

or requested by applications. It closes unused connections and automatically disconnects connections when they have been idle for a specified period of time. Also, it closes low-priority connections to open high-priority connections. In Windows Mobile 6, voice and data communications are labeled as high-priority and low-priority connections respectively. The Connection Manager supports simultaneous voice and data communications [44], which means that high and low priority connections can communicate at the same time.

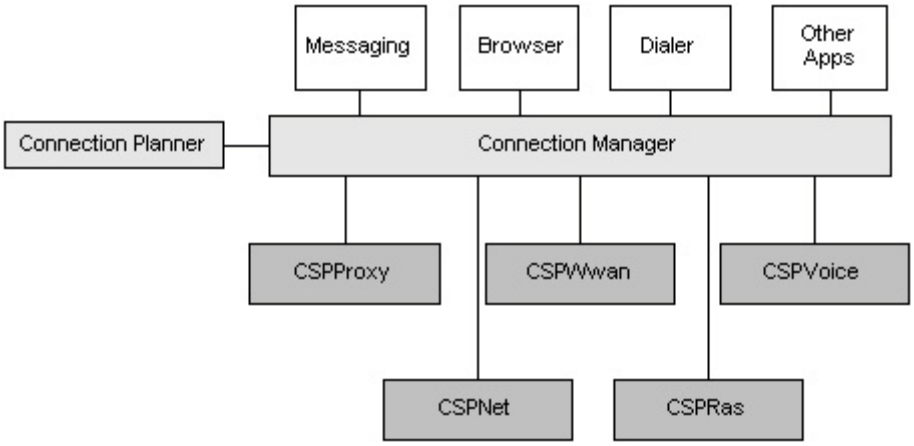


Figure 2.8: Windows Mobile Connection Manager Architecture.

Fig. 2.8 illustrates the Connection Manager architecture. There are three main parts of the Connection Manager:

i. **Connection Manager Application:** ConnMgr.exe plays the role as Connection Manager in Windows Mobile OS. It interacts with multiple device applications such as Internet Explorer and Opera Mobile to schedule and manage network connections. It prepares a list of all connection requests, their priorities and the available connection service providers for each application. When an application call places, the Connection Manager application must first determine the end-to-end paths from the device to the target network, for example, the Internet or a corporate network. Connection Manager retrieves all possible connections from the Connection Manager configuration service providers, and finds all the paths from the device to the target network. Then, the Connection Manager uses Connection Planner to determine the best connection to the target network.

ii. **Connection Planner:** ConnPlann.dll acts as Connection planner chooses the best connections that serve the request for particular connection. Connection planner receives the end-to-end path information from the Connection Manager application. It then queries each Connection Manager configuration service provider to determine cost, bandwidth and latency of each path. With the above mentioned information, it decides which connection is optimal, based on some selected heuristics.

    The Connection Planner also takes the decision about the priority of a connection whether it is highest or lowest and establishes the particular

connection. For example, if a low priority connection is active, Connection planner disconnects it if a connection request for a higher-priority connection is made, unless the high priority connection request is the same as the low priority request. Then, Connection Planner calculates the optimal path for the higher-priority connection and activates that connection. A lower priority connection is re-established after higher priority connection has finished its tasks. For example, A VPN connection using GPRS (either PPP over GPRS or NDIS GPRS) cannot be made if the Wi-Fi interface is already used for a VPN connection.

iii. **Connection Service Providers:** The Connection Service Providers perform the following tasks:

a. Connection information such as route and cost are provided to the Connection Manager application.
b. Store provisioning information received from the Connection Manager Configuration Service Providers to the registry.
c. Bind connection requests to the NDIS User Mode I/O driver.

Connection service providers are implemented as dynamic link libraries (DLLs) which are responsible for specific connection paths. For example, CSPNet is responsible for DTPT and wired network card connections while CSPRas is responsible for GPRS and PPP connections. A table which enumerates the connection service providers and shows the configuration information are given here [45].

Voice over IP (VoIP)

The Voice over Internet Protocol (VoIP) specifies the transmission and reception of audio over the internet. A connection between two peers can be established for example using the Session Initiation Protocol (SIP) protocol [46]. SIP has many functions, such as negotiating the codecs used during the call, transferring calls, and terminating calls. VOIP is an integrated feature of Windows Mobile 6 onwards but depends on OEMs or providers whether it should be included or not.

VOIP application use the Dialplan [47] component, Real Time Communication (RTC) [48] client API that use the VOIP Application Interface Layer (VAIL) [49] to abstract the lower-level interface, and provisioning [50] steps to activate the VOIP settings to make it work in Windows Mobile 6 platform. There is no in-built security with VoIP features in Windows Mobile 6 OS. Authentication and encryption should be included in the VOIP application to make it secure in the network.

Mobile Virtual Private Networking (VPN)

Windows CE supports Virtual Private Networking (VPN). The VPN support in Windows CE includes Layer Two Tunneling Protocol (L2TP), IP Security Protocol (IPSec) [51] and Point-to-Point Tunneling Protocol (PPTP) [52]. Windows Mobile 6.1 onwards provides the Mobile VPN as virtual private network component (VPN). L2TP/IPSec enables enhanced security for VPN client connections from Windows CE-based devices to corporate servers. PPTP is a network protocol that adds a security infrastructure for the transfer of data from a remote client to a private enterprise server, thus creating a VPN using TCP/IP based networks.

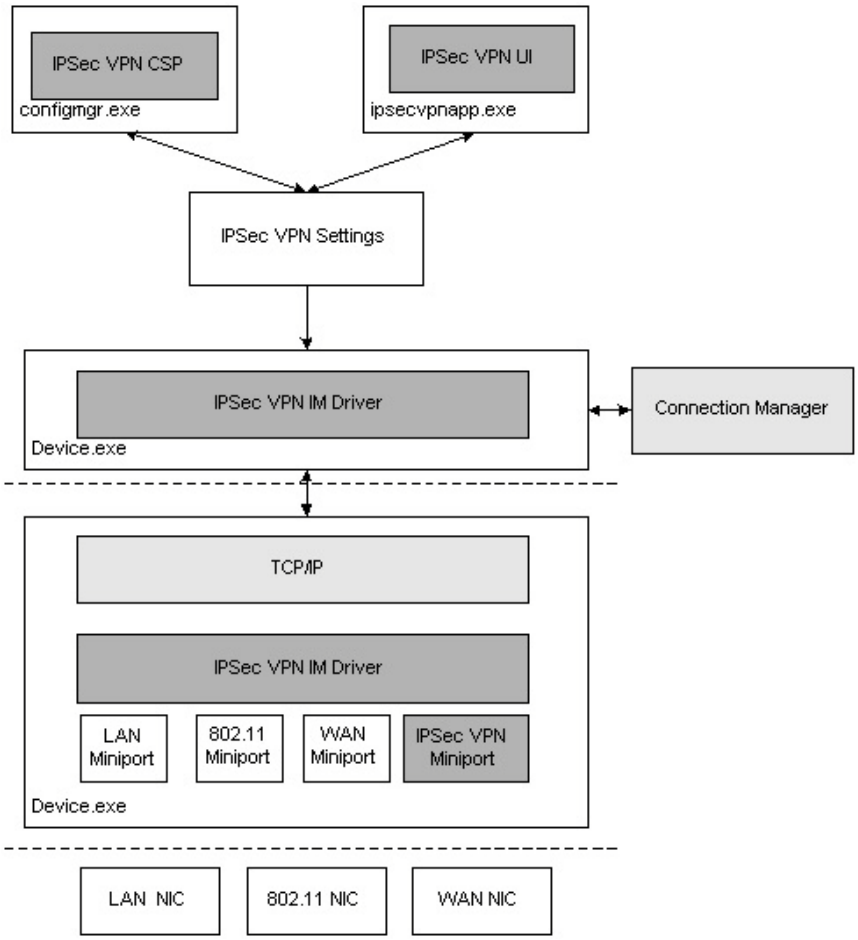


Figure 2.9: Mobile VPN system Architecture.

Figure 2.9 shows the Mobile VPN architecture in Windows Mobile 6.1 which includes five components:

i. **IPSec VPN Policy Manager:** IPSec VPN Policy Manager (ipsecvpnpm.exe) implements the IKEv2 [53] logic, the control and management logic to run the Mobile VPN connectivity and perform Connection Manager operations, NAT timeout detections, persisted connections, and performance measurement, as well as providing APIs for the other system components to control, query the status of, and get notifications from the VPN connections.

ii. **IPSec VPN Configuration Service Provider (CSP):** The IPSec VPN CSP is invoked when VPN group policies are applied to the device, parses the policy values and writes them into the registry for the IPSec VPN Policy Manager to track the changes.

iii. **IPSec VPN User Interface (UI):** It displays the VPN status page to the user. VPN UI is launched when the user selects the Control Panel and Settings entry.

iv. **IPSec VPN IM driver:** This is the NDIS intermediate driver that provides the functionality such as IPSec data-path transformations, packet filtering, routing, and NAT keepalives.

v. **IPSec VPN 802.3 miniport:** This is called the virtual NIC, or the virtual miniport driver that represents the IPSec tunnel. VPN miniport provides a virtual interface to be assigned the allocated private IP address, and allows applications to bind to it to exchange packets through the IPSec tunnel transparently.

Mobile VPN uses the Internet Key Exchange (IKE) v2 to establish an encrypted and authenticated channel that will handle VPN traffic. There are two phases involved in negotiation process. In first phase, an IKE tunnel is established through certificate and Diffie-Hellman key negotiation. In the second phase, the IKE tunnel is used to negotiate the parameters that the L2TP/IPSec security associations will use, such as encapsulation mode, hashing algorithms, encryption algorithms and so on. NAT traversal [54] is supported by the Mobile VPN. The Mobile VPN can use the MOBIKE [55] protocol to update the VPN security associations (SAs) in case of re-negotiation of IKEv2. Faster reconnection, performance improvements through better usability and responsiveness and less bandwidth usage are the major benefits of this approach.

2.2.5.3 Networking (Wireless)

…….....

Networking (wireless) in Windows Mobile 6.1 can be divided into two parts, the wireless telecommunication technologies and the wireless local and personal area networking technologies. Wireless telecommunication technologies can be divided into two categories, like GSM, 3G/WCDMA, CDMA for mobile telephony and GPRS, EDGE, HSPA, EVDO for data communication. Wireless local and personal area networking technologies include Bluetooth, Infrared and Wi-Fi standards. In the following few sections a brief overview is given regarding different technologies and standards of wireless telecommunication and wireless local and personal area networks available in Windows Mobile 6.1.

Wireless Telecommunication Technologies

Wireless telecommunication has evolved over time to provide enhanced bandwidth and additional multimedia services like TV, Video, High-speed internet access. Therefore, previous generation of second generation (2G) is already replaced by current third generation (3G) technologies.

- **GSM:** The Global System for Mobile Communications (GSM) [56] is an open, digital cellular technology standard used in most parts of the world for mobile voice and data services. It supports voice calls and data transfer services via Short Message Service (SMS) at the speed up to 9.6 kbit/s.

- **GPRS:** The General Packet Radio Services [57] is a widely deployed data service and advanced data communication technology for GSM. GPRS is an always-on technology where the customers can be online all the time but have to pay by the transfer volume (kilobytes) instead of connection time. It has also better throughput than the GSM technology, up to 40 kbit/s, which is similar to a dial-up modem. GPRS technology is the base of advanced and feature-rich data services like email, multimedia message service (MMS).

- **EDGE:** The Enhanced Data rates for GSM Evolution (EDGE) [58] is a superset of GPRS technology and is backwards compatible with GPRS. EDGE theoretically supports throughput up to 473.6 kbit/s and can be used for any packet-switched application like high-speed internet access on the move. It can be categorized either as 2.5G or 3G depending on the implemented data rate.

- **3G:** It was developed by GSM community to migrate into 3G evolution to get higher speed than 2.5G. It is based on the International Telecommunication Union (ITU) [59] family of standards under the IMT-2000 [60]. There are several radio interfaces, such as WCDMA [61], CDMA2000 [62] according to ITU's family of 3rd generation mobile communication systems. ITU has not provided a clear definition of the data rates to expect from the 3G providers. UMTS (WCDMA) offers data speeds up to 384 kb/s along with voice services.

- **HSPA:** High Speed Packet Access (HSPA) [63], standardized by 3GPP [64], is the set of two technologies High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA), that extends and enhances the performance of existing WCDMA protocols. Mobile broadbands are deployed around the world with the introduction of this technology. HSPA improves the user experience by increasing data rate up to 14 Mbit/s in the downlink and 5.8 Mbit/s in the uplink. There is another version of HSPA which is called HSPA+ or HSPA evolution increases the data rates up to 42 Mbit/s in the downlink and 11 Mbit/s with MIMO [65] technologies and higher order modulation.

- **CDMA:** The Code Division Multiple Access System (CDMA) [66] is a "spread spectrum" technology, allowing many users to occupy the same time and frequency allocations in a given band. It assigns unique codes to each communication to differentiate it from others in the same spectrum. The maximum data throughput speed is 9.6 kbit/s offered by CDMA.

- **EVDO:** The Evolution-Data Optimized [67] technology is a telecommunications standard typically for broadband internet access in CDMA networks. It was designed as an evolution of the CDMA2000 standard to support high data rates. It is the same as GPRS, an always-on technology, therefore the billing is done based on the transfer volume. EVDO is a 3G technology which offers datarates up to 2.4 Mbit/s in revision 0 and 3.1 Mbit/s in revision A.

Windows CE supports both GSM and CDMA radio stack, and, therefore, Windows Mobile equipped devices include any radio stack which is chosen by the OEMs based on the appropriate hardware. Wireless telecommunication technologies, those discussed above are all supported in the Windows Mobile devices available in the market.

Wireless Local and Personal Area Networks

During the last few years, wireless Local Area Networks (LANs) and Personal Area Networks (PANs) have changed the perception of the mobile devices like smartphones extensively. Multiple wireless interfaces in smartphones let the users to be connected at all time and therefore provides easier access to online resources easier

than ever. Not all personal wireless technologies were developed for network access but also to replace cables by interconnecting devices. The most common and widely used technologies for Windows Mobile phones such as Bluetooth, IrDA and Wi-Fi (IEEE 802.11) are discussed in following.

- **Bluetooth:** It is a wireless communication technology that allows devices to communicate with each other in 10-meter proximity. Usually, Bluetooth [68] is a personal networking (cable replacement) technology which was mainly developed to connect small devices like smartphones and mobile phones with mices, keyboards, headsets etc. Interoperability between devices built by different manufacturers was the main motivation behind it. Bluetooth was specified by the Bluetooth SIG (Special Interest Group) and defines many "profiles". A Bluetooth profile such SIM access or headset profile specifies how devices communicate from the low-level radio protocol, up to the application level protocols. Bluetooth operate in the 2.4 GHz ISM band, same as IEEE 802.11 (WLAN). To avoid interference it uses the frequency hopping spread spectrum and divides the band into 79 channels (each of 1 MHz wide) and changes channel up to 1600 times/sec.

  Bluetooth uses a pairing mechanism where a 4-digit shared secret (PIN) is used between a pair of devices to establish a trusted relation. Then, the SAFER+ block cipher (called E21 and E22) is used for shared link key derivation which is termed as master key. The shared link key is exchanged to authenticate each other and devices can exchange data onwards. Data is encrypted with E0 stream cipher during the Bluetooth communication.

  By definition, Bluetooth is not a networking technology. It can be used as networking client but, in contrast to IEEE 802.11 technology, its purpose is to interconnect the small devices. Bluetooth points to multiple transport and service discovery protocols, and therefore, is much more than a networking infrastructure technology. Most of the devices running Window Mobile 6.1 OS include the Microsoft Bluetooth stack 2.0/2.0+EDR (Enhanced Data Rate) [69].

- **Infrared Communications:** Infrared or IrDA [70] is an old cable replacement technology for the PDAs and mobile phones. It supports low-speed, point-to-point data transfers between devices within visible range. Windows CE as well as Windows Mobile provides support for Infrared communications but currently, most smartphones don't include the feature. IrDA specifies some of the data exchange formats which are adopted by Bluetooth and, therefore, discourage the OEMs to include Infrared in their devices. As the communication requires a short line of sight between devices, IrDA is considered secure. However, no security mechanisms are implemented.

- **Wi-Fi (Wireless Fidelity):** It is a wireless technology based on the IEEE 802.11 [71] standard. It is often refereed to as wireless Local Area Networks (WLAN). IEEE 802.11 defines the physical layer and media access control layer (MAC) sub-layer for wireless communications. The 802.11 standard has evolved over time to enhance the data rates. The first version of the 802.11 standard provided 2 MBit/s, while the second version, called 802.11b,

provided 11 MBit/s. The most recent version 802.11g of this standard provides 54 MBit/s. All of the standards noted above operate in the 2.4 GHz ISM (Industrial Scientific Medical) band. The 802.11a standard provides 54 MBit/s but operates in the 5 GHz band. Currently, all the available Windows Mobile 6.1 smartphones support Wi-Fi interfaces with IEEE 802.11 b/g standard.

IEEE 802.11 supports two operating modes: infrastructure mode and ad hoc mode. In infrastructure mode, a network access point (AP) manages network access among the associated wireless or possible wired clients. In ad hoc mode, wireless clients communicate directly with each other without the use of a wireless AP or wired network. All network nodes have equal control over the spectrum, and, therefore, have to compete against each other for spectrum access.

As the wireless spectrum is a shared medium, any node can listen to other senders if the nodes are within range. Due to this problem, a lot of security mechanisms have been introduced. Windows Mobile OS 6.1 supports all the latest security mechanisms for authentication in WLANs like Wi-Fi Protected Access (WPA) [72] and Wi-Fi Protected Access v2(WPA2). It also supports the Wired Equivalent Privacy (WEP) encryption mechanism which is already proved to be broken [73]. Both strong data encryption standards like Advanced Encryption Standard (AES) and Temporal Key Integrity Protocol (TKIP) are supported in Windows Mobile 6.1.

2.3 Implications

In the previous sections, a detailed discussion has been given about the OS architecture of Windows Mobile 6.1. Based on the discussion some implications regarding different components of the architecture would be helpful while designing applications on these platforms. In the next few sections, observations and limitations regarding the OS architecture will be discussed.

2.3.1 Core OS and Memory Architecture

There are no kernel mode drivers in Windows Mobile 6.1 (based on Windows CE 5.2), and, therefore, processing a system call for an application always need process switching in the kernel (NK.exe). This makes an application execution tedious in the Windows Mobile OS when compared to desktop OS like Windows XP.

Windows CE 5 based operating systems like Windows Mobile 6.1 can run maximum 32 processes including gwes.exe, filesys.exe, device.exe, services.exe and the kernel process simultaneously. Generally, a fresh Windows Mobile 6.1 OS runs 14-15 processes (depends on OEM) which limits the number of running 3rd party processes. A rogue application can easily launch multiple processes to exhaust the virtual address space. Moreover, the 32 process slots are protected from each other by a hardware based MMU (Memory Management Unit) but it is not obvious. Because, there is a simple API call SetProcPermissions [74] in Windows CE 5.0 which can be abused by a poorly coded or rogue application to enable access to the address space of another process.

In Windows CE 5 virtual user space, the lower two slots of 64 MB are used for ROM-based DLLs and active applications. The upper slot of 32 MB is reserved for XIP DLLs and other shared DLLs and the lower 32 MB is reserved for RAM-based DLLs and other application specific resources like stack, heap and .exe image. As the virtual address space in both slots are limited, there can be several problems while loading DLLs by an application. In slot 0, a large DLL loaded by a particular process can leak into the stack space of current process. For example, suppose three processes are loading a series of DLLs. Process 1 has loaded DLL A and B, then Process 2 loads DLL A and C where the DLL C is a larger DLL which consumes most of the address space. In this case, if process 3 now loads a small DLL and a large .exe file, there is a risk it leaks into current process resources. In slot 1, a large XIP DLL or a collection of small DLLs loaded by a rogue or badly coded application can leak into slot 0 to exhaust the address space of the current process.

In Windows CE, an application can reserve a maximum of 32 MB of virtual address space. If any application wants to reserve a large memory chunk greater than 2 MB, the address space will not be reserved in the 32 MB slot, it will be reserved in the Large Memory Area (LMA) which is also called shared memory or memory-mapped files area. This area is accessible and visible to any process, thus provides no security at all. In this area, one process can corrupt other process's data without giving any error or warning in either process. LMA or shared memory can also be used for interprocess communication.

2.3.2 File System Architecture

Windows Mobile uses FATFS for storage cards. It mounts FATFS as the root. FATFS is used for its good performance over small volume, portability (supported by virtually every desktop) and it's easy to implement. In addition to the above mentioned advantages, FATFS introduces FAT related problems as well. FATFS has got no built-in security mechanism and it is prone to fragmentation. A FAT file system offers poor fault tolerance. A survey [75] by Datalight Inc. (a firm dealing in software products for risk-free mobile data) shows that nearly 80 percent of embedded product returns are due to FAT related problems such as data or driver corruption and broken MBR (Master Boot Record). In the survey, Datalight calculates a $6.4 million worth of returned products per year for an OEM due to FAT file system failures.

The FAT file system's interruptible file operations also result in corruption of data. Windows Mobile counters this corruption by using the TFAT (Transaction-safe FAT) File system for its internal memory on top of the block driver. TFAT is designed to ensure the safety of transaction for data stored/modified on the disk. It makes sure that transactions are completed successfully or nothing is modified at all. This ensures the validity of data in case of interruptions during file operations. For external flash memory FAT file system may be used because these devices are supposed to be used/shared with windows desktop environment which does not have TFAT implementation

Details about Protected EAP (PEAP), the TCP/IP architecture, Windows Networking API (WNetAPI) and Windows Sockets are discussed in core networking section. Based on these discussions, some general and security implications are given below.

The **Extensible Authentication Protocol (EAP)** implementation uses third-party authentication code to interact with the Point-to-Point Protocol (PPP) included in Remote Access Service (RAS). PEAP is an extension of EAP and provides secure framework with the help of Transport Layer Security (TLS). It ensures mutual authentication, client identity protection and key generation. There are lots of attack possibilities available against PEAP framework. In one attack scenario [76], the attacker sets up a Windows 2000 CA server and an incorrectly formatted HTML page which might exploit the browser vulnerability to add their CA to a specific client. In this way, the attacker can launch a man-in-the-middle-attack (MITM) against a victim using valid certificates from a rogue CA server and establish a TLS tunnel. In another attack scenario [77], the attacker can establish two different anonymous tunnels. The first tunnel with the access point and the second tunnel with the STA (peer-station). In the first tunnel, the attacker masquerades as the STA and in the second tunnel, attacker acts as the AP. In phase 2 of the PEAP session, the attacker can see the true identity information of the STA as well as gets the ability to forge EAP control messages. Moreover, the channel between the authentication server and access point should be protected as the channel is not protected by the TLS tunnel. As the TLS tunnel is only used for authentication, data transmitted after PEAP authentication is not encrypted. Based on above cases, implementation of PEAP must be setup correctly, otherwise poor settings can introduce severe vulnerabilities.

The **TCP/IP** architecture of Windows Mobile 6.1 is similar to that of Windows XP and Windows Server 2003. There are lot of attacks available in desktop OS versions which have already been patched, but which may be and are successful against Windows Mobile 6.1 OS.

Windows Networking API/redirector is a network file system driver that provides access to files on remote computers. WNetAPI communicates via SMB (Server Message Block) and SMB is susceptible to attacks like MITM (Reflection attack) [78], as the data transmitted between peers are in readable form. Applications in peers should not allow users to directly run an executable file on a network share since it could expose the device to potential threats. SMB signing can be used to prevent a third-party from modifying or replaying requests and responses in the network during transport between a client device and a server. In this case, SMB signing must be supported by the server.

Windows Sockets (WinSock) supports third-party extensions. If these extensions do not use proper security and authentication procedures, they can compromise the security of a mobile device or local network. Winsock supports Secure Sockets Layer (SSL) version 2.0 and 3.0 that provide enhanced security for network communications.

Remote API (RAPI), ActiveSync service, Telephony API, Connection Manager Architecture, VoIP service and Mobile VPN are discussed quite extensively in the

networking (remote) section. Based on this discussion, several general and security implications are listed below.

RAPI security policy is introduced with the release of Windows Mobile 5.0. It has implications for both RAPI and RAPI v2 functions. RAPI security policy provides the following functions.

- **Closed mode:** All RAPI calls are disabled and access to ActiveSync is denied.
- **Restricted mode:** RAPI calls and access to ActiveSync are enabled. ROM-based files and system files are read-only through RAPI. Also, in restricted mode, protected registry keys are read-only. When a device is running in restricted mode, an application, that attempts to use a restricted file, dll or, registry key via RAPI, will simply fail.
- **Open mode:** Enables full access to all RAPI calls. Anyone including other applications can install a DLL file using RAPI, and therefore, invoke certain function to run it trusted.

The Windows Mobile OS runs RAPI in restricted mode by default. But, applications like "SDA_Application_unlock.exe" can use vulnerabilities in the system to unlock the security policies even if the device is running in restricted mode, which is a critical flaw.

ActiveSync poses a potential security risk as the device uses a TCP/IP network. DTPT (Desktop Pass-through) technology is enabled by default to connect the device to a desktop which actually exposes the device in public network thus exposes to network threats. Moreover, attackers can sniff data in wireless environments where the communication channel between the desktop and the device is open and unencrypted. Also, the ActiveSync software can be used to mount an attack [79] against the desktop which can compromise the whole network. DTPT is enabled even if the desktop is password-protected or locked, which is a critical feature in terms of security.

The **Connection Manager** has a dual-homing feature enabled by default which poses a security threat to the smartphone. There are a couple of scenarios when a device would attempt to establish multiple types of connections using the Dual Homing feature. Some examples are:

- The user walks into an area with Wi-Fi coverage while having an active GPRS connection.
- While having an active GPRS connection, the user docks the device and establishes a DTPT connection.

In both cases, dual homing introduces a security threat, and offers potential for bridging traffic between two networks.

In the **Connection Manager**, all traffic is handled by the **VPN** connection until it has been disconnected or a specific request to route traffic to a connection other than the VPN. In this case, the particular connection's security level should be higher or similar to VPN. In table 2.1 security levels are listed with respect to connection types supported in Windows Mobile OS by default.

Table 2.1: Security Level based on Connection-type

Connection Type	Security Level
DTPT	Level 1 (Most Secure)
NIC and Wi-Fi	Level 10
CSD (Circuit Switched Data) and GPRS.	Level 100
1xRTT (CDMA)	Level 100 (Least Secure)

Voice over IP (VoIP) in handheld devices or smartphones has two variants: VoIP over a cellular network (3G/GSM/CDMA-EVDO) and VoIP over Wi-Fi. VoIP over a cellular network is offering some sort of security since most operators today are using NAT for connections through cellular network. This scenario may change in the future when 4G technology will be introduced. On the other hand, VoIP over Wi-Fi is vulnerable in the same way as a regular WLAN environment where users connect to an access point to use data connections over the internet. It is very easy to find a required Wi-Fi hotspot and initiate an attack like MAC spoofing, eavesdropping or spawn a denial-of-service (DOS) attack by enrolling into the network. In most cases, the user looks for a low-cost solution and Wi-Fi is the cheapest technology for VoIP. As VoIP technology itself offers no security at all, it is susceptible to attacks like eavesdropping or injection of VoIP traffic. Considering these scenarios, there are approaches to make the channel secure for VoIP applications which is termed as Secure VoIP [80] [81].

Wireless telecommunication technologies and wireless local and personal area networks are discussed briefly under the wireless networking section. Some implications regarding these technologies related to security will be discussed in the next few sections.

GSM is a widely used telecommunication technology all over the world and has several security flaws since it was designed with a moderate level of security. SIM attacks [82] and SMS-based attacks [83] are the most common in the GSM network. Moreover, GSM only authenticates the user to the network but not vice-versa which gives the attacker a possibility to set up a fake base station that uses no encryption. To avoid eavesdropping, GSM uses the cryptographic algorithms A5/1 and A5/2. Both of them were found fundamentally flawed: A5/1 found vulnerable to two possible attacks by Biryukov et.al. [84], and A5/2 can be broken in real-time with a ciphertext-only attack [85]. The development of UMTS in 3G network introduced an optional USIM which uses a longer authentication key to give higher security than GSM. Also, the base station and the user are mutually authenticated in a 3G network, which eliminates the possibility of installing a fake base station by an attacker. In addition, 3G networks use the KASUMI block crypto instead of the A5/1 stream cipher but have other serious weaknesses identified by Eli Biham et.al [86].

CDMA air interface is inherently secure compared to other cellular technologies like analog Time Division Multiple Access (TDMA) systems. Its security strength comes from the fact that CDMA is a spread spectrum technology that uses Walsh codes.

CDMA uses the specific spreading sequences and pseudo-random codes for the forward (BS to MS) and backward (MS to BS) link. CDMA handoff capability makes difficult for the attacker to eavesdrop in CDMA channel. Synchronization is an important part of CDMA channel to have a stable link. An attacker would only hear noise if the synchronization is not correctly handled while eavesdropping. CDMA2000 includes the 1xEV-DO technology which is an evolution of 1x (1xRTT), which has built-in security that protects the identity of users and makes interception quite difficult. In addition, the authentication procedure is encrypted and an HARQ (Hybrid Automatic Repeat Request) mechanism makes it virtually impossible to identify the user and to correlate user packets.

Bluetooth was conceived as a wireless alternative for RS232 cables originally. This is the reason why security was not a big issue for the Bluetooth stack. In consequence, a lot of vulnerabilities have been found in the Bluetooth stack since its invention. The Bluetooth security model is based on authentication and link encryption. In Windows Mobile 6.1, three security modes are supported:

- Mode 0: No security.
- Mode 1: Service level security in L2CAP layer [87].
- Mode 2: Link encryption in LMP layer [88].

As Windows Mobile 6.1 use the Microsoft Bluetooth 2.0 stack which doesn't support secure simple pairing, it is vulnerable to eavesdropping and man-in-the-middle attacks. Moreover, there were lot of historical attacks like Bluejacking, Bluebug, BlueSmack, BlueSnarf and many more attacks [89] available in the earlier versions of the Bluetooth stack. In January 2009, Alberto Moreno Tablado has shown a new vulnerability [90] in Microsoft's Bluetooth stack which is used in most of the Windows Mobile based devices. The vulnerability has been found in the OBEX FTP Service which is used to share files, browse local shared folders and download and upload files through Bluetooth in a remote WM device. A user, who has been authenticated, can browse parent directories outside local shared folders by using "**. .**/ or **..**\\". During the attack, an attacker can download files without user permission and upload malicious files to compromise the device.

Wi-Fi is a shared medium by definition. Today, most Windows Mobile operating systems support the IEEE 802.11b/g standard to communicate via WLAN. As Wi-Fi unsecured hotspots are increasing rapidly in the public areas like hotels, café shops, airports, train stations etc, devices like smartphones have to faced many more attacks ever before. As a consequence, a thorough analysis has been done of Windows Mobile 6.1 smartphones in WLAN environments and will be discussed in chapter 5.

Besides, above mentioned security implications regarding the Windows Mobile OS, it offers a fairly granular security infrastructure which is discussed in the next section.

Chapter 3
Windows Mobile Security Model

Today's smartphone features play a big role in the security of mobile phones. Mobile phones, unlike desktop PCs, are not kept in a single place. Due to their mobile nature they are more susceptible to stealing or getting lost. Mobile phones offer strong connectivity with more than one way to connect and are always connected to at least one network and are therefore more susceptible to eavesdropping. Confidentiality becomes a bigger challenge when it comes to mobile phones where immature protocols are used with relatively new technologies like Bluetooth. Mobile phones/smartphones combine different features like phone, camera, PDA, walkman and word processing in a single device thus expanding the surface for possible attacks. Finally, mobile phones have very limited resources and processing power compared to PCs and deploying heavy weight security mechanisms is therefore not possible. Limited resources can easily be exhausted which is an ideal situation for Denial of Service (DoS) attacks.

Mobile Phones differ from desktop computers in the sense that they are used by a single user instead of multiple users logging on and off. Consequently, some familiar security measures like Access Control List (ACLs) and User groups that are based on permissions granted to different users are not applicable. Another reason for not using ACLs is the strain put on the limited resources of a mobile phone by ACLs.

Windows Mobile uses a light-weight security model that grants permissions to applications using Certificates. It offers security policies that are used to configure a security setting suitable for the device. This security setting is enforced with the help of security roles and certificates. We will proceed by describing each of the above mentioned terms one by one and explaining how they help in securing a mobile phone.

3.1
Access Permissions

Windows Mobile classifies its assets like Files, Registry Keys and APIs into **Normal** and **Protected**. Basically all those assets (in addition to some others) are protected against manipulation, where an application is able to change the security policy thus compromising the security of the mobile phone [91]. On top of this classification, Windows Mobile uses a concept of permissions for applications. There are three permission types.

- **Privileged:** A privileged application can do anything. It can write to any file or registry key. It can call any API and install certificates. It has full access to all the protected assets. Privileged mode is not needed by most applications.
- **Normal:** A normal application has full access to normal assets only. It cannot call protected APIs or write to system files. Most applications run in Normal mode.
- **Blocked:** If an application is blocked, it is not allowed to run.

3.2
Certificates

A system that runs unauthenticated applications will attract more attackers because it allows the attackers to stay anonymous. Windows Mobile uses certificates [92] in order to authenticate applications. These certificates also determine the applications' permissions.

Windows Mobile maintains a number of certificate stores. The Privileged and Normal certificate stores are two such stores. These stores determine an application's permissions. If an application is signed with a certificate stored in the privileged store then the application runs in Privileged mode. The application runs in Normal mode if it is signed with a certificate from the Normal store.

In most cases, certificates stores on mobile phones are controlled by the mobile phone service provider instead of the owner of the device. Therefore, developers have to cooperate with the OEM, the service provider or any other organization trusted by the service provider in order to get their application signed with a certificate that is in target mobile phones' Normal or Privileged certificate store. In this way developers can get their applications to run on mobile phones in that particular service provider's network. However if a developer wants to target a wider range of mobile phones then he/she can certify the application from Microsoft's certification and marketing program for mobile applications known as **Mobile2Market.** Mobile2Market certificates are included in the Normal or Privileged certificate stores by almost all service providers.

In order to get an application signed with a Mobile2Market signature the developer has to cooperate with a trusted Certificate Authority (CA). The developer purchases a certificate that identifies the developer's applications to the CA. It signs its application with that certificate and submits it to the CA. The CA confirms the developer's identity from the signature and replaces the developer's signature with one of the Mobile2Market signatures. This Mobile2Market signed application can now run on a wide variety of mobile phones.

Mobile2Market provides Logo Certification *'Designed for Windows Mobile'*. Applications are logo certified in order to ensure that they are consistent around different platforms, supports essential APIs and don't cause problems for other applications or features in a mobile phone. Developers submit their applications for testing and certification. This testing is done by Independent Test Labs (ITL) like NSTL, QualityLogic and Veritest. The application is given a Product Certification Code, when it passes the test, enabling it to be submitted for the Mobile2Market catalogue. Developers have to pay for this testing.

3.3
Security Policies

Security policies work on top of the certificates and permissions in order to decide if an application is allowed to run or not, and if allowed to run, then with what privileges. Security policies configure security settings which are enforced with the help of certificates and security roles.

A device can enforce a one tier security or a two tier security models. In **one tier security** it is necessary that all applications are signed and there is no concept of access limitation. A signed application runs in privileged mode with full access to all assets, protected or normal. In **two tier security** the concept of restricted permission (Normal mode) is introduced. So, if an application is signed with a privileged certificate, it runs in privileged mode otherwise it runs in normal mode if signed with normal certificate. Windows Mobile platforms support either One or Two Tier security or both. The following table shows which configuration is supported by a platform.

Table 3.1: Tier Support

Platform	**One Tier Support**	**Two Tier Support**
Windows Mobile 5.0 Smartphone	Yes	Yes (default)
Windows Mobile 5.0 Pocket PC Phone	Yes	No
Windows Mobile 5.0 Pocket PC	Yes	No
Windows Mobile 6 Professional	Yes	No
Windows Mobile 6 Classic	Yes	No
Windows Mobile 6 Standard	Yes	Yes (default)

Some of the most important security policies [93] are:

- **Unsigned CAB policy**: Whether an unsigned cab is allowed to install or not
- **Unsigned Prompt policy**: Whether a user is prompted to accept or reject an unsigned CAB, EXE or DLL.
- **Unsigned Application policy**: Whether an unsigned application is allowed to run or not.

These policies are combined together to form five common security configurations shown in the table below.

Table 3.2: Security Configurations

Configuration	**Description**
Security off	Signed and Unsigned applications run alike with all privileges without prompting the user. Used for development and testing.
One Tier Prompt	Allows signed applications to execute. User is prompted before executing unsigned application. Once allowed to execute, the application runs with full permissions.
Two Tier Prompt	Allows signed applications to execute with permissions derived from their certificates. User is prompted before executing an unsigned application. Once allowed, the unsigned application runs with restricted permissions.
3rd Party Signed	Same as Two Tier Prompt but unsigned applications are not allowed to execute

Locked	Only applications signed with OEM or Operator's (or Enterprise) certificates are allowed to run. Mobile2Market certificates are removed from the certificate stores.

3.4 Security Roles

Windows Mobile uses security roles in addition to certificates in order to protect its assets. The OS makes sure that the role of an operation or a message accessing a resource is equal to or higher than the role assigned to that resource, access is denied if this condition is not satisfied. Table 3.3 describes some security roles. [94] gives a complete list of security roles.

Table 3.3: Security Roles

Security Roles	**Description**
(SECROLE_MANAGER) Manage	This role has unrestricted access to the system. Equivalent to Desktop Administrator.
(SECROLE_ENTERPRISE) Enterprise	Exchange Administrator role. IT administrators have this role to manage enterprise owned devices' settings such as setting password policies or wiping a device.
(SECROLE_OPERATOR) Operator	The role of service provider who can change device settings through Wireless Application Protocol (WAP) Trusted Provisioning Server (TPS).
(SECROLE_USER_AUTH) Authenticated User	The device owner can have this role. Equivalent to User role in Desktop. If a security policy has this role then the user can change that policy.
(SECROLE_USER_UNAUTH) Unauthenticated User	This role is equivalent to Desktop's guest user role. Using this role with an asset allows unauthenticated users full access to that asset.

These roles are assigned to resources like registry keys, databases and files. The resource to role association is stored in a database called Metabase which is initially configured by OEM. This Metabase can be updated or reconfigured through Metabase Configuration Service Provider. Role based security also supports adding new assets to the system.

Operations or messages are assigned roles on the basis of their origin or the certificates they are signed with. If a message originates from a process running in Normal code group then it is given the role SERCROLE_USER_UNAUTH. Similarly a message is given SECROLE_USER_AUTH when it originates from a process running in Trusted code group. If a message is signed with a known certificate then its role is taken from the role associated with that certificate.

3.5
Users' Influence

Security configurations like one-tier prompt and two-tier prompt where users make the final decision about allowing an unsigned application to install and run provide attackers with an opportunity to 'talk' a user into running their malware. Some users fall prey to social engineering and do just what an attacker wants them to do. The fact that a huge amount of users want to tweak their mobile phones by installing free third party software (freeware/shareware) makes the situation more exploitable because this discourages Operators and OEMs from banning unsigned applications to install and run which results in less secure security policies. Two-tier prompt mitigates this problem by restricting the permission of such applications but One-tier prompt, where all running applications have privileged permissions, will not provide any resistance to malware once it is allowed by the user to run. One-tier prompt is used by most of Windows Mobile powered mobile phones today.

Application Locking

There are operators that actually disallow unsigned application to install or run but as expected the user response has not been good. In most cases the users have found a way around this by dismantling the security policy somehow and install unsigned applications. An example is SDA_applicationunlock.exe which was written for T-Mobile's SDA phone. T-Mobile SDA uses Windows Mobile 5 smartphone edition which uses Two-tier prompt configuration in addition to that T-Mobile disallowed unsigned applications to run. SDA_applicationunlock.exe is a desktop application that successfully unlocks the device through a bug in ActiveSync. Now there is a security policy called RAPI (Remote API; used by ActiveSync) policy that is aimed at protecting the device from desktop applications by restricting their access. When the RAPI policy is in restricted mode, the desktop applications cannot access protected APIs or edit protected registry keys. RAPI's restricted mode is one of the best security practices recommended by Microsoft [95]. SDA_ applicationunlock.exe dismantles the application lock by changing the following registry keys in:

HKEY_LOCAL_MACHINE\Security\Policies\Policies\0000-

1001(RAPI policy) → 1 (changed to grant full access).
1005(Unsigned CAB installation policy) →16 (changed to allow installation).
1006(Unsigned Application execution policy) →1 (changed to allow running).
1017(Grant Manager Policy)→ 16 (changed to elevate other roles to Manager).
101b (Privileged Applications policy) →1 (All applications run with privileged mode).

All these keys are protected. Theoretically, a desktop application must not be able to modify these policies if RAPI policy is restricted but the tool mentioned above seems not to be affected by it. It renders the device completely insecure and ready to install and run anything in privileged mode. This puts a big question mark on Windows Mobile security model. We have tested this tool with other platforms like Windows Mobile 6.1 (Professional) and Windows Mobile 5 (Pocket PC) and it successfully unlocks the device. One can argue that this attack may not be successful if password is required for synchronization, but imagine a scenario (ignoring password/PIN cracking or guessing by an attacker for the time being) where an employee wants to unlock his work phone. Even more interesting is the scenario where such a tool is

resident on the employee's computer and the employee is not aware of it. This can expose the device and more importantly the corporate network, of which the device is a part to some serious threats. An attacker can install an application like Flexispy Pro [96], turning the mobile phone into a spying agent that can record calls, normal conversations and other information like call logs, messages, emails and send them to the attacker.

3.6 Security Services

The following security services are also offered by Windows Mobile

3.6.1 Cryptography

Window mobile provides support for cryptography through its Cryptographic API (CAPI). Services provided by CAPI enable developers to encrypt/decrypt data using secure algorithms. Services like encryption, hashing and signing are offered by CAPI. CAPI supports AES (128, 192, 256), DES, 3DES, SHA-1 and RSA.

3.6.2 Storage Card Encryption

Windows mobile uses encryption file system filter in order to encrypt data to the storage card. This can help in keeping the sensitive data confidential in case the storage card containing the data is lost or stolen. Storage card encryption uses AES-128. This feature is not enabled by default. It can be turned on by the user. Enabling storage card encryption can be made mandatory with the help of security policy.

3.6.3 Wi-Fi Encryption

Windows Mobile also supports encryption standards like Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA, WPA2) for use with 802.11a/b/g wireless LANs. WEP is already broken and is therefore not a good choice. WPA2 on the other hand provides stronger encryption with key lengths of 128 and 256.

Other features like SSL, VPN and IPSec are also there in order to secure data in transmission. Internet Information Service (IIS) and Internet Explorer Mobile implement SSL. It has built-in support for Virtual Private Networking (VPN) using Point to Point Tunneling Protocol (PPTP) or Layer 2 Tunneling Protocol with Ipsec (L2TP/IPSec).

3.6.4 Additional Security Features

Apart from the main security model which entails Security Policies, Security Roles and Certificates, Windows Mobile offers some additional features that enhance the security of the device. These features are:

LASS and LAP
Local Authentication SubSystem (LASS) enables standard user authentication that is independent of application. In addition to passwords, it provides the possibility of using sophisticated authentication mechanisms like fingerprints, smart cards etc. Using Authentication Events (AEs) [97] with LASS, one can specify event-based authentication policies that specify when the user should be verified.

Local Authentication Plugin (LAP) is an authentication mechanism that plugs in to LASS. Window CE comes with a default password based LAP. An OEM or ISV can also create a LAP that can be based on advanced authentication mechanisms.

Two types of passwords can be enforced with Microsoft's default LAP: Simple PIN or an alphanumeric password. It also specifies minimum length for password. Windows Mobile 6 onwards provides Enhanced PIN strength which prevents the user from selecting a PIN with simple patterns like 1234, 1111. Password or PIN expiry time can also be set so that the user keeps changing the password. Record of the old passwords is kept so that the user chooses a fresh PIN or password.

Device Lock
Windows mobile can lock the device after a specified time of inactivity. The device configuration can be tuned to enable all the favorable (to the user or to the enterprise) features. Features like password required, password strength, password expiry, password history and automatic lock can be enabled. All this information is kept in protected registry keys.

Device Wipe
Due to the extensive and increasing use of mobile phones in business environments, which usually have sensitive business data, it is a good idea to wipe the data off the device's memory if it is stolen. Windows mobile supports Local and Remote wipe. Wiping overwrites the memory with a fixed pattern which makes it difficult to recover the data. All the settings and user data is wiped out.

Local wipe can be set to activate if the user fails to input correct password for a specified amount of time. **Remote wipe** [98] is activated by issuing a wipe command through the Exchange Activesync management interface [99]. The device acknowledges remote wipe back to the administrator indicating the completion of process. The user of the device cannot block the remote wipe.

3.7 Summary

The Windows mobile platform provides a range of security features that help keep the device safe. Security policies aimed at allowing only signed applications to run and restricting the permissions for most of applications to run in normal mode, as in two-tier prompt, help keeping the devices safe from malicious software (malware) and corruption. Its strong password protection, Device Lock, wipe and password policies like password required for synchronisation, help protecting data in case of device theft or loss. VPN support, SSL and encryption support aims at protecting the data in transmission. Unauthorized penetration of the corporate network is prevented by using strong and flexible client authentication using SSL/TLS, certificates or Exchange

Activesync. Security policies that control OTA (Over the Air) access to the device also help protecting against unauthorized penetration of the device. WM 6 onwards can prohibit Bluetooth discovery mode in order to avoid penetration through Bluetooth.

There are many players involved in the Windows Mobile Security model. First is Microsoft's Windows mobile team, second OEM, third operator and fourth IT administrator or User in case the device is not managed by an IT administrator. The role of the user cannot be ignored even if he/she has been granted the least amount of privileges because he/she is the one in custody of the device. All of these participants have some role to play in security settings. In such a setting it becomes easy to make mistakes like assigning a bad choice of roles to assets in the metabase or carving out a flawed security policy resulting in tools like SDA_application_unlock.

Fool-proof security is only theoretical. We have seen many such claims proven wrong in the past. What can be achieved is to make it as costly as possible for an attacker to break into a system. Windows Mobile tries to provide the infrastructure to ensure just that.

There is a tradeoff between security and usability. The most secure device will be an isolated device which offers no connection to the outside world and does not install any applications, and make it even more secure, it is locked and no one uses it. Only such a device will be perfectly secured (if no one plans to steal or destroy it) but it will not be useful. Therefore it is necessary to come up with mechanisms that makes the device secure without losing on much of its usability. Windows Mobile can be completely locked by not allowing third party applications, signed or unsigned, to run but unfortunately such a device will be considered useless by a great chunk of users (Power users) who want to be able to use their mobile phones like their personal computers. Nevertheless such devices will be useful to corporate authorities that value security. Solutions like Microsoft's Mobile Device Manager [100] that help manage mobile phones like PCs can be helpful to corporate authorities who want to control their mobile fleet, keep it secure and up to date from a single point of control, carving out a security policy that makes the device secure enough while keeping the required level of usability.

Chapter 4
Security Issues

Smartphones are potential victims to all sorts of attacks that have been done against desktop computers. Denial of Service (DoS) attacks and breaking into the system using buffer overflow vulnerabilities or some logical errors have been common and are demonstrated by researchers [8]. Viruses and worms are as much a threat for smartphones as they are for PCs. Cross-infector viruses like Cardtrap [101] that can spread from smartphones to PCs via memory cards or activesync are also there. There are trojans like WinCE/InfoJack [102] that is packed inside legitimate installation files like Google Maps, mobile games etc. In addition to sending some system related information to its author, this trojan also dismantles the application installation security of Windows Mobile powered devices thus making silent installation of malware possible. Smartphones can be targeted by Cross Service attacks that originate from one service/interface (WLAN) and end up using another service (GSM). All these attacks may have following goals:

1. **Proof-of-Concept**: Usually demonstrated by researchers in order to prove that a certain attack is doable in mobile environment.
2. **Denial of Service**: To deny servie to the rightful owner/user. An example of a DoS attack is sending a specially crafted SMS or other network packet that triggers implementation errors thus rendering the device unresponsive.
3. **Stealing Information**: Stealing information like contacts or other sensitive data. This information can be used for spam calls in the same way as spam emails.
4. **Taking Control**: Taking full or partial control of the device so that the attacker is able to do whatever he/she wants. The attacker may make calls from the victim's phone or use other features like camera, microphone or GPS to spy on the owner. Using a paid service like calling or sending MMS will result in overcharging the victim.

4.1 Malware

Proof-of- Concept Worm for WM5

Researchers have worked on proving that worms and viruses can be a threat to mobile devices. For this very purpose they have demonstrated proof-of-concept attacks like Feak worm [103]. Michael Becher et al [104] demonstrate the development of a worm for Windows Mobile 5. They proceed with their worm development by dividing it into two parts; the infector and the spreader. The infector is further divided into two parts where the first part is shell code that is injected into the system by exploiting vulnerability like a buffer overflow in an application over WLAN. The main task of the first part is to download the second part into the device. The second part is composed of an EXE file and an EXE starter. The EXE is started by using the **CreateProcess** API. CreateProcess is a protected API and therefore the exploited

process has to be privileged for the exploit to work in a Windows Mobile Smartphone platform. This is not a problem in Windows Mobile Pocket PC edition as it uses One-Tier prompt configuration. Prompting the user is disabled by changing the **Unsigned Prompt policy** so that the EXE is run silently. The spreader, which is embedded in the EXE, spreads by stealing NetBIOS UDP port 137. Port stealing is a vulnerability that has been patched in Desktop OSes but it is still present in Windows Mobile. The spreader finds the active IP addresses in the network by listening to NetBIOS events on the stolen port. It then starts infecting other devices in the network.

This approach is most likely to work in Windows Mobile 6.1 because first, like Windows Mobile 5, it is based on the Windows CE 5. Second, most of the Windows Mobile 6 platforms (Professional and Classic) do not support Two-Tier security model which means exploiting any application will work. Third, buffer overflow vulnerabilities or other logical errors are bound to be present in the code (Microsoft, OEM or third party) used in Windows Mobile 6.1. Finally, most vulnerabilities like port stealing that were present in Windows Mobile 5 are likely to be present in Windows Mobile 6.1, as Windows Mobile vulnerablities are patched less frequently compared to its desktop counterparts.

Although having Two-Tier security configuration enabled may make it a bit complicated for attacks of this sort to be successful but it does not stop attacks. The attacker can target a vulnerabiltiy in a privileged application. Mulliner's MMS exploit [8] is a good example of vulnerabilities in trusted code. He noticed an overflow vulnerabilty in MMS handling code. Using this vulnerability, he sent an MMS containing the exploit to a device. This way he was able to run arbitrary code on the device without user interaction. Vulnerabilities like this can be used by attackers to launch attacks like worms that spread via MMS. Consider a worm sending the exploit to all the contacts in the phone via MMS. Besides other consequences (depending on what else the worm does), this will result in overcharging of the device owner.

BotNets

BotNet is a network or collection of computers running an exploit that is installed by a virus or worm. Botnets have been used by attackers in scenarios ranging from launching of DoS attacks against a site or server to email spamming. Mobile worms and viruses are a reality now. Phone BotNets are very much possible [105] and such BotNets can be used to launch DoS attacks against the cellular network.

Audio/Video based spyware

Spyware is software that is installed on devices like computers or smartphones in order to spy on the user of device. Spyware steals sensitive/personal information on the device and reports it back to the attacker. Spyware has been targeting information like data stored or input from keyboard on PCs but with the advent of smartphones and the increasing capabilities of this device, especially features like camera and voice recorder have given rise to relatively new kind of spyware, the audio/video based spyware that target smartphones mainly.

Flexispy is an example of such software. It is sold with the tagline of "Catch cheating spouses". This software turns the device into a spying agent that can record calls, normal conversations and other information like call logs, messages, emails and send them to the attacker.

Nan Xu et al [106] introduce a proof-of-concept stealthy video capturer that is designed to work on Windows Mobile 5/6 stealthily. It is designed to invoke video capturing in a way that it is least probable for the user to detect. The captured video is compressed and stored in hidden folders in small files. These files are combined by the sender, which is invoked at an appropriate time, and sent to the attacker via email. Users are lured into installing this spyware by accompanying it with a benign looking tic-tac-toe game.

4.2 Cross Service Attacks

Cross Service attacks are relatively new. They target the primary feature of combining multiple networking technologies (WLAN, Bluetooth, GPS) in a single platform. These attacks can prove quite useful from an attacker's perspective. Cross service attacks have better chances of success because they target a wider surface and in that, they target the weakest link. For example an attacker can attack the smartphone from a relatively insecure interface like WLAN or bluetooth and then try to take control of GSM. The current security model of Windows Mobile does not provide a specific mechanism in order to protect from this type of attack.
In his work on smartphone security, Mulliner introduces a Cross Service attack. He demonstrated his attack on Windows CE .Net. He injected shell code into a device by exploiting buffer overflow vulnerability in a third party application. The vulnerable application, an FTP server, is targeted via WLAN. Once the vulnerability is exploited the injected shell code is able to invoke the calling APIs. Using this approach, an attacker can dial a 900 number owned by him thus crossing the service boundary (from WLAN into GSM) and charging the victim a considerable amount which goes into the attackers account.

4.3 Third Party Applications

Although third party applications are not a part of the operating system but nevertheless they are allowed to install and run by the OS. Therefore, they can compromise the security of a secure OS by introducing new vulnerabilities into the system which provides the attackers new and probably easier ways to break into the system. That's why, it will be useful to give an idea of the security consequences of such applications.

When it comes to security, third party applications are the ones that are blamed the most for their negligence. Most applications require the user to login; these login credentials are then stored in clear text in the registry or in a file. There are applications that use encryption for sensitive information but most of the times the encryption algorithms are really weak. Even though it is a bad practice, most users use the same password for different accounts like email, Voip client, Instant Messaging client, Bank account etc. Using the same password for multiple accounts makes life easier because one does not have to remember a whole lot of passwords. Imagine the consequences if such a password is compromised by a third party application. Some security conscious users use different passwords for different accounts but most have problems remembering them. This problem is solved by saving them in a file or using a third party password or account manager. Everyone knows that the first solution is a bad idea but the second solution can be equally bad or even worse if the password

manager is vulnerable. It is worse because in case of a file of passwords, usually the attacker does not know if the user keeps such a file or not. He/she will have to do a thorough search of the file system in order to get hold of such a file. On the other hand the attacker only has to find out if the user is using some known vulnerable password manager, which is easier, and then exploit the application. The attacker knows exactly where the passwords are kept by the vulnerable application.

Users who care more about security install third party antivirus programs. Antivirus programs can be useful but they can be lethal if they are vulnerable themselves. A sense of insecurity is better than a false sense of security which is offered by a vulnerable antivirus program. A user feeling insecure will be cautious but a user with false sense of security will run into dangers right away thinking that he/she is invincible.

Seth Fogie [107] has done some interesting work on the security of third party applications. He has examined a set of third party mobile applications and unveiled many vulnerabilities. We would like to cite two examples from his work here so that the readers have an idea of what we have said above.

Vulnerable Antivirus

An antivirus program called **BullGuard** has been examined. This program requires a user account in order to update the virus definitions and account credentials are stored on the device. These are sent (encrypted) to the server each time an update is requested.

Vulnerabilities

i. Weak encryption was used to protect the stored password. An attacker can decrypt the existing password.
ii. Virus signatures were stored in plain text. A signature can be changed by an attacker thus enabling an otherwise known virus to infect the system.
iii. Auto delete function can prove lethal. Because the virus definitions database is not protected, an attacker can insert a signature that can match all exe and dll files thus triggering automatic deletion of all those files.

Data protection and management program

A program called **Password Master** has been examined. This program is used to manage information like passwords, credit card numbers and other details in a safe way. It requires the user to login. The user can create a Master Key in order to unlock the details kept by Password Master.

Vulnerabilities

i. An attacker can reset the master key by deleting a specific registry key. This allows full access to the credentials.
ii. Discloses password due to a bug in password hint feature. If the user makes an account without a hint for password, the password hint feature discloses the password regardless of the hint being right or wrong.

4.4
Protection Mechanisms

Along with exposing the threats to mobile security, researchers have been busy coming up with suggestions to enhance the security of the mobile world. Mulliner [8] proposes a Labeling mechanism in order to counter cross-service attacks. He labels processes and resources on the basis of network interfaces they have come in contact with. In this way each process or resource develops a list of labels showing the network interfaces they have been in contact with directly or indirectly. A policy specification file is used to determine whether a process is allowed access to a resource or interface. This is done by specifying in the policy file which label/s a process should not have in order to be allowed to access a particular interface or resource. For example the following rule in the policy file can stop the cross service attack demonstrated by Mulliner:

access wireless_nonfree deny wireless_free

where:
wireless-free => infrared|wifi|bluetooth_voice|bluetooth_data
wireless nonfree => gsm_voice|gsm_data|gprs

This rule denies process that has a label from any of the free wireless services, access to non-free wireless services. The use of customizable policies enables his mechanism to counter new cross service attacks as well. According to the low overhead this mechanism can be termed as light weight solution to cross-service attacks.

A lot of the research is done around Intrusion Detection (IDS) and malware detection. Comming up with effective Antiviruses and IDS has been a challenge in the mobile world. Huge signature databases and complex intrusion detection systems are bound to take a heavy toll on the limited resource pool of smartphones. That’s why most of the intrusion detection is done on the network level. Keeping in mind the evolving processing power and storage capabilities Aubrey-Derrick et al [108] propose an anomaly detection system that runs a monitoring client on the device. The monitoring client extracts features like free RAM, count of SMS sent, process count etc and sends them to a remote server that runs the IDS. Similarly, Markus Miettnen et al [109] propose host-based IDS to be used along with network based IDS. Michael and Ralf [110] use kernel-level system call interception for Windows CE based OSes for dynamic malware analysis. Their solution works by running in kernel mode and denies all other programs the kernel mode which also proves helpful in the sense that potential malware will always have rights inferior to the protecting program thus the malware cannot tamper with it.

In the next chapter, penetration testing has been performed in order to find vulnerabilities in the Windows Mobile 6.1 system.

Chapter 5
Penetration testing of Network Stack

Today, most smartphones support wireless (IEEE 802.11) access as a must have feature which actually makes it possible to compare these devices with the latest notebooks in terms of functionality. The WLAN feature in smartphones makes it possible to access internet in high-speed to a low price or even for free in public places. By introducing this feature, the network stack of smartphones is more exposed than it is through a cellular network. Most of the cellular service providers still use the NAT functionality while giving the IP functionality to the smartphones which offers some sort of security. It is therefore not easy to scan these devices directly in the cellular network and plant attacks against them. But, it is very easy to find out some insecure Wi-Fi hotspots by war driving and attack the users connecting to those hotspots. As the WLAN feature is new in smartphones, it is very normal that these devices can contain an immature network stack which can be compromised by denial-of-service attacks. Network stack implementations in different operating systems (e.g Symbian, Windows Mobile, IPhone OS, Android OS etc.) is responsible for these vulnerabilities. Here, only Windows Mobile and IPhone OS have its predecessors (e.g. Windows XP, Windows Server2003, MAC OS) among the ordinary operating systems. Section 2.2.5.1, fig. 2.5 shows the network stack architecture of Windows Mobile OS, which is similar to that of Windows XP and Windows Server 2003. This similarity actually makes the attackers' task easy to find the well-known tools related to different attacks that were used against the desktop versions. In the next section, the test methodology that is followed to test the robustness and stability of the Windows Mobile 6.1 network stack is discussed.

5.1
Test Methodology

NIST [111] has developed nine useful techniques to assess the network by means of network security testing. These are network scanning, vulnerability scanning, password cracking, log review, integrity checkers, virus detection, war dialing, war driving, and penetration testing. Some of the tests are manual where an individual has to initiate and conduct the test and some of the tests are automated and requires only a little human involvement. Several of the above mentioned tests are often used together to get complete assessment of the security of a network. In our case, the test environment will mainly focus on the network stack of the Windows Mobile 6.1 operating system. So, password cracking, log review, integrity checkers, virus detection, and war dialing will be irrelevant regarding the practical analysis of the network stack. Network scanning and vulnerability scanning might be enough for security testing for smartphones but those two scanning techniques don't give the full security posture. So, penetration testing is also needed to attack different protocols using exploit codes and known penetration tools. We followed a three-step methodology in order to analyze the network stack which are:

i) Network scanning.
ii) Vulnerability scanning.
iii) Penetration testing.

5.1.1 Network Scanning

In the network scanning, a port scanner is involved to identify all targeted hosts on a network. There are some specific services which are very common in Windows Mobile devices such as netbios-ns (netbios name service), wap-push protocol etc. These services can be used for scanning purpose. A comprehensive list of active ports and the services associated with them is listed through the port-scanning tool. Port scanners like Nmap [112], first identify active hosts in the address range specified by the user using ICMP echo and reply packets through TCP/IP stack. After active hosts have been identified, they are scanned for open TCP and UDP ports which will identify the network services running on that host. Moreover, scanners like Nmap gather some additional information through open ports to identify the target operating system. This process is called operating system fingerprinting. For example, if a host has a TCP or UDP port 2948 (WAP-push) open, it is most likely an operating system for handheld devices. Other properties such as TCP packet sequence number generation and responses to ICMP packets (e.g. the time to leave (TTL) field), also provide a clue to identify the operating systems.

The port scanners identify active hosts, services, applications and operating systems, but they don't identify vulnerabilities (except some common Trojan ports). Vulnerabilities can only be identified by a qualified individual who analyze the scanning results. In order to assess the vulnerability in a highly automated way, rather than manual interpretation, a vulnerability scanner should be used against the target hosts.

5.1.2 Vulnerability Scanning

Vulnerability scanners use the concept of port scanners to get a more realistic view about the target device. These types of scanners identify open ports and services on the target hosts and also provide information regarding vulnerabilities through different ports and related descriptions. Moreover, scanners help to identify out-of-date software versions, applicable patches or system upgrades and possible threats to the operating systems and applications by comparing them with lists of known exposures. This type of test is very helpful for selecting the right penetration tools to test the network stack of smartphones. It is very important for the vulnerability scanners to be up-to-date in order to recognize the latest vulnerabilities. These scanners are better at detecting well-known vulnerabilities in a particular surface but not for the zero-day vulnerabilities. As vulnerability scanners are highly automated, high false positive error rate while reporting vulnerabilities is one of the major weaknesses of these scanners. An individual with expertise in networking and OS security must interpret the results in this case. In order to get the full security picture of a host, penetration testing is needed to check the OS network stacks whether they are ready to face the real-world attacks.

5.1.3 Penetration testing

Penetration testing is a security testing methodology in which experts try to evaluate the security features of a system based on their understanding of the system design

and implementation. The main objective of penetration testing is to choose the appropriate tools and techniques used by attackers to identify methods of gaining access to a target device. Penetration testing can be designed to simulate an inside or outside attack. Inside attack doesn't take the security software (e.g. firewalls, antivirus, IDS) into account. On the other hand, an outside attack includes all the perimeter defense tools like firewalls, antivirus etc. In our case, an inside attack is done against the Windows Mobile 6.1 based smartphone.

According to NIST testing methodology guidance [111], there are four phases needed on the way to the successful penetration testing.

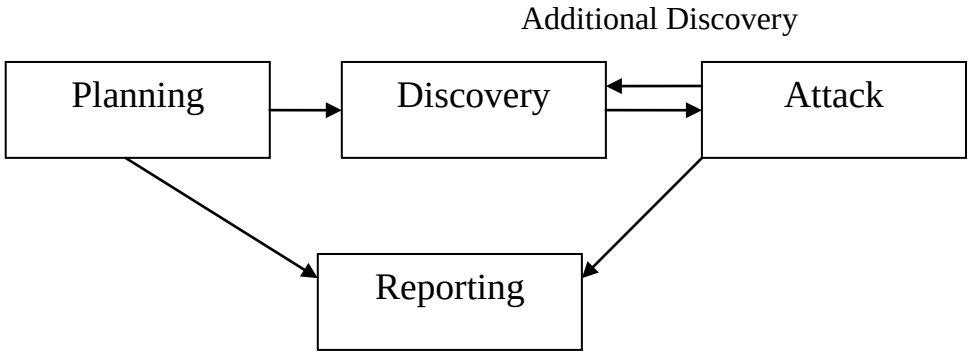


Figure 5.1: Four phases for Penetration testing.

In fig. 5.1, four phases of penetration testing are shown. In the planning phase, lists of the attacks, tools and descriptions of the attacks are reported in the same table. The planning phase sets the groundwork for penetration tests. Reporting is also an essential part for this phase.

Next, in the discovery phase, port scanning and vulnerability scanning is done to find out open ports and vulnerabilities related to them. OS Fingerprinting is done in order to identify the OS running on the target device. This gives the attacker useful information about the architecture and the potential vulnerabilites of that system. In this phase, packet capture technology is a very important tool for analyzing the attacks.

In the attack phase, more emphasis is given to those attacks which will trigger vulnerabilities in the network stack of smartphones. Vulnerabilities are analyzed and reported manually and also with the help of vulnerability scanners. In this phase, real exploit code and different tools which are frequently used by attackers are taken into account

5.2 Penetration Test Results and Discussion

As a part of the planning phase, a complete list of attack tools along with brief descriptions are given in Appendix A. In the discovery phase, Nmap and Nessus [113] were used as network scanning and vulnerability scanning tools respectively. In

the attack phase, a number of attack scripts and some well-known attack tools like HPing [114], arpspoof [115], Packit [116] were used.

All tests were initiated from two clients, one running Windows XP (victim) and another Ubuntu Linux (attacker). The target was a HTC Touch Diamond smartphone running Windows Mobile 6.1 OS. A DLink access point (also used as victim) was used to isolate the attack environment from the public network and simulate a typical real-world scenario. Wireshark [117], a well-known packet sniffing tool was used to monitor the OS responses. In the next few sections, results are reported according to the test methodology stated above.

5.2.1 Network Scanning

In the network scanning phase, firstly, ping scan first was used to find the target smartphone in the network. The MAC address of the target device was also retrieved through ping scan. Second, in order to find out the open ports in the target device, TCP and UDP intense scanning was performed respectively. At last, the OS fingerprinting technique of Nmap was used to identify the operating system running on the target, something which confirmed the resemblence between the Windows Mobile and desktop OS based on TTL, window size, initial sequence numbers, etc. These tests gave several hints about the internal network stack architecture of the device and the services it offers. The detailed test results are given in Table 5.1.

In Table 5.1, the port discovery technique shows that the system has two services running which are netbios-ns (137), netbios-dgm (138), both using UDP ports. The tests show that the Netbios-ns service discloses the netbios-name of the machine, the netbios-datagram service (netbios-dgm) offers support for connection-less file sharing activities. All these services are enabled by default.

By examining the results of the OS fingerprinting test from Table 5.1, according to database of Nmap, an exact match of the OS (Windows Mobile 6) can be found. Moreover, the detailed result also reveals the underlying kernel and processor architecture of the target device, which is based on Windows CE 5.0 and ARM architecture respectively. This investigation showed that it is reasonable to assume that the device has a network stack architecture which is similar to its desktop counterparts like Windows XP and Windows Server 2003. Therefore, it is useful to try some old desktop OS based attacks against the relatively new Windows Mobile in order to see whether those attacks are countered here as well.

Table 5.1: Network Scanning Report

Name of the scan	Port	Command	Results
Host discovery-ping scan	ICMP	# nmap -sP 192.168.0.100	Target device is found alive and MAC address shown.
Intense scan, all TCP ports	TCP	# nmap -PE -v -p1-65535 -PA21,23,80,3389 -A -T4 192.168.0.100	None of the tcp ports were open. Target host was shown alive in the network.

Intense scan plus UDP	UDP	# nmap -PE -v -PA21,23,80,3389 -sU -A -T4 192.168.0.100	**PORT - STATE - SERVICE** 137/udp open netbios-ns 138/udp open|filtered netbios-dgm
OS fingerprinting	TCP,UDP	# nmap -O 192.168.0.100	MAC Address: 00:18:41:94:C7:EE (High Tech Computer) Device type: terminal|general purpose|media device| phone| specialized Running: HP Windows PocketPC/CE, Microsoft Windows 2003|PocketPC /CE|XP, Microsoft embedded OS details: HP Compaq t5520 thin client (Microsoft Windows CE 5.00), Microsoft Windows Server 2003 SP1, Microsoft Windows Mobile 6 or Zune audio player (firmware 2.2), HTC TyTN II (Kaiser) mobile phone (Microsoft Windows Mobile 6), Microsoft Windows CE 5.0 (ARM), Microsoft Windows XP Professional SP2, Microsoft Windows XP SP2

5.2.2 Vulnerability Scanning

To perform vulnerability scanning, the Nessus scanner was used to assess the vulnerabilities related to the open ports and services running in the device. We have summarized the results from the vulnerability scanning of Windows Mobile 6.1 OS in Table 5.2. No significant vulnerabilities were reported by Nessus scanner except the traceroute information marked as (Risk: Low) for the target OS since it leaked information about the devices that could be useful for an attacker. As, the vulnerability scanning has been performed in the internal network, the traceroute information found against the target OS is irrelevant here.

TABLE 5.2: Vulnerability Scanning Results

Port name	**Synopsis**	**Description**	**Risk factor & Reference ID**
netbios-ns(137/udp)	It is possible to obtain the network name of the remote host	The remote host listens on udp port 137 and replies to NetBIOS nbtscan requests.	Risk: None. CVE : CVE-1999-0621 Other references : OSVDB:13577 Nessus ID : 10150
general/udp	Traceroute	Makes a traceroute to the remote host.	Risk: Low Nessus ID : 10287
general/tcp	i)It is used to ping the remote host. ii)Also, used for resolving the IP address to device name	This script displays, for each tested host, information about the scan itself: -The version of the plugin set -The type of plugin feed (Direct, Registered or GPL) -The version of the Nessus Engine -The port scanner(s) used -The port range scanned -The date of the scan	Risk: None Nessus ID : 19506

		-The duration of the scan -The number of hosts scanned in parallel -The number of checks done in parallel	
general/icmp	It is possible to determine the exact time set on the remote host.	The remote host answers to an ICMP timestamp request. This allows an attacker to know the date which is set on your machine. This may help him to defeat all your time based authentication protocols.	Risk: None CVE : CVE-1999-0524 Nessus ID : 10114

5.2.3 Penetration Testing

In the penetration testing phase, we have performed many attacks against the target. The following paragraphs show the detailed results from these tests.

5.2.3.1 TCP Syn Flooding

The Windows Mobile 6.1 operating system's network stack (TCP) is resistant to a well-known exploit, the Neptune or TCP SYN flooding attack [118]. Hping3 was used to carry out this attack and we used the Linux machine as the attacking host in the WLAN environment. The results are shown in Table 5.3.

During the flooding attack, it is very natural that a target device will communicate slowly on Internet because of network congestion. But, the device running WM 6.1 operating system never experienced any denial-of-service due to the attack. The reason behind this behavior is that no TCP ports were open in the target device.

5.2.3.2 IP Options (IGMPv3 Exploit)

The network stack (IP) turned out to be vulnerable against a well-known exploit, the "IGMP V3 DoS vulnerability" [119]. For more details about the attack see Appendix B. We used the Linux machine as the attacker with the Windows Mobile 6.1 based smartphone as the target in a WLAN environment where both machines were connected to internet through a DLink access point. Moreover, a real exploit tool [120] was used after being modified to gcc compatible code to suite our Linux system. The result is shown in Table 5.3.

The device hangs after receiving a specially crafted IGMP packet. It can be sent to the target system and causes it to hang until it is reset. It is easy for an attacker to continuously send such packets to the target and render it completely useless. Some reports also suggest that the vulnerability could be used to execute arbitrary code on affected systems. There is no patch available in Microsoft's security bulletin website related to this attack for Windows Mobile 6.1 [121].

5.2.3.3
ARP Spoofing

…….....

We observed that the target device running WM 6.1 OS, is vulnerable against a denial-of-service attack when receiving a gratuitous ARP for its own IP address. The attack has been done in two ways.

The first attack was done using the target's own IP address but with a MAC address not belonging to the device in order to create an address conflict. In a second test, the attack was done with the target's own IP and MAC address with the intention to see if we could confuse the target. During these tests, we also observed that traffic can be hijacked between the target and router/access point using forged ARP request and reply packets.

We used the arpspoof tool from the dsniff 2.3 suite and also Packit tool for the ARP spoofing attacks. Testing was performed on the Ubuntu Linux host as the attacker, one DLink router/gateway as a victim (whose ARP entry was being spoofed), and one smartphone (HTC Touch Diamond) to which the gratuitous ARPs were sent. The hosts were 192.168.0.101 (attacker), 192.168.0.100 (target- HTC Touch Diamond), and 192.168.0.1 (victim-Access Point). The results are shown in Table 5.3.

Table 5.3: Penetration testing results

Attack name & network stack level	Command to initiate the attack	Result
TCP SYN flood	# hping3 --flood --syn –rand-source 192.168.0.100	During the flooding, the phone can communicate but slowly on Internet. It is not vulnerable to DoS and never freezes. Because, none of the TCP ports were open in the device.
IP Options attack (IGMPv3 exploit)	*# ./exploit 192.168.0.100 192.168.0.101*	Target OS completely hanged after receiving the crafted IGMP packet. Soft reset needed to put the target OS into normal state.
ARP spoofing (gratuitous ARP reply to target itself)	# arpspoof -t 192.168.0.100 192.168.0.100	No changes observed in the target.
ARP spoofing (gratuitous ARP reply to target itself)	# packit -t arp -A 2 -x 192.168.0.101 -X 01:01:01:01:01:01 -y 192.168.0.101 -Y 00:18:41:94:c7:ee -i eth1 -e 00:00:00:00:00:05 -E 00:18:41:94:c7:ee -c 0	No changes observed in the target.
ARP Spoofing request	# packit -t arp -A 1 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.101 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	Hijacks the traffic between router (AP) and target. Here, DoS attack was observed immediately if attacker doesn't reply to the routed packets received from victim. No warning message shown in the smartphone.

ARP Spoofing reply	# packit -t arp -A 2 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.101 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	Hijacks the traffic between router (AP) and target. Here, DoS attack was observed immediately if attacker doesn't reply to the routed packets received from victim. No warning message shown in the smartphone.
ARP spoofing using Broadcast	# arpspoof -i eth1 192.168.0.101	Hijacks the traffic between router (AP) and target. Here, DoS attack was observed immediately if attacker doesn't reply to the routed packets received from victim. No warning message shown in the smartphone.
RARP(request)	#packit -t arp -A 3 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.101 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	No answers. No changes observed in the target.
RARP(reply)	#packit -t arp -A 4 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.102 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	No answers. No changes observed in the target.
Inverse ARP (request)	#packit -t arp -A 8 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.101 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	No answers. No changes observed in the target.
Inverse ARP (request)	#packit -t arp -A 9 -x 192.168.0.1 -X 00:0e:35:73:02:10 -y 192.168.0.102 -Y 00:18:41:94:c7:ee -i eth1 -e 00:14:6c:7b:4a:ad -E 00:18:41:94:c7:ee -c 0	No answers. No changes observed in the target.

5.2.3.4
Other Historical Attacks

…….....

Some other historical attacks such as the Land attack, Ping of Death, Teardrop, Blat, Bonk, Boink, Naptha, Newtear, Opentear, Syndrop, and NETBIOS session request flooding were performed in order to test the network stack's stability and robustness. All of the attacks mentioned above were handled successfully by the Windows Mobile 6.1 network stack except the NETBIOS session request flooding attack.

In this attack, a lot of data traffic was sent through port 139 to crash the remote device. In this case, since port 139 is not open, no responses were coming from the device but TCP retransmission was occurring rapidly without specifying the NetBIOS

name for the destination and source. Netbios names were randomly chosen by the flooder tool. After 2 minutes (approx.), the WM 6.1 smartphone experienced a denial-of-service and couldn't connect to local access point anymore. The target device could not be turned off and the Connection Manager service became useless. The only way to solve this situation was to take out the battery and restart the OS.

Since the target device has a dual stack architecture (IPv4 and IPv6), Neighbor discovery spoofing was performed and was successfully handled by the target OS. This is an attack where a malicious user conducts a spoofing attack through the Neighbor discovery protocol present in IPv6. The vulnerability discloses sensitive information and results in a DoS attack. The vulnerability is caused due to a bug in some implementations of the Neighbor discovery protocol while processing neighbor solicitation requests. It can be exploited by someone adding a faked entry to the neighbor's cache via an ND solicitation request, which may contain spoofed IPv6 address.

5.3 Summary

In this chapter, we have analyzed the stability and robustness of Windows Mobile 6.1 based smartphones. We have carefully selected tools and attack methods in order to be as successful as possible in the tests.

The tests include some rather well-known attacks against vulnerabilities like session hijacking using forged ARP request and reply packets which are patched in desktop operating systems.

The WM 6.1 was found vulnerable. It ceased to function and could not communicate reliably on the WLAN during a SYN DoS attack where the devices were flooded with SYN packets.

The Windows Mobile device was also found vulnerable against ARP spoofing. It was possible to hijack the network traffic from the device using faked ARP replies. This problem is perhaps somewhat expected but the devices could at least give a warning when MAC or IP addresses suddenly change.

The tests show that it is easy to spawn DoS attacks against the Windows Mobile based smartphones. The Windows Mobile 6.1 device faced DoS against a well-known IGMPv3 exploit. In an earlier paper [14], we have shown that it was also possible to do similar attack against Windows Mobile 5 devices with similar results where the devices were rendered useless. In another attack scenario, session request flooding was directed at NETBIOS protocol 139 (tcp) of the target device which completely made the device unresponsive.

Looking at the results from the tests, it seems that the network stack in Windows Mobile devices contain several vulnerabilities and is not fully capable to operate in complex and hostile network environments such as on the Internet where hosts are constantly searched and scanned and many of these attacks are known to be constantly present [122]. We believe that these findings will be beneficial to the vendors, security researchers and practitioners and third party vendors who develop firewalls,

antivirus, and intrusion detection systems for Smartphones. The work should also be useful for end-users and organizations who need to know what level of security and stability these devices offer. The results here should make it possible to know in what situations these devices should or should not be used.

In the next chapter, mobile VoIP application is shown as an example to point out the security issues that it will face from application and network perspective.

Chapter 6
Case Study: Mobile VoIP Application

Mobile Voice over IP (VoIP) is a communication technology where an individual can dial and receive voice calls in handheld devices (e.g. smartphones or PDAs) via the IP network. Typically, voice calls are made through a voice carrier's network but in VoIP, voice calls are made with the help of a VoIP service provider (e.g. SIP server). Calls can be transported through Wi-Fi (known as VoWLAN or Voice over WLAN) or a cellular carrier's data network (known as Vo3G or Voice over 3G). Due to the VoIP technology, subscribers of cellular network are not subjected to long distance charges from their cell operator. Although subscribers have to pay applicable charges, they are very low compared to the long distance cellular charges. In case of Wi-Fi, subscribers can take the advantage of free hotspots and transport the call only paying the VoIP service providers' charges.

6.1
Overview

Any handheld device that supports high-speed IP communications can use the mobile voice over IP (VoIP). Several survey reports show that, VoIP subscribers are going to increase significantly in coming years. Watchguard Technologies, a security firm, has pointed out in a report [123] that around 75 percent of corporate lines will be using VoIP in next 2 years. Moreover, by the end of this year, the total number of VoIP subscribers worldwide is expected to reach nearly 100 million. A fairly large number of these users will use the smartphone as VoIP platform. So, the OS vendors are integrating VoIP support such as SIP Clients in the smartphones. According to another prediction by Disruptive Analysis [124], there could be over 250 million VoIP over cellular network users by the end of 2012. As the underlying protocol for the VoIP application has no built in security, there could be serious threats towards the voice data over the air as well as in the operating system on which the application will run. This is the reason why we have chosen a Mobile VoIP application as a part of our case study while evaluating the security of Windows Mobile 6.1 operating system. In the next few sections, there will be discussion on Mobile VoIP support in Windows Mobile 6.1 and related security issues.

6.2
Mobile VoIP in Windows Mobile 6.1

The Windows Mobile 6.1 operating system includes the "Internet Calling" feature by integrating a SIP client. But, in most cases, OEMs and the cellular operators don't enable this feature as a business strategy. This strategy does not prove helpful in stopping the users from using VoIP application over cellular network or Wi-Fi. A number of technical forums have already published the way to enable VoIP support in Windows Mobile 6 based devices. As a result, third party unsigned applications are becoming a part of VoIP support in Windows Mobile OS. This can pose more security threats.

6.3
Mobile VoIP Security

Security threats to Mobile VoIP come in two ways. First, attacks can be done over the air when VoIP is running over Wi-Fi or 3G environments. This is termed as external attacks. Second, a third party application running on the Windows Mobile OS can be used to perform internal attacks. Both types of attacks will be discussed in the following two sections.

6.3.1
External Attacks

Mobile VoIP over Wi-Fi faces threats similar to other applications using Wi-Fi. Major threats include ARP spoofing, eavesdropping, denial-of-service, SPAM over Internet Telephony, and voice phishing (Vishing) attacks. A brief description of these attacks is listed below.

ARP Spoofing
If an individual is using a public Wi-Fi access point to place a VoIP call, an attacker residing in the same network can send gratuitous ARP reply packets (spoof the MAC of AP) to the target to redirect the voice traffic to attacker's device. The attacker can inject malicious traffic or simply observe the traffic to retrieve confidential information. In section 5.3, the penetration result shows that the Windows Mobile 6.1 OS is vulnerable to this type of attack.

Eavesdropping
As the voice traffic is sent in clear through Wi-Fi during VoIP conversation, malicious user can sniff the traffic over the air. Attackers can inject traffic into the wireless network without even connecting to the access point.

Denial-of –Service Attack
This type of attack is quite common in data networks. There can be several scenarios related to DoS attacks. Attackers can simply send crafted TCP or IP packets to the target which can interrupt the VoIP conversation. In another scenario, an attacker can run multiple packet streams such as a lot of SIP call requests or SIP registrations to exhaust the server resources causing busy signals or disconnecting of a Mobile VoIP users. Another typical scenario is that a malicious user can inject excessive traffic to flood the network causing other VoIP users to disconnect. In section 5.3, Windows Mobile 6.1 is found vulnerable to such DoS attacks.

Spam over Internet Telephony (SPIT)
Spam over Internet Telephony (VoIP) are sent for direct monetary benefits or denial-of-service which can damage recipient's business investment in terms of missed calls or disruption of calls. SPIT can be generated in similar ways as ordinary spam with botnets that target millions of VoIP users from the compromised systems. In case of Mobile VoIP, like junk e-mail, SPIT can slow system performance, clog voicemail boxes and even annoy the user to switch off the mobile phone which is unlikely in terms of email.

Voice Phishing (Vishing) Attack
This type of attack attempts to get users to disclose personal and sensitive information, such as user names, account numbers and passwords. The trick works by spamming users using SPIT technique and luring them to call their bank or service provider to verify account information.

6.3.2 Internal Attacks

Unsigned third-party applications have always been a blessing for the users to tweak the target OS platform and do some useful tasks which cannot be done with bundled applications that come from OEMs or OS vendors. This scenario is quite popular in the smartphone operating systems like Windows Mobile and Symbian. In Windows Mobile 6.1, Microsoft has included the SIP client as a part of VoIP support but OEMs or operators disable the feature as a business strategy. As a result, the users install unsigned applications to enable the feature and use a third party SIP client to place a VoIP call. Some threats introduced by this situation are listed below

- In the first scenario, by installing a third-party SIP client in Windows Mobile 6.1 OS with one-tier security (most of the devices in the market are using the one-tier security based WM platform) can compromise the whole platform. In one-tier security, after installation, any unsigned third-party application runs in privileged mode and it can access any restricted API of the target OS. A rogue SIP client may act as a Trojan by including an extra functionality which can record the VoIP calls and send it to an attacker through any interface (e.g. WLAN or Cellular Network).

- In the second scenario, the rogue application may enter the wrong settings for SIP server. That particular server may be installed by the attacker to give proper service to the user but intercept all voice traffic on the way where the user will not notice anything at all. In this scenario, the rogue SIP server will act as SIP proxy just to relay the traffic to the original server which can be treated as man-in-the-middle-attack.

- In the third scenario, The rogue application may include a key logger software that records the type sequence while installing the SIP client. In this way, the SIP settings including user name and password can be leaked easily which can be used later on for the attacker's own purpose.

To overcome the security issues that are faced by Mobile VoIP architecture, a number of solutions are available to secure it. Since Windows Mobile OS supports SIP (Session Initiation Protocol) protocol by default, secure mobile VoIP solutions discussed here mainly focus on SIP. Israel [81] has proposed a secure model of mobile VoIP which will counter most of the above mentioned attacks like DoS attacks, ARP spoofing, reply attacks or message integrity violations, as well as eavesdropping of the media session. Moreover, Rachid [125] has discussed about SPAM over Internet Telephony (SPIT) and its protective measures in detail. Most of the solutions for secure mobile voip focus on the external attacks but not the internal attacks which are related to the platform security on which the voip application runs.

The next section provides a general guideline for platform security.

6.4
Suggestions

Here are some of the suggestions that can help towards providing a more secure system:

Security Awareness
Well, this is the most cliched suggestion for security. We had to start with it because we feel that it is really needed in the mobile world by users and application developers. Awareness can really strengthen security in the case of smartphones by enabling the application developers to take care of the security while developing software. It also helps the users to make good use of the provided security mechanisms like passwords, encryption, security policies etc.

Development specific
Applications like voip client or other streaming applications use jitter buffers. These buffers can have large sizes (> 2MB) depending on the underlying algorithms. Special care must be taken not to reserve space for them in the Large Memory Area (LMA) (Slot 33- Slot 58) because it is unprotected and accessible to all other applications. By default, Windows mobile reserves address space larger than 2MB in LMA instead of the 32 MB process slot. This behaviour can be changed by the developer by reserving the address space in small chunks (< 2MB).

Carving out a strong security policy
Since OEMs or corporate IT administrators can change or tune the security behaviour of smartphones therefore they must carve out strong security policy for their devices. By strong security policy we mean that unsigned applications must not be accepted wherever it is possible. All the assets must be assigned highest possible required roles. Settings like RAPI restricted mode must be used and all other features provided by Windows Mobile like password required, device lock and wipe etc must be enabled.

Check if policies are intact
We have seen that the policies can be changed by applications like "SDA_application_unlock". Carving out strong security policies is of no use if we are not sure that they will remain intact. Therefore, it is useful to check frequently if the policies are intact. This can be done by the IT administrator or the cellular network operator. This practice is highly recommended until one is sure that the policies cannot be changed maliciously.

Two-Tier Access
Devices that handle sensitive personal or business data must have Two-Tier Access enabled. This will provide some protection against the unsigned or normal applications by restricting their access to unprotected assets only.

Limit applications' capabilities
Applications (privileged or normal) must not be allowed to change the security policies. Privileged certificates provide applications the capability of changing the security policies. This can be dangerous because even if the applcation is not

malicious, it can have a vulnerabilty (e.g buffer overflow) that can be exploited by an attacker thus enabling the attacker to change the security policies. This leads us to our next suggestion for Windows Mobile vendors.

Windows mobile applications can be privileged or normal. Privileged applications have full access to all the assets (protected or otherwise). This means that such an application is capable of doing anything like changing the security policies or sending files to remote locations even though its original job maybe browsing the Internet. This is rather coarse grained. The assets of Windows Mobile must be divided into more classes and new types of certificates must be introduced to grant access to those classes. At the very least, all the assets that are part of the security policies must be classified under a new class say Root and Root certificates must not be issued to ordinary applications.

Third Party Security Products

Security products like antivirus software or firewalls were not a good solution for smartphones in the past because of smartphone's limited processing power and other resources but with the increase in smartphone capabilities their usage in this environment is becoming possible. Major players in the field of security products have now started taking interest in introducing products for smartphones. It will be very useful to have an extra line of defence in shape of antivirus software and firewalls.

Patches

We have seen that there are vulnerabilities in the operating systems. The Trusted code can have vulnerabilites like buffer overflows and the network stack can be vulnerable to attacks that are already patched in the desktop environment. These vulnerabilities must be patched with the same zeal and speed as in desktops. Windows Mobile automatic update feature can be helpful but it is useless if the vendors do not introduce patches for their mobile OS.

Chapter 7
Conclusions

In this thesis, we have focused on Windows Mobile operating system security since this is the second most popular OS in the consumer market of smartphones and we feel that its similarity to Windows desktop systems will attract the Windows traditional clientelle. Most attacks today are aimed at Symbian platforms due to its higher availability in the consumer market, but this can soon change and be Windows Mobile very soon.

Windows Mobile supports a limited number of processes and it uses a slot based memory architecture. It uses a hardware based MMU that protects each prosess's slot. The limited smartphone resources are easy to consume thus DoS attacks are also easy to perform. DoS attacks may not be as lethal in the mobile world as they are in the Desktop world but they are often combined with other attacks (e.g Blind Spoofing) which make them more dangerous.

Smartphone operating systems face the same threats that are faced by the desktop OSes face. The threat from malware like viruses, trojans and worms is realized and BotNets are going to be a reality in the mobile world very soon. Threats from vulnerabilities like buffer overflows or other logical errors in trusted code or third party applications cannot be ignored.

The problem with the smartphone operating systems is that a known vulnerability (that is long patched in Desktop OSes) tends to exist for a longer time due to delayed patches thus they can be an easy target to exploit. In this thesis, penetration testing has been done against the Windows Mobile 6.1 operating system in order to expose the vulnerabilities on the network level. The network stack of Windows Mobile is found vulnerable to some known attacks which have been taken care of in desktop operating systems.

Windows Mobile does offer a security infrastructure that uses security policies with code signing. But, as we have seen, it is not impossible to bypass the security policies. The division of application permissions into only two levels (privileged and normal), with privileged applications being capable of doing virtually anything in the system, is coarse grained. Normally, an application that needs access to a single protected asset in order to carry out its job should not be allowed to access all the protected assets, but Windows Mobile does not offer such a restriction.

Although our report focuses on Windows Mobile, we have realized that most of the problems we have pointed out in this report are common to almost all mobile operating systems. We can say that while the vendors of mobile OSes have started to pay attention to OS security, these mobile OSes still have a long way to go in order to prove themselves truly worthy of the trust that is invested in them by their users today.

We suggest the following future work in connection to this thesis:

- Third-party security products are being introduced in the mobile world. It will be interesting do evaluate the of mobile OSes in presence of these products.
- Intrusion detection or prevention has always been a big challenge for the smartphones due to its limited processing resources and disk memory. IDS/IPS development on the smartphones' operating systems in an efficient way can be a good research area.
- The development of an automated penetration testing tool will ease the security assessment for mobile devices.

REFERENCES

[1] Andrew Bittau, Mark Handley, Joshua Lackey, *"The Final Nail in WEP's Coffin", http://tapir.cs.ucl.ac.uk/bittau-wep.pdf*, visited on 7th July 2008.

[2] Jesse D´Aguanno, *"Blackjacking-Owning the Enterprise via the Blackberry", http://www.praetoriang.net/presentations/blackjack.html*, presented at Defcon 14-Las Vegas, NV 2006.

[3] *http://cdecas.free.fr/computers/pocket/simon.php,* Visited on 11th March 2009.

[4] Jukka Ahonen, *"PDA OS Security: Application Execution"*, Publications in Telecommunications Software and Multimedia TML-C7, ISSN 1455-9749, Helsinki University of Technology.

[5] Arto Kettula, *"Security Comparison of Mobile OSes", http://www.tml.tkk.fi/Opinnot/Tik-110.501/2000/papers/kettula.pdf* , visited February 2009.

[6] S. Perelson, R.A. Botha, *"An Investigation into Access Control for Mobile Devices", ISSA 2004*, June 2004, Gallagher Estate, Johannesburg, South Africa.

[7] Collin R.Mulliner, *http://mulliner.org/* , visited January 2008.

[8] Collin Richard Mulliner, *"Security of Smart Phones"*, Master's Thesis submitted in University of California, Santa Barbara, June, 2006.

[9] Sheikh Mahbub Habib, Cyril Jacob, Tomas Olovsson, "*A Practical Analysis of the Robustness and Stability of the Network Stack in Smartphones*", 11th IEEE International Conference on Computer Science & Information Technology (ICCIT 2008), pp. 393-398. ISBN: 1-4244-2136-7.

[10] Sheikh Mahbub Habib, Cyril Jacob, Tomas Olovsson, "*An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones*", Accepted in Journal of Networks (special issue), Academy Publisher, ISSN: 1796-2056.

[11] Windows Embedded CE, *http://msdn.microsoft.com/en-us/library/bb847932.aspx* , Published on 27th Aug 2008, visited on April, 2009.

[12] Windows Embedded NavReady, *http://msdn.microsoft.com/en-us/library/cc664218.aspx* , Published on 13th June 2008, visited on April 2009.

[13] Windows Embedded Standard 2009, *http://msdn.microsoft.com/en-us/library/ms376738.aspx* , visited on April 2009.

[14] Windows NT Embedded, *http://msdn.microsoft.com/en-us/library/ms950386.aspx* , visited on April 2009.

[15] Windows Embedded for Point of Service, *http://msdn.microsoft.com/en-us/library/bb848016.aspx* . visited on April 2009.

[16] Windows .NET Micro Framework, *http://msdn.microsoft.com/en-us/library/cc533015.aspx* . visited on April 2009.

[17] Board Support Package Overview, *http://msdn.microsoft.com/en-us/library/ms901743.aspx* , visited on April 2009.

[18] ARM BSPs, *http://msdn.microsoft.com/en-us/library/ms901733.aspx* , visited on April 2009.

[19] MIPS BSPs, *http://msdn.microsoft.com/en-us/library/ms902328.aspx* , visited on April 2009.

[20] SHx BSPs, *http://msdn.microsoft.com/en-us/library/ms902369.aspx* , visited on April 2009.

[21] x86 BSPs, *http://msdn.microsoft.com/en-us/library/ms902398.aspx* , visited on April 2009.

[22] Windows Mobile 6, *http://msdn.microsoft.com/en-us/library/bb158486.aspx* , Published on 16th Oct 2008, visited on April 2009.
[23] Windows CE 5.0, *http://msdn.microsoft.com/en-us/library/ms905511.aspx* , visited on April 2009.
[24] Windows Embedded CE Documentation, *http://msdn.microsoft.com/en-us/library/bb159115.aspx* , Published on 27th Aug 2008.
[25] Windows CE 5.0 Kernel Overview, *http://msdn.microsoft.com/en-us/library/aa450566.aspx* , visited on April 2009.
[26] Windows CE 5.0 Memory Architecture, *http://msdn.microsoft.com/en-us/library/aa450572.aspx* , visited on April 2009.
[27] Windows Mobile 6.1 Memory Management Changes, *http://bolingconsulting .com/ blog/?p=4* , published on 6th June 2008.
[28] Internal File System Selection, *http://msdn.microsoft.com/en-us/library/aa910649.aspx* , published on 28th Aug 2008.
[29] CDFS/UDFS File Systems, *http://msdn.microsoft.com/en-us/library/aa914741.aspx* , published on 28th Aug 2008.
[30] Developing a Device Driver, *http://msdn.microsoft.com/en-us/library/ms923714.aspx* , visited April 2009.
[31] Stream Interface Driver Architecture, *http://msdn.microsoft.com/en-us/library/ms894042.aspx* , Visited April 2009.
[32] Windows CE Drivers, *http://msdn.microsoft.com/en-us/library/aa447511.aspx* .
[33] Extensible Authentication Protocol (EAP), *http://www.ietf.org/rfc/rfc3748.txt* ,RFC 3748, June 2004.
[34] Challenge-Handshake Authentication Protocol, *http://msdn.microsoft.com/en-us/library/aa924332.aspx* , published on 28th Aug 2008, visited April 2009.
[35] Transport Level Security, *http://msdn.microsoft.com/en-us/library/aa924160 .aspx* , published on 28th Aug 2008, visited April 2009.
[36] Protected Extensible Authentication Protocol, *http://msdn.microsoft.com/en-us/library/aa921396.aspx* , 28th Aug 2008, visited April 2009.
[37] Microsoft Challenge-Handshake Authentication Protocol 2.0 (MS-CHAP V2), *http://msdn.microsoft.com/en-us/library/aa922194.aspx* , published on 28th Aug 2008, visited April 2009.
[38] Network Drivers, *http://msdn.microsoft.com/en-us/library/ms892542.aspx*, visited April 2009.
[39] Connection of IPv6 Domains via Ipv4 clouds, *http://www.ietf.org/rfc/rfc3056.txt* , RFC 3056, Feb 2001.
[40] Intra-site Automatic Tunnel Addressing Protocol (ISATAP), *http://tools.ietf.org/html/rfc5214* , RFC 5214, March 2008.
[41] Layered Protocols and Provider Chains, *http://msdn.microsoft.com/en-us/library/aa925739.aspx* , published on 28th Aug 2008, visited May 2009
[42] ActiveSync Desktop Pass-through (DTPT), *http://msdn.microsoft.com/en-us/library/aa910850.aspx* , published on 28th Aug 2008, visited may 2009.
[43] CellCore, *http://msdn.microsoft.com/en-us/library/aa921520.aspx* , published on 17th Sept 2008, visited April 2009.
[44] Simultaneous Voice and Data Connections, *http://msdn.microsoft.com/en-us/library/bb416439.aspx* , published on 17th Sept 2008.
[45] Connection Service Providers, *http://msdn.microsoft.com/en-us/library/bb416493.aspx* , published on 17th Sept 2008.
[46] SIP: Session Initiation Protocol, *http://www.ietf.org/rfc/rfc3261.txt*, RFC 3261, June 2002.

[47] DialPlan component, *http://msdn.microsoft.com/en-us/library/aa921954.aspx* , published 28th Sept, 2008.
[48] Real-time Communications (RTC) Client API, *http://msdn.microsoft.com/en-us/library/ms931946.aspx* , visited May 2009.
[49] VOIP Application Interface Layer (VAIL), *http://msdn.microsoft.com/en-us/library/bb416401.aspx* , visited May 2009.
[50] VoIP Provisioning for Windows Mobile Powered Devices, *http://msdn.microsoft.com/en-us/library/bb416401.aspx* , visited May 2009.
[51] Securing L2TP using IPSec, *http://tools.ietf.org/rfc/rfc3193.txt* , Nov 2001.
[52] Point-to-point Tunneling Protocol, *http://www.ietf.org/rfc/rfc2637.txt* , July 2001.
[53] Internet Key Exchange (IKEv2) Protocol, *http://www.ietf.org/rfc/rfc4306.txt* , Dec 2005.
[54] Negotiation of NAT-Traversal in the IKE, *http://www.ietf.org/rfc/rfc3947.txt* , January 2005.
[55] IKEv2 Mobility and Multihoming Protocol, *http://tools.ietf.org/html/rfc4555* , June 2006.
[56] Global System for Mobile Communications (GSM), *http://www.gsmworld.com/technology/gsm/index.htm* , visited May 2009.
[57] General Packet Radio Service (GPRS), *http://www.gsmworld.com/technology/gprs/index.htm* , visited May 2009.
[58] Enhanced Data rates for GSM Evolution, *http://www.gsmworld.com/technology/edge.htm* , visited May 2009.
[59] International Telecommunication Union, *http://www.itu.int/net/home/index.aspx* , visited May 2009.
[60] IMT-2000, *http://www.itu.int/home/imt.html* , visited May 2009.
[61] Wideband Code Division Multiple Access, *http://www.ericsson.com/technology/tech_articles/WCDMA.shtml* , visited May 2009.
[62] 3G CDMA2000, *http://www.cdg.org/technology/3g.asp* , visited May 2009.
[63] HSPA, *http://www.3gpp.org/HSPA* , visited May 2009.
[64] The 3rd Generation Partnership Project, *http://www.3gpp.org/about-3gpp* , visited May 2009.
[65] Jacob Sharony, *"Introduction to Wireless MIMO-Theory and Applications"*, IEEE LI, Nov 2006.
[66] CDMA Technology, *http://www.cdg.org/technology/index.asp* , visited May 2009.
[67] 3G - CDMA2000 1xEV-DO Technologies, *http://www.cdg.org/technology/3g_1xEV-DO.asp* , visited may 2009.
[68] Bluetooth, *http://www.bluetooth.com/Bluetooth/* , visited May 2009.
[69] Core Specification v2.0 + EDR, *http://bluetooth.com/Bluetooth/Technology/Building/Specifications/* , visited May 2009.
[70] Infrared Data Association, *http://www.irda.org/* , visited May 2009.
[71] IEEE Standards Association, IEEE 802.11, *http://standards.ieee.org/getieee802/802.11.html* , visited May 2009.
[72] Introduction to Wi-Fi (802.11), *http://en.kioskea.net/contents/wifi/wifiintro.php3* , visited May 2009.
[73] Andrew Bittau, Mark Handley, Joshua Lackey, *"The Final Nail in WEP's Coffin"*, *http://tapir.cs.ucl.ac.uk/bittau-wep.pdf*, visited May 2009.
[74] SetProcPermissions, *http://msdn.microsoft.com/en-us/library/aa908794.aspx* , published 28th Aug 2008.

[75] Datalight Reliance Return on Investment Analysis Guide, *http://www.datalight.com/resources/* , visited May 2009.
[76] Joshua Wright, "Attacking 802.11 Networks", LightReading Live!,1st Oct 2003.
[77] Problems Created by Man-in-the-Middle Attacks, *http://www.wireless-center.net/Wi-Fi-Security/808.html* , visited April 2009.
[78] Vulnerability in SMB Could Allow Remote Code Execution, *http://www.microsoft.com/technet/security/Bulletin/MS08-068.mspx* , published 11the Nov 2008.
[79] Exploited Systems through ActiveSync, *http://www.informit.com/guides/content.aspx?g=security&seqNum=326* , published 26th Sep 2008.
[80] Alex Talevski, Elisabeth Chang, Tharam Dillon, *"Secure Mobile VoIP"*, pp. 2108-2113. IEEE ICCIT 2007 Proceedings, ISBN: 0-7695-3038-9.
[81] Israel M. Abad Caballero, *"Secure Mobile Voice over IP"*, Master of Science Thesis, June 2003, KTH, Sweden.
[82] GSM Cloning, *http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html* , visited May 2009.
[83] N.J Croft, M.S Olivier, "A Silent SMS Denial of Service (DoS) Attack", University of Pretoria, South Africa, Oct 2007.
[84] Alex Biryukov, Adi Shamir, David Wagner, *"Real Time Cryptanalysis of A5/1 on a PC"*, Fast Software Encryption Workshop 2000,April 2000, Newyork,USA.
[85] Elad Barkan, Eli Biham, and Nathan Keller,*"Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication"*, pp.600-616, Vol. 2729/2003, Lecture Notes in Computer Science, Springer, 2003.
[86] Eli Biham, Orr Dunkelman, Nathan Keller, *"A Related-Key Rectangle Attack on the Full KASUMI"*, pp. 443-461, Vol. 3788/2005, Lecture Notes in Computer Science, Springer, 2005.
[87] Logical Link Control and Adaptation Protocol (L2CAP), *http://www.bluetooth.com/* , visited May 2009.
[88] Link Manager Protocol, *http://www.bluetooth.com/* , visited May 2009.
[89] M. Herfurt, M. Holtmann, A. Laurie, C. Mulliner, T. Hurman, M. Rowe, K. Finisterre, J. Wright. the trifinite group. http://www.trifinite.org , visited April 2009.
[90] Microsoft Bluetooth Stack OBEX Directory Traversal, *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0244* , under review, visited May 2009.
[91] Privileged APIs, *http://msdn.microsoft.com/en-us/library/aa919335.aspx*, published 28th Aug 2008.
[92] Certificate Management, *http://msdn.microsoft.com/en-us/library/bb416317.aspx* , published 28th Aug 2008.
[93] Security Policy Settings, *http://msdn.microsoft.com/en-us/library/aa455966.aspx* , visited April 2009.
[94] Security Roles, *http://msdn.microsoft.com/en-us/library/ms890461.aspx* , visited April, 2009.
[95] Certificate Mgmnt., *http://msdn.microsoft.com/en-us/library/bb416308.aspx* , published 28th Aug 2008.
[96] Flexispy Pro, *http://www.flexispy.com/spyphone-call-interceptor-gps-tracker-windows-mobile.htm* , visited 20th May 2009.
[97] Authentication Events, *http://msdn.microsoft.com/en-us/library/ms926477.aspx* , visited April 2009.

[98] Remote Wipe, *http://msdn.microsoft.com/en-us/library/bb737595.aspx*, published 19 Sept 2008.
[99] Remote Wipe, *http://www.microsoft.com/downloads/details.aspx?familyid=E6851D23-D145-4DBF-A2CC-E0B4C6301453&displaylang=en* ,published 9th Feb 2008.
[100] MS Mobile Device Manager, *http://www.microsoft.com/windowsmobile/en-us/business/solutions/enterprise/mobile-device-manager.mspx* ,visited May 2009.
[101] Cardtrap virus, http://www.f-secure.com/v-descs/cardtrap_a.shtml, visited May 2009.
[102] WinCE/Infojack, *http://www.avertlabs.com/research/blog/index.php/2008/02/26/windows-mobile-trojan-sends-unauthorized-information-and-leaves-device-vulnerable/* , visited May 2009.
[103] P.Haas, *http://www.cs.ucsb.edu/~seclab/projects/smartphones/2005_haas_cellviruses.pdf* , visited May 2009.
[104] Michael Becher, Felix C. Freiling and Boris Leider, *"On the Effort to Create Smartphone Worms in Windows Mobile"* , IEEE proceedings 2007.
[105] Georgia Tech Information Security Center report, *"Emerging Cyber Threats Report for 2009"*, October 2008.
[106] Nan Xu, Fan Zhang, Yisha Luo, Weijia Jia, Dong Xuan and Jin Teng, *"Stealthy Video Capturer: A New Video-based Spyware in 3G Smartphones"*, ACM, Wisec'09, Zurich.
[107] Seth fogie, *"Airscanner Vulnerability Summary: Windows Mobile Security Software Fails the Test"*, Airscanner Corp., 14th Aug 2004.
[108] Aubrey-Derrick Schmidt, Frank Peters, Florian Lamour, *"Monitoring smartphone for anomaly Detection"*, ACM , Mobilware'08, Innsbruck.
[109] Markus Miettinen, Perttu Halonen, Kimmo Hätönen, *"Host-Based Intrusion Detection for advanced mobile devices"*, IEEE, AINA'06.
[110] Michael Becher and Ralf Hund, *"Kernel-Level Interception and Applications on Mobile Devices"*, May 2008.
[111] John Wack, Miles Tracy, Mrurgiah Souppaya, *"Guideline on Network Security Testing"*, NIST Special Publication 800-42, October 2003.
[112] Nmap security scanner, *http://nmap.org/* , Accessed on Feb 2009.
[113] Nessus Vulnerability Scanner, *http://www.nessus.org/nessus/* , released on Feb 2009.
[114] Salvatore Sanfilippo, "Hping", http://www.Hping.org/ . Accessed on Feb 2009.
[115] Dug Song, "Dsniff 2.3", *http://monkey.org/~du gsong/dsniff/* . Accessed on Feb 2009.
[116] Packit (network injection and capture tool), http://www.packetfactory.net/projects/packit/ , Accessed on Feb 2009.
[117] Wireshark, *http://www.wireshark.org/* , Accessed on Feb 2009.
[118] Syn flood, http://www.cert.org/advisories/CA-1996-21. html . Accessed on April 2009.
[119] CVE ID: CVE-2006-0021 "IGMP v3 DoS Vulnerability", *http://nvd.nist.gov/nvd.cfm?cvename=CVE-2006-0021* , National Vulnerability Database, National Institute of Standards and Technology, Published on February 14, 2006.
[120] Alexey Sintsov, "Microsoft Windows XP/2003 (IGMP v3) Denial of Service Exploit (MS06-007)", *http://www.milw0 rm.com/exploits/1599* , Published on March 21, 2006.

[121] Microsoft Security Bulletin MS06-007, http://www.microsoft.com/technet/security/Bulletin/MS06-007.mspx , 14th Feb 2006.
[122] Wolfgang John, Tomas Olovsson: *Detection of malicious traffic on back-bone links via packet header analysis*. Campus-Wide Information Systems, 2008, Volume:25, Issue:5, Page:342 – 358, ISBN/ISSN 1065-0741.
[123] David Needle, "Security Firm Lists Leading VoIP Threats", eSecurity Planet, 16th April 2009.
[124] Nicole Lewis, "VoIP Over 3G Wireless Gets Real", VOIP-News, 28th Jan 2008.
[125] Rachid El Khayari, "SPAM over Internet Telephony and how to deal with it", Diploma Thesis, Fraunhofer Inst. of Secure IT.

APPENDIX A
LIST OF PENETRATION TOOLS

Nmap(Network Mapper) Scanner: It is a free and open source (GNU GPL) utility for network exploration and security auditing by Insecure.com LLC. Nmap is used in the network scanning section of this thesis.

Nessus Vulnerability Scanner: It is a free (Home feed only) and licensed utility by Tenable Network Security for vulnerability analysis in the network level. Nessus (Home feed) is used in the vulnerability scanning section of this thesis.

Wireshark: It is a free and open source (GNU GPL) protocol analyzer tool widely used in industries and educational institutions. In our thesis, Wireshark is used to analyze the related protocols involved in the practical analysis of network stack.

HPing: It is a free and open source (GNU GPL) command-line oriented TCP/IP packet assembler/analyzer. HPing is used for TCP Syn flooding and Land attack in our thesis.

DSniff: It is a collection of tools for network auditing and penetration testing by Dug Song. We used arpspoof tool of DSniff package for ARP spoofing attacks.

Packit: It is an auditing tool that allows penetration testers to monitor, manipulate, and inject customized IP traffic in the network. Some ARP and RARP (reverse ARP) spoofing attacks are done with this tool.

Netwox (Network Toolbox): It is a free and open source (GNU GPL) collection of 222 tools used for security auditing. In our thesis, IPv6 attack like neighbor discovery spoofing is done using this tool.

APPENDIX B
IGMPV3 EXPLOIT

IGMPv3 exploit is one of the well-known vulnerabilities in desktop operating systems. Same attack has already caused vulnerability in Windows Mobile 6.1 powered smartphones. Alexey Sintsov published the exploit code on March 22, 2006, where the bug was introduced in the IP header.

```
//Ip options
        ipHeader.options[0]=htons(0x0000); //bug is here
        ipHeader.options[1]=htons(0x<0000);


        igmpHeader.type=0x11; //v3 Membership Query
        igmpHeader.code=5;
        igmpHeader.num=htons(1);
        igmpHeader.ResvSQVR=0x0;
        igmpHeader.QQIC=0;
       igmpHeader.group=inet_addr("0.0.0.0");
        igmpHeader.addes=dst_addr;

        igmpHeader.checksum=0;
```

Figure B.1: Partial IGMPv3 Exploit Code.

If we investigate the code given in Figure 1, the bug is seen in the IP Header's options field. Network stack of target device cannot process that particular buggy packet and get frozen permanently until it has been reset by the user. When the packet is analyzed by a packet analyzer, source or target none complained it as a malformed packet. It is shown as a normal IGMP V3 membership query packet.