DAI-Labor
TU Berlin

Malicious Software for Smartphones

Technical Report: TUB-DAI 02/08-01

Aubrey-Derrick Schmidt and Sahin Albayrak

February 10, 2008

CC ACT
Agent Core Technologies

CC COG
Cognitive Architectures

CC IRML
Information Retrieval
& Machine Learning

CC NEMO
Network & Mobility

CC NGS
Next Generation Services

CC SEC
Security

DAI-Labor der Technischen Universität Berlin
Prof. Dr.-Ing. habil. Sahin Albayrak
Technische Universität Berlin / DAI-Labor
Institut für Wirtschaftsinformatik und Quantitative Methoden
Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation
Sekretariat TEL 14
Ernst-Reuter-Platz 7
10587 Berlin
Telefon: (030) 314 74000
Telefax: (030) 314 74003
E-mail: Sekretariat@dai-labor.de
WWW: `http://www.dai-labor.de`

Disclaimer

This work was funded by Deutsche Telekom Laboratories.

Abstract

Smartphones are ubiquitous devices that represent the next generation of mobile phones. They combine mobility with high computational power and provide standardized operating systems allowing 3rd party software development. The problem about this is that besides normal software so called malicious software (malware) can be developed using the same development tools and application programming interfaces (APIs). Such malware targeting smarpthones can cause serious damage and costs where the first smartphone malware appeared in June 2004. Since then, the amount of new emerging malware raised steadily.

In order to show that smarphone malware is highly underestimated we describe smartphones characteristics and how malwares are affecting these. We present a list of the most common behavior patterns and investigate possibilities how to exploit the given standard Symbian OS API for additional malware functionalities. The corresponding results are presented and example malwares are given. Additionaly, we introduce a scenario basing on the same results which underlines the danger of smarpthones malware.

Contents

List of Figures

List of Tables

1 Introduction

More than 80 million cellular phones are registered in Germany since 1. August 2006 [1], which statistically means, that every registered inhabitant of Germany owns a mobile phone. Together with the steadily increasing capabilities, ordinary cellular phones get more and more similar to current smartphones, which are wide spread mobile devices with high computational power and sophisticated functionalities. Due to this evolution, it is imaginable that in the near future most mobile phones can be categorized as smartphones. These phones provide standardized operating systems with development environments for 3rd party software. This allows the creation of, e.g. business application and games, but also the development malicious software (malware).

Malware, e.g. like virus, worms and Trojan horses have been threats to computer systems for many years, so, it was only a question of time when malicious software writers would get interested in mobile platforms, such as Symbian OS or Windows Mobile. In 2004, the first articles about malware for mobile phones [2], [3] appeared saying that the next generation of targets are mobile devices like smartphones. Since then, the number of malwares increased every month, and variants for various platforms appeared.

Commercially available countermeasures to smartphone malware are not sufficient since they rely on signature lists or static rules. This leaves more and more people exposed to these kinds of threats, where the group of possible victims includes children, youths, and older persons besides professionals. Most of these types of users, even most of the professional users, cannot be aware of the danger coming from these devices, due to the lack of experience or training. This makes these groups attractive to attackers, as it may be easier convincing such a person to install a foreign program than convincing an experienced computer user.

Despite this situation, the danger of smartphone malware is still highly underestimated. Hence, the goal of this work is to raise awareness when using and working with such devices. Therefore, in Section 2, we start with explaining, what a smartphone actually is, how it is used, and which information can be found on it in order to understand possible threats. This is followed by the main part referring to *mobile* malware and its effects on smartphones. Future malware is described and we give an example how such a malware could work in real life. We briefly describe well-known countermeasures and then point to related smartphone malware research in Section 4. In Section 5, we finally conclude.

2 Smartphones

For better understanding the impact of malwares on smartphones, we examine the characteristics of these types of cellular phones in this section. Therefore, we introduce important smartphone-related topics, especially which Operating Systems (OSs) are available, how software is developed, which applications are mostly used, and finally what kind of data and information are mostly stored on them.

2.1 Characteristics

Mobile phones can be basically distinguished into two categories: *feature phones* and *smartphones*. A feature phone represents the standard cellular phone with all of its functionality, e.g. calendar, messaging or often support for Java Platform Micro Edition (Java ME) software. A Smartphone whereas mostly unifies functionalities of a cellular phone, a PDA, an audio player, a digital camera, a GPS receiver and a PC. It often uses PC-like *QWERTY* keyboards in order to increase typing speed and sometimes PDA-like pen displays for improved data and command handling. One significant characteristic is the ability of installing proprietary third party application. This presumes the existence of a publicly available Software Developing Kit (SDK). Smartphones use various techniques for creating wireless connections, which represents their most essential purpose:

- GSM[1] is used for voice calls and services like SMS[2].
- GPRS[3] provides voice and packet data communication.
- W-CDMA (UMTS[4]) is able to transport data at higher speed than GSM.

Additionally, the devices provide Bluetooth (BT), Wireless LAN (WLAN) or IrDA[5] support for shorter range wireless connectivity. Using one of these connections, a user is able to make phone calls, use an internet browser, play multi-player games, or read emails.

Furthermore, the smartphone can be seen as the first platform for *pervasive computing* [4], where interesting areas of application are pointed out by Roussos et al. [5]:

- Information service endpoint, e.g. applied as navigational assistance or location-based services.
- Remote controllers for devices, like television or HiFi station.
- Pervasive network hubs to provide wide area connectivity, e.g. for wearable systems that need to communicate in order to transmit health related data.
- ID tokens in order to store information used to provide accountability.

2.2 Operating Systems for Smartphones

Most mobile phones use proprietary OSs, which has the disadvantage, that only few or even none additional software is available, e.g. SonyEricsson K850i[6]. On most smartphones this disadvantage does not exist, as they mostly use one of the following OSs that enable 3rd party software installation. Following Canalys [6], in the third quarter of 2006 the three main competitors in this field (Figure 1 and Figure 2) were Symbian

[1]Global System for Mobile Communications
[2]Short Message Service
[3]General Packet Radio Service
[4]Universal Mobile Telecommunications System
[5]Infrared Data Association
[6]`http://www.sonyericsson.com/cws/products/mobilephones/overview/k850i`

OS from Symbian Ltd. [7], Windows Mobile from Microsoft [8] and Blackberry from Research In Motion (RIM) with 78.7%, 16.9% and 3.5% market share on the smart mobile device market, respectivly. These operating systems allow the development and installation of third party applications for customizing the device according to the software needs of the user. Additional system are just emerging, e.g. Google Android[7] and the Apple iPhone[8]. Both systems can have a serious impact on the smart mobile device market as soon as Google releases its' first Android devices and Apple its SDK for the iPhone.

EMEA total smart mobile device market
Market shares Q3 2006, Q3 2005

OS vendor	Q3 2006 shipments	% share	Q3 2005 shipments	% share	Growth Q3'06/Q3'05
Total	**7,319,690**	**100.0%**	**6,552,850**	**100.0%**	**11.7%**
Symbian	5,757,540	78.7%	5,022,710	76.6%	14.6%
Microsoft	1,235,130	16.9%	1,179,530	18.0%	4.7%
RIM	253,420	3.5%	230,190	3.5%	10.1%
Others	73,600	1.0%	120,420	1.8%	-38.9%

Source: Canalys estimates, © canalys.com ltd. 2005-2006
Smart mobile device market: handhelds, wireless handhelds, smart phones

Figure 1: Canalys Analysis Q3 2006 [6]

2.2.1 BlackBerry (RIM)

Research In Motion (RIM) provides proprietary operating systems for its own BlackBerry devices. The target customers are business managers as the included techniques and services are intended to support their daily needs [9]. Although RIM only address a small customer group, they have got a recognizable share with their OS in the smart mobile device market.

Since only proprietary and restricted Java application are allowed on BlackBerry devices at the moment, only few official documents can be found about palm operating system or application security. Most security information address certain parts relating to the BlackBerry PUSH technology, which represents a mechanism for instantly bringing email messages to mobile devices.

2.2.2 Palm OS

Palm OS from PalmSource was originally released in 1996, previously, it was developed for the "Pilot PDA" from US Robotics. There were several Palm OS versions until Palm OS 5 and 6. These were developed parallel as Palm OS 5 "Cobalt" aimed for a certain

[7] `http://code.google.com/android/`
[8] `http://www.apple.com/iphone/`

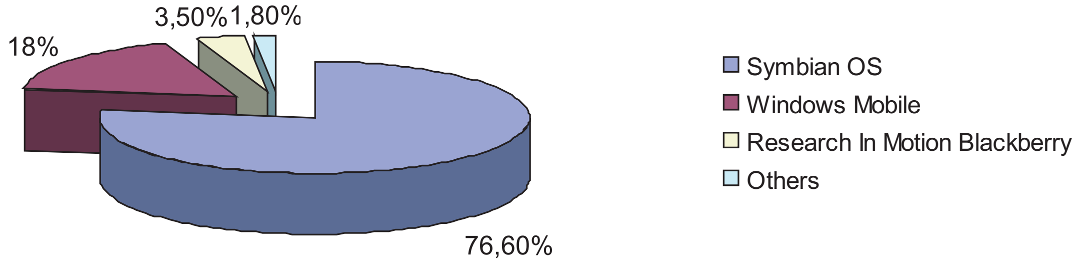

Figure 2: Smartphone market Q3 2006 basing on a Canalys analysis [6]

kind of PDAs[9] and Palm OS 6 for smartphones and integrated wireless handhelds[10]. In September 2005, PalmSource announced that it was acquired by ACCESS calling the current OS variant "Garnet OS".

As it can be seen on Figure 3, ARCchart estimations show over 20000 available applications for Palm OS [10] which makes this platform a plausible target for attackers. Security measures preventing this are, e.g. application certificates and application signing which control access to restrictive application programming interfaces (APIs).

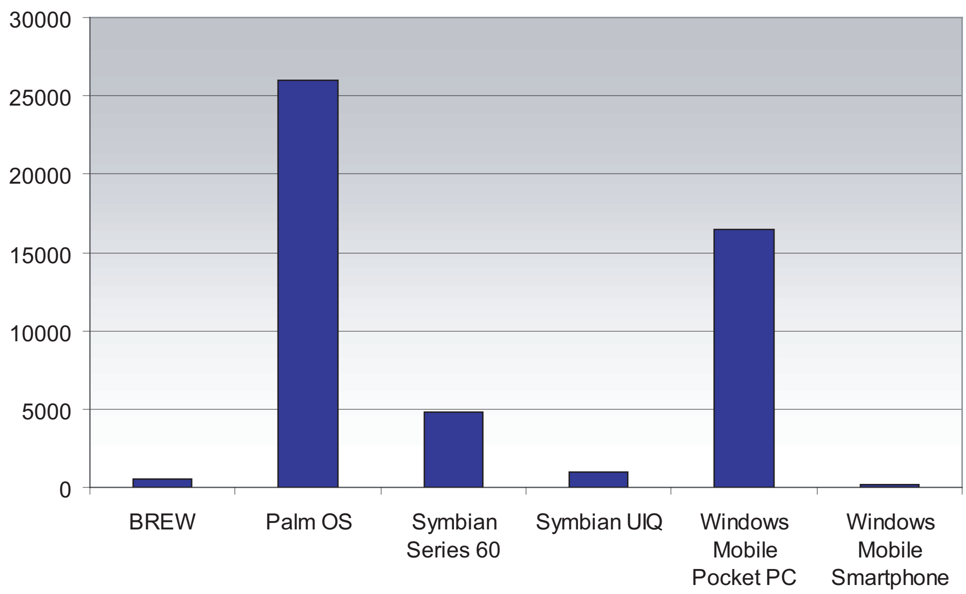

Figure 3: Number of applications available for each mobile platform in 2005 according to [11]

[9]Native 32-bit, ARM-4T-based architecture, 16-bit ARM Thumb support, ARM 7TDMI based microprocessors

[10]e.g. WiFi 802.11b and Bluetooth

2.2.3 Windows Mobile

The Windows Mobile operating system is based on Windows CE and was developed for mobile devices like PocketPCs, PDAs, smartphones, and embedded systems, e.g. smart fridges [8]. In 2003 there were three different version of Windows Mobile: Windows Mobile 2003 for Pocket PC, Windows Mobile 2003 for Pocket PC Phone Edition and Windows Mobile 2003 for Smartphones. The current version Windows Mobile 5.0, called "Magneto", was released in the year 2005. It uses the .NET Compact Framework 1.0 SP2 to enable .Net applications and bases on Windows CE 5.0. Windows Mobile supports WIFI, GPS, PC (Windows) synchronization.

Windows Mobile security addresses three major approaches: *security roles*, *security policies*, and *application signing*. Security roles define users or groups having predefined rights on a device. The most privileged role allows changing security policies, which are rules permitting certain actions on the device, e.g. installing and running unsigned applications[11]. Application signing principles of Windows Mobile are very similar to the ones we will present in the Symbian OS part more detailed. Basically, Windows Mobile software is signed in order to permit access to sensitive APIs[12].

2.2.4 Symbian

Symbian Limited is a software producing company located in London, UK. It is owned by Ericsson (15.6%), Nokia (47.9%), Panasonic (10.5%), Samsung (4.5%), Siemens (8.4%) and Sony Ericsson (13.1%). Symbian core business is developing and licensing Symbian OS, an operating system for mobile devices which has evolved from former Psion's Epoc. The Symbian OS licensees represent over three quarters of mobile phone shipments globally [7]. Smartphone manufacturers that license Symbian OS are Arima, Ben Q, Fujitsu, Lenovo, LG Electronics, Motorola, Mitsubishi, Nokia, Panasonic, Samsung, Sharp and Sony Ericsson.

Symbian OS introduces three security concepts, which are *capabilities*, *installation file signing*, and *data-caging*. Capabilities limit access to sensitive APIs. There are basically three levels of limitation where on the highest level full device and network access is granted to the corresponding application[13]. These limitation levels are defined by certificates that are used to sign Symbian OS Installation System (.SIS) files. Without a valid signing, it is not possible to install application on Symbian OS devices[14]. Data-caging extends this approach as it limits access to the file system. Depending on the limitation coming from the certificate, application can only write to certain areas, like the application folder, user data folder, or system folder[15].

[11] more information at `http://msdn2.microsoft.com/en-us/library/aa455966.aspx`
[12] Security policies can make application signing unimportant
[13] `http://developer.symbian.com/main/downloads/files/Symbian_Signed_Grid.pdf`
[14] S60 3rd
[15] `http://wiki.forum.nokia.com/index.php/Data_caging`

2.2.5 Google Android

Google Android[16] is a package of software that includes an operating system, middleware and basic applications. The Android system is built upon the Linux 2.6 Kernel and supports most of original provided functionalities. Android treats every application equal, which means on the one hand that a developer is able to replace every single android program. On the other hand it means that if you develop an android application it can be run on any Android device, only being limited by the provided functionalities[17].

The security mechanisms that can be found in Google Android basically are based on the same mechanisms that can be found in the linux system. Especially access control, e.g. user and group IDs, is managed where every installed application gets its own user ID with its' specific permissions. These permissions allow finer-grained access adjustment for processes using certain functionalities, e.g. sending SMS messages or dialing a phone call.

2.2.6 Apple Mac OS X for iPhone

The Apple iPhone[18] is a very interesting device that can be classified as smartphone since Steve Jobs from Apple announced a SDK for it[19]. The device runs a modified version of Mac OS X and includes several applications, e.g. the Safari browser, a music player, and digital camera. As long as the SDK is not officially released, only little can be said about security approaches used in this version of the OS. It is imaginable that they are similar to the ones of the original version with a focus on mobility-related measures.

2.2.7 Java Platform Micro Edition

The Java Platform Micro Edition is no stand-alone operating system, but since most of today's cellular phones support it and hence allow Java ME application installation, it will be described in this part of this work. Java ME is upward compatible to J2SE and J2EE and supports dynamic downloading. It enables cross-device compatibility and has similar capabilities as modern object oriented programming language. There is a large developer community and the possibility of using already existing sources for several different purposes, e.g. web services or communication. Java ME introduces three OS levels [12], each with separate security approaches, where they structured as follows: runtime (bottom), configuration, and profile (top).

On *runtime* level, Java ME uses a virtual machine known as KVM[20], which executes precompiled byte code. This feature enables Java to run built code on almost every system compatible to the libraries used for the build. The KVM is optimized for less memory consumption than native code. It is the pendant to the bigger "brother" JVM - Java Virtual Machine.

---

[16] `http://code.google.com/android/`
[17] Internet tablet and navigation system
[18] `http://www.apple.com/iphone/`
[19] `http://www.macworld.com/article/60555/2007/10/iphonesdk.html`
[20] Kilobyte Virtual Machine, depends on device type

An example *configuration*, the Connected Limited Device Configuration (CLDC), contains the KVM and several core device libraries. It was designed for small, resource constrained devices with 160-512 Kbytes Memory.

Java ME *profiles* have a small size so they fit in almost every compatible current low-end phone. The Mobile Information Device Profile (MIDP) enables sandbox security and it also introduces digital signatures and rights categories.

Java ME applies security mechanisms to all of these levels. At runtime level code is run in a "sandbox" environment and verification system checks applications and classes for failures. At configuration level several approaches were chosen. Depending on the profile, various limitations are given, signed classes are introduced, or even the complete Java SE security package is provided. At profile layer application signing is demanded, which means that developers are assigned to protection domains These protection domains are: *manufacturer*, *operator*, *identified third party*, and *unidentified third party*. Each domain allows access to more or less critical API calls, where the calls are *allowed*, *user permitted*, or *denied*. If a call is allowed, it can be used without restrictions, whereas a user permitted call asks the user, whether he wants the specific application to do a certain action or not[21]. Unsigned application have restricted capabilities and need a user permission for most calls, where it is possible to deactivate the the permission request in the corresponding application manager.

2.3 Software Development for Smartphones

Developing, building, and testing smartphone software requires tools which are often included in a SDK or integrated development environment (IDE). These software *packages* will be briefly introduced in this section.

2.3.1 The SDK

A Software Developing Kit (SDK) is a collection of software that gives a software developer the ability to create and deploy software for a certain framework, platform, operating system, programming language or hardware. Most SDKs can be downloaded for free from the internet. Current examples are the *Symbian S60 3rd SDK*[22], the *Windows Mobile 5.0 SDK*[23], or the *SUN Wireless Toolkit (WTK)*[24]. Most SDKs are delivered with a software emulator.

2.3.2 The (Software-) Emulator

A software emulator gives a developer the ability to run and test software on his computer though is is developed for other systems, e.g. Symbian OS devices. This can reduce costs, as prototypes can already be programed and run without buying a real device. On the other hand, the emulator often does not support all functionalities[25] of a real

[21] various time spans can be chosen similar to cookie handling on PCs
[22] `http://www.forum.nokia.com/main/platforms/s60/`
[23] `http://msdn.microsoft.com/windowsmobile/`
[24] `http://java.sun.com/products/sjwtoolkit/`
[25] e.g. connectivity like GSM

device. This can lead to serious problems. Unlike a simulator, which reproduces programs behavior, an emulation attempts to reproduce the same states the original devices would enter at several points, but regarding current SDKs, the so called software emulators *simulate* connections, interface and functionality through mapping e.g. simulator Bluetooth port to PCs serial port.

Comparable software is available with the VMware products[26] It enables parallel installation and simultaneous running of different operating systems like Windows or Linux.

2.3.3 The IDE

The integrated development environment is very similar to SDKs, it often combines most of the tools to be able to write, compile, build and debug software. The main difference is, that today's IDEs integrate all tools into one single tool that has Graphical User Interface (GUI) especially developed for certain platforms. All needed actions then can be done through the GUI, which often speeds up development. Examples for IDEs for mobile devices are MS Visual Studio, Metroworks Codewarrior, Nokia Carbide, and Eclipse.

2.4 Smartphone Usage

Table 1: TNS GTI 2005 top ten applications/services

No.	Application	Usage
1.	SMS	83%
2.	Games	61%
3.	Camera	49%
4.	MMS Picture	46%
5.	PDA Functions	36%
6.	Internet	31%
7.	WAP	30%
8.	Bluetooth	28%
9.	Email	27%
10.	Video Camera	27%

In order to understand which applications and information can be found on a smartphone, we refer two surveys [13] and [14] which base on participants using mobile devices, like a laptop, a cellular or smartphone, or a PDA. In Global Technology Insight 2005 [13], TNS Technology identified the most used applications in 2005. The results base on data coming from 6807 people using a mobile phone (6517 persons), PDA or laptop and accessed the internet at least once a week. The study partly focused on the adaption of technology applications on mobile devices, which we used to excerpt the top ten applications. Table 1 shows the extracted top ten.

[26] `http://www.vmware.com/`

Another survey from InfoWatch and Zoom.CNews in 2007 [14] bases on information coming from 1500 people using mobile device, like laptops (46.3%), smartphones (45.8%), Pocket computers (26.6%), and other devices (30.4%)[27]. Figure 4 shows the most common areas for which mobile devices are used. Most of the user kept contact records on their devices (77.7%). Email checking (70.8%) and Internet usage (63.8%) were further important fields of application. More than a third of the respondents used calendar/diary-applications to organize their schedule (33.5%) and almost a quarter were shopping online (23.4%).

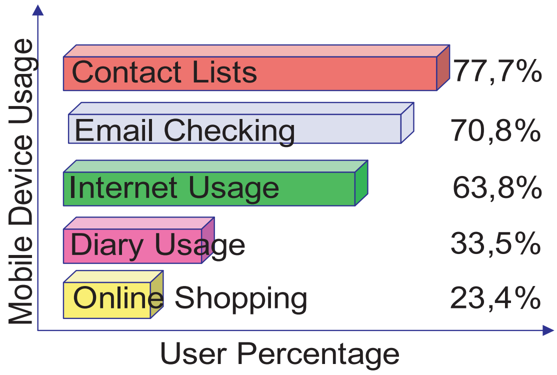

Figure 4: Mobile device usage [14].

The survey additionally showed, what type of data is actually stored on a mobile device. Only 29.3% classified their data as "Nothing important", 68.1% of the respondents kept personal and private data on the device, 16.5% confidential corporate documents, 14.9% intellectual property of the employing company, and 12.2% stored private client and partner data. The resulting chart can be seen on Figure 5.

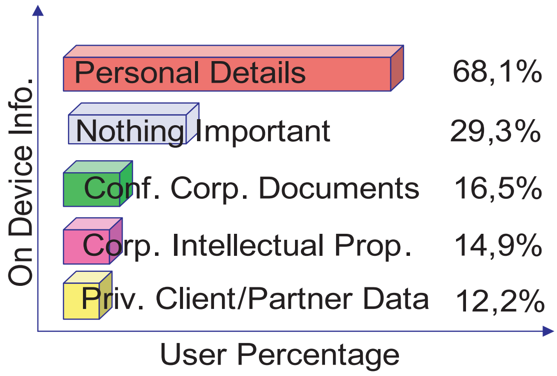

Figure 5: Data stored on a mobile device [14].

Referring to these Charts, a loss or theft of the mobile device will probably be followed by the leakage of personal and private information which can have a serious impact on a person's life[28].

But not only theft and loss threaten users of mobile devices. As these devices have standardized OSs and various programmable interfaces, virus writers and black hat

[27]The total percentage exceeds 100%, as multiple answers were allowed.
[28]e.g. bank account + PIN or pictures of persons in awkward situations

hackers can use these to attack the OS or device in order to gain control or information. This may be done by using malicious software (malware) such as viruses, worms or Trojan horses.

3 Malicious Software for Smartphones

In this section we discuss the danger of malware for smartphones. Therefore, we give a brief introduction to *common* malware principles. We present threats ragarding smartphones and review the history of smartphone malware. Additionally, we give an outlook on how smartphone malware may evolve, and present possible countermeasures.

3.1 Malware Basics

Malware is a portmanteau of the two words *mal*icious and soft*ware*, which clearly indicates that malware is a computer program with malicious intentions. In order to un-

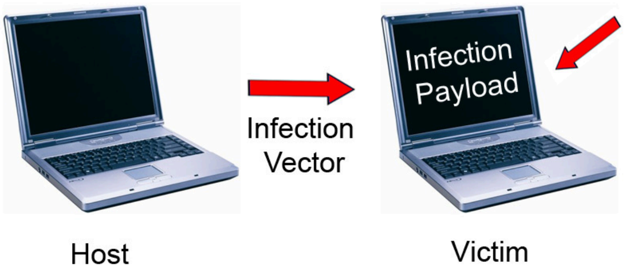

Figure 6: Infection vector and payload

derstand what these malicious intentions actually are, we introduce the terms: *infection vector* and *infection payload*. The infection vector describes which techniques are used to distribute the malicious application. Several approaches are known: e.g. file injection, file transport, exploit[29], or boot sector corruption. The infection payload represents the actual harmful code that is run on the victims machine. There are several possibilities for this[30], e.g. deleting files, denying service, or logging keystrokes. A basic drawing of these malware functionalities can be found on Figure 6

There are three main categories of malicious software: *virus*, *worm*, and *Trojan horses*. A *virus* mostly comes in a hosting file that can be, e.g. an executable. If the user executes this file the virus processes its' malicious commands, which can be almost everything the OS allows. A *worm* can often spread without user interaction. Once started, it searches for infectable victims in range. If a victim is found, it uses an exploit to attach itself to the victim and then repeats this behavior. Sometimes worms drop other malware that can be backdoors that allow remote access. *Bot* programs installed that way can make the victim to a remote triggered Denial of Service (DoS) attacker. A *Trojan horse* is a program that is disguised, e.g. as a popular application, in order to pursue a user to execute or install it. This is done by choosing a well-known name

[29] uses some kind of hardware, software, service, or protocol weakness
[30] We suggest [15] for further readings.

Table 2: Common virus, worm, and Trojan horse characteristics

	Appearance	User Interact.	Vector	Payload
Virus	needs a hosting medium	usually needed	file transport, file injection, exploit	several, e.g. replication, modification
Worm	independent program	usually not needed	exploit	several, e.g. replication, remote access
Trojan Horse	malicious functionalities disguised	usually needed	file transport, exploits	several, e.g. remote access, destructive functionalities

like from a popular game and placing the malware for download on a web page or file sharing tool. Using Table 1, additional popular application categories can be found that help to convince the victim to execute the malware.

These malwares[31] present the most common categories of malicious software. On Table 2 a short overview on these is given. Several further categories can be found in [15]. Malware can be propagated using several techniques and communication interfaces, ranging from an exploit to using social engineering. Regarding smarpthones, the most used infection mediums are Bluetooth, Internet, MMS, Memory Card, and USB, which can also be seen on Figure 7.

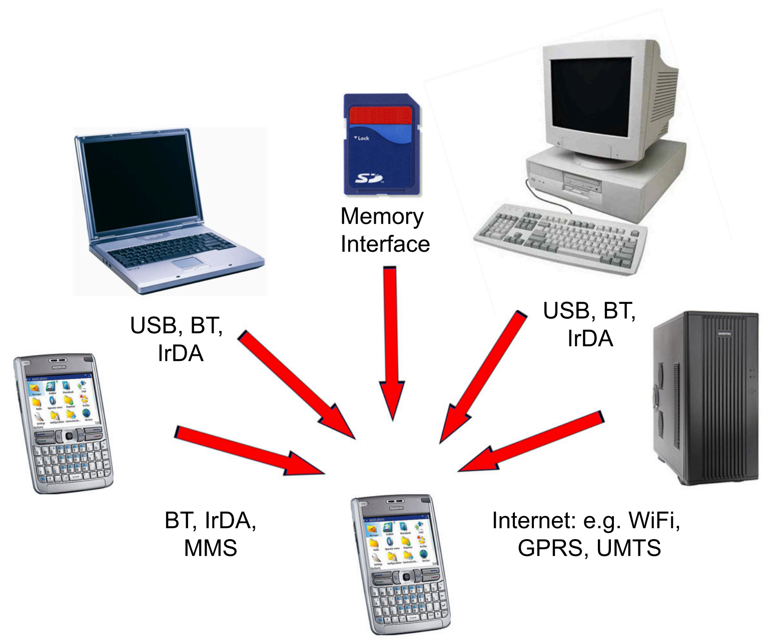

Figure 7: Smartphone malware propagation

[31] applicable to most computer system

3.2 Smartphone Threats

A smartphone is threatened in several ways: e.g. it can be stolen, mechanically damaged, or manipulated. In this section we will discuss the threats concerning smartphones whether under foreign physical or software control.

In comparison to ordinary feature phones, smartphones have a standardized operating system, often with available SDK, more computing power, and *more* programmable interfaces. Figure 8[32] shows a simplified view on the most common smartphone interfaces which can be accessed through libraries included in the SDKs.

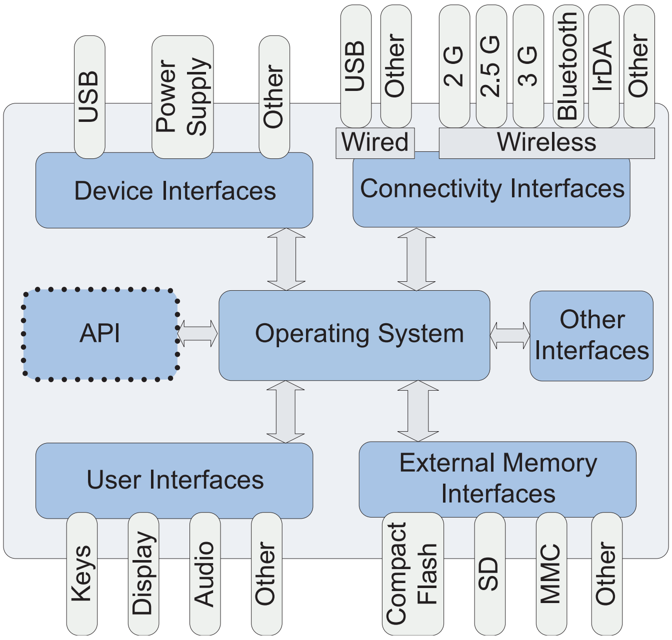

Figure 8: A simplified view on the most common smartphone interfaces

The shown *external* interfaces are sorted by their parent interfaces, which are *Device Interfaces*, *Connectivity Interfaces*, *User Interfaces*, and *External Memory Interfaces*. Each external interface faces various threats[33], not only if the device is stolen, but also the ones shown below and on Table 3.

• The *Device Interfaces* include external interfaces, like e.g. the mini USB interface, power supply, or headphone interface. These can be attacked by destroying the interface mechanically or by giving not specified inputs, like a too high voltage, to them.

• The *Connectivity Interfaces* include wired and wireless communication interfaces. They have implementation for the 2nd, 2.5th, and 3rd generation (2G, 2.5G, 3G) of mobile communication with GSM, GPRS, and UMTS as examples, respectively. Over-Air communication cannot guarantee confidentiality, since attacks can come from any place in radio range. This allows various kinds of attacks.

• The *User Interfaces* consist of the interfaces related to user in- and output. Most attacks of PCs are applicable here, for instance people can misuse unattended foreign devices for dialing premium numbers, altering data, deny incoming calls, or read and fake information.

[32] extended from [16]

[33] See the Appendix A for more detailed descriptions.

Table 3: Smartphone interfaces and the threats through them

	Masquerading	Eavesdropping	Auth. Violation	Loss or Mod.	Repudiation	Forgery	Sabotage
Device							•
Connectivity	•	•	•	•	•	•	•
User	•	•	•		•	•	•
External Memory	•						•
API	•	•	•	•	•	•	•

• The *External Memory Interfaces* can be used to insert infected or corrupted memory cartridges in order to distribute malware and run attacks. Common card formats are Compact Flash, Secure Digital (SD) Memory Card, and Multimedia Card (MMC).

But not only the external interfaces are threatened by attackers, internal interfaces like the OS's application programming interfaces (APIs) can be used to program malicious code that is able to control most of the device's features[34]. And as most of the available APIs normally can be accessed by ordinary developers[35], the danger of smartphone malware is very high. Usually, APIs are provided for: file system, SMS, MMS, email, OS, audio, camera, phone call, and several other tasks.

3.3 Smartphone Malware Review

Following Peter Gostev from Kaspersky Lab. [17], looking from the first appearance of smartphone malware in June 2004 until August 2006, 31 families[36] of malware for mobile devices with 170 variants were recognized. Most of this time, Symbian OS stood in the focus of malware writers, but malicious software for Windows Mobile and Java ME appeared as well[37]. Comparing these numbers with the entries in the Symantec Virus database, we identified, that Symantec listed 245 malwares with detailed description until August 2006, which is a significant greater number. But most of the anti virus companies do not offer enough information and the reason for this cannot be resolved at this point. [38]

Figure 9 displays the extracted information from the Symantec virus database enriched with information coming from F-Secure and Kaspersky. The graph starts in June

[34] e.g. key-Logger, bot-clients, and dialer
[35] Symbian OS 3rd limits this by certificates
[36] A family includes all malwares, basing on the same basic code
[37] PDA malware for Palm is not regarded in this work, although its appearance in the year 2000 can be seen as the appearance of the first malware targeting mobile devices
[38] Additionally, Symantec lists 76 malwares, which do not have any descriptions available, not even from other well-known anti virus companies, e.g. like Kaspersky, McAfee, Norton, Symantec, Sophos, and similar. Examples for this are *Commwarrior.U* and *Locknut.D*.

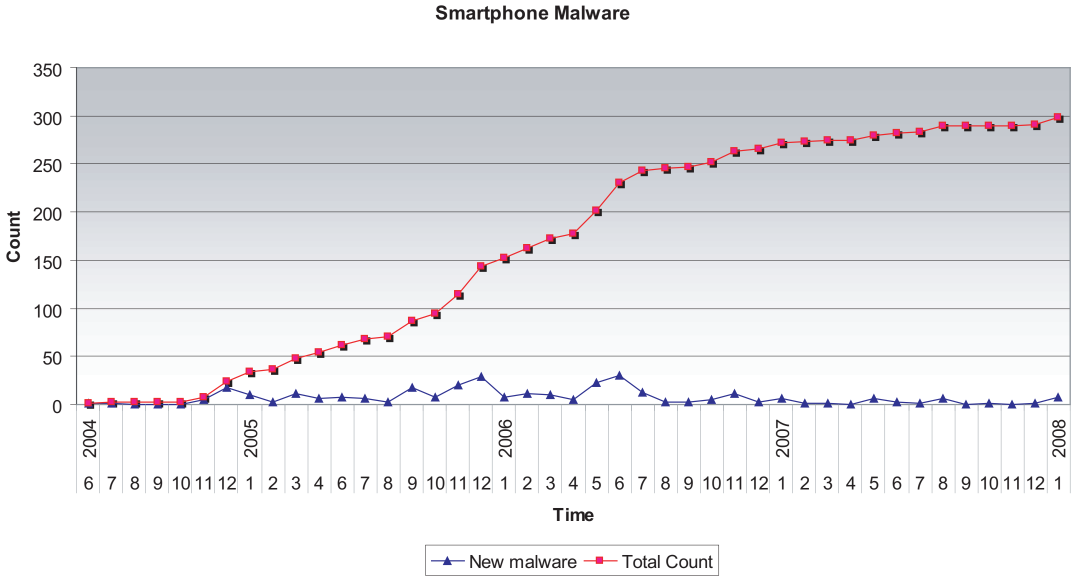

Figure 9: Smartphone malware count

2004 and ends in January 2008, the last published appearance of smartphone malware. The highest peaks relating to new emerging malwares can be found at the end of 2005 and in the middle of 2006, where 49 and 53 new malwares were identified, respectively.

It is interesting that only few platforms were targeted during the that time. And as most malware targeted Symbian OS (280 malwares), only 5 Windows Mobile and 2 Java ME malwares were recognized. But this does not mean that their payload was less dangerous. It included remote access, file deletion, and abuse of the SMS in order to cause high costs.

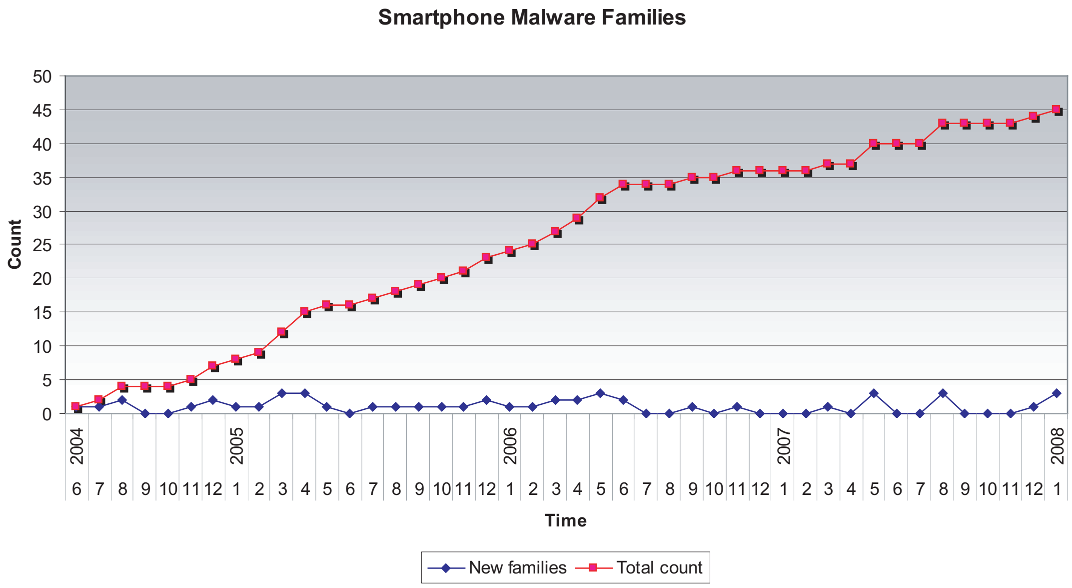

Figure 10: Number of smartphone malware families

On Figure 10 you can see how many malware families emerged by now. As presented on the chart, the growth of families was almost linear which shows that only few people with malicious intentions were familiar to smartphone programming at this time. Nonetheless, there was a recognizable growth which started with *Caribe*, the first worm to hit the mobile community (Symbian OS). *Caribe* first appeared in June 2004 [17] when a professional virus writing group called “29A” released a worm in order to show that “there were new, previously unexplored infection vectors”. Kaspersky Lab named this malware “Worm.SymbOS.Cabir” where Cabir checks autonomously for Bluetooth devices in range and establishes a connection to devices with Bluetooth activated together with “open for communication” setting in order to transmit an infected Symbian OS Installation System (SIS) file. If the user executes the .SIS file, Cabir installs and repeats the same behavior on the newly infected device.

In general, the appearance of new malware families for smartphones was almost linear, about one new family per month[39] and mostly concentrated on Symbian OS [18]. The payload of these malwares can be reduced to the ones that can be found on Table 4 where file manipulation is the most used malicious operation in smartphone malware. More over, there are 47 malwares that do not even have a payload, since their only function is to propagate. Table 4 refers to Table 14 in the Appendix. The total count of entries in this table exceeds the total count of malwares, as a malicious program is not limited to one payload.

Table 4: Smartphone malware payload

Payload	Quantity	Example
Manipulate files	155	Trojan.SymbOS.Skulls.A
Disable applications	125	Trojan.SymbOS.Drever.A
Drop	84	Trojan.SymbOS.CDropper.A
None	47	Worm.SymbOS.Cabir.A
Disable device	45	Trojan.SymbOS.Doomboot.A
Infect Memory Card	32	Trojan.SymbOS.Cardtrap.A
Access private Information	12	Worm.SymbOS.Commwarrior.G
Send private information	8	Trojan.SymbOS.PBStealer.A
Abuse Messaging	7	Trojan.J2ME.RedBrowser.A
Boot loop	2	Trojan.SymbOS.Romride.J
Lock Memory Card	1	Trojan.SymbOS.CardBlock.A
Backdoor	1	Trojan.WinCE.Brador.A

In order to propagate, these malwares used various strategies. The simplest variant was to place a disguised installation file on a webpage or filesharing network. This is possible for most smarpthone malwares as many of them come with an installation file. But other more comprehensive methods were developed. Using Bluetooth and MMS messages were the most famous approaches in order to replicate. All other approaches occurred very rare, e.g. two malwares send SMS messages including a download link pointing to their installation files. A summary of the propagation methods can be seen on Table 5. More detailed descriptions of the malwares can be found on Table 14 in

[39] June 2004 to January 2008 (43 Months) with 45 new families

Appendix and on the web pages of Symantec[40], F-Secure [41], and Kaspersky Lab [42].

Table 5: Smarpthone malware infection vector

Medium	Quantity
Bluetooth	55
MMS	20
Memory card	3
File injection	2
Sent download link	2

With the introduction of application signing in Symbian OS[43] things changed in the field of Symbian OS malware. In the signing process, a trusted Symbian partner checks the complete source code and binaries for meeting certain criteria, like being free from memory leaks and abusive methods. If the check is successful, the application gets signed with an certificate[44] and stays clearly identifiable through a given unique ID. This signing gets mandatory for the current Symbian S60 *3rd* Version which is installed on most Nokia smartphones since the beginning of 2006. By now, no malware for this Symbian version appeared to the knowledge of the author. Every *new* Symbian malware addressed the older Symbian S60 *1st* and *2nd* version.

Table 6: Malware count for mobile platforms

Platform	Count
Symbian OS pre 3rd	280
Windows Mobile	5
Palm OS	4 PDA[45]
Java ME	2
RIM	0

But this does not mean that new Symbian Devices are not threatened any more. A company managed to get an application signed [19], which Symantec categorizes as spyware[46] [20]. This spyware is able to read and send sensitive and private information extracted from an *infected* device. It requires the buyer of the spyware to install the corresponding file to a Symbian OS, Windows Mobile or Blackberry device. After installation, the spyware sends the extracted information to a remote server that stores this data and makes it accessible to the buyer via web interface. Furthermore, the on-device spyware client seems to be able to take commands via SMS messages. This can be done by putting an SMS listener to the SMS inbox which analyzes every incoming message for commands. After extracting the commands, such a client can delete the message in order to prevent a message notification to the unknowing user.

[40] `http://www.symantec.com`
[41] `http://www.f-secure.com`
[42] `http://www.kaspersky.com`
[43] `http://www.symbiansigned.com`
[44] installable for all compatible devices
[45] Palm PDA malwares
[46] we position spyware on the same level as malware

In 2007, the anti-virus companies listed some new malware families. But as they only address old Symbian devices, nothing really new can be presented. This fact does not really match with the forecast of McAfee, placing mobile malware to the estimated top ten threats in 2007 [21]. It seems to be even the opposite: equal to 2004, 2007 had the least count of new appearing families[47].

As a result people might say that smartphones are only little threatened. But making such a conclusion is a mistake. The SDKs for smartphones are evolving and it is getting easier to develop soft-/malware. Moreover, additional companies offer platforms for smartphones, e.g. Googles Android[48] and Openmoko[49], which seems to influence the formerly stricter signing considerations of Symbian Ltd.[50]. Additionally, researchers are permanently working on finding new security flaws. For example, Mulliner et al. [22] presented a way to perform vulnerability analysis on MMS user agents of Windows Mobile phones. They use “fuzzing” in order to detect vulnerabilities which may be used to create remote fault code injection attacks through MMS messages as infection vector. Another interesting work on Windows Mobile malware development is the master thesis of Boris Michael Leidner [23], where the development of a worm is described in detail. This worm uses a stack-overflow attack in order to infect Windows Mobile 5 devices via WiFi connection. Jesse D’Aguanno[51] gives detailed information on how to attack RIM Blackberry supporting networks[52]. Racic et al. [24] present an operating system-independent attack exploiting vulnerabilities in the MMS protocol. The exploit drains the power from an attacked device up to 22 times faster than in ordinary operation. It is imaginable that some of these proof-of-concepts together with new approaches will find their way into future smartphone malware.

3.4 Smartphone Malware Outlook

As most users of smartphones have private information on their handset, we will focus on possible mal- and spyware targeting these. Therefore, we will have a short look on where to find information describing how to develop such malware. For example, most of the functionalities used in the FlexiSPY spyware are described in several forum threads[53] by professionals and other developers which can be put together by experienced Symbian programmers easily. In addition, several other methods can be found or developed from ideas pointed out there which makes these places to a good starting point for developers of mal- and spyware. By now, not all of the described techniques/code that can be found there are used, especially for malicious extraction personal and private data, but it is imaginable that this will be done in the near future.

In detail while working on an anomaly detection system [25] we found out that it is possible to extract various information and to perform several harmful actions on Symbian OS smartphones. On Table 7 examples on harmful actions are given where the cor-

[47] only 7 new families were recognized

[48] `http://code.google.com/android/`

[49] `http://www.openmoko.com/`

[50] `http://www.symbian.com/news/pr/2007/pr20079685.html`

[51] presented as speaker with the pseudonym “x30n”

[52] proof-of-concept at `http://www.praetoriang.net/presentations/blackjack.html`

[53] `http://newlc.com/forum` and `http://forum.nokia.com/`

responding header and library files are provided. Table 8 shows a subset of extractable values from Symbian OS where a more detailed list can be found in Appendix B. These values can have various purposes, e.g. creating location profiles of observed persons or spying messages.

Table 7: Symbian functions libraries

Functions	Header Files	Library
manipulate contact list	CNTDB.H	cntmodel.lib
manipulate messages (SMS, MMS, email)	MSVAPI.H	msgs.lib
manipulate files	f32file.h	efsrv.lib
take pictures with camera	ECam.h	ecam.lib
record and play audio files	MdaAudioInputStream.h MdaAudioOutput-Stream.h	mediaclientaudiostream.lib
record and play video files	VideoRecorder.h Video-Player.h	mediaclientvideo.lib
determine IMEI[54], IMSI[55], cell location	Etel3rdParty.h	Etel3rdParty.lib
locate user via GPS	lbs.h	lbs.lib
capture key events	COEMAIN.H	cone.lib
send gathered information	ES_SOCK.H	esock.lib

All of these functions can be triggered remotely using e.g. text or email messages in order to submit commands. This can be seen as a first step towards smartphone bots.

Table 8: An excerpt of extractable Symbian OS values

Name	Description
PROCESSES	Running processes, tasks, and threads
CONTACT LIST	Represents the whole contact list
INSTALLED APPLICA-TIONS	List of installed applications (IDs, names)
MAIL DATA	Inbox, Outbox, Sentbox, Draft, receipents, contents
MMS DATA	Inbox, Outbox, Sentbox, Draft, receipents, contents
SMS DATA	Inbox, Outbox, Sentbox, Draft, receipents, contents
LOCATION	Cell and GPS information

Fortunately, it seems like that there are no exploits available which would help creating worms that do not need user interaction in order to propagate (at least for Symbian OS). If such an exploit appears, it could be used to infect a lot of devices in a short time, e.g. in order to create smartphone bot nets which may represent a serious threat in future. The inability of logging 100% reliable data from wireless connections is already used to scatter spam emails via open WiFi access points today. But with the

[55]International Mobile Equipment Identity
[55]International Mobile Subscriber Identity

high number of existing and new smartphones, possible victims for hosting such actions could be found at much more places than now. This even could end in the first artificial pandemic affecting a wide range of people in this world[56]. With such a wide-range affection, huge networks could be built for stealing, abusing, and exchanging data from infected devices.

Table 9: Possible future mobile malware threats

Threats
Read and send local files
Take/send picture with/from camera(s)
Remote connection and control
Exploits
Locate device/person
Personalized spam
Local denial of service
Abuse of various services
Abuse of listeners
Abuse of private information

Basing on the idea of Jamaluddin et al. [26], we developed a Symbian OS malware[57], capable of taking pictures and sending these via MMS to a predefined number in order to show that privacy-related attacks can be easily implemented. The corresponding picture-results can be seen on Figure 11 where we have to note that the users were not aware that a picture was taken[58]. Pressing on "2" triggered the malicious process.

```
Result: Takes and sends pictures
start KeyListener
if KeyEvent = "2" then
    take picture;
    add picture to MMS message;
    send MMS message;
else
    wait;
end
```

Procedure 1 Picture malware

Additionally, tests with manipulating the phone book via SMS commands succeeded too. Therefore we triggered a listener on the SMS inbox folder that only reacted on messages containing two leading "%" characters. Whenever such a message is received, the malware deletes the complete phonebook. Pseudo code for both malware can be found on Procedure 1 and Procedure 2 table.

[56] Smartphone malware is not limited by borders or regions as it can propagate through various communication variants, e.g. BT, IrDA, or internet

[57] code by Frank P.

[58] The users were informed afterwards. For privacy issues, pictures were chosen, that had an average quality.

```
Result: Receives command SMS message and deletes contact list
start SMSListener
if IncomingSMS starts with "%%" then
    prevent normal SMS handling;
    delete contact list;
else
    allow normal SMS handling;
end
```

Procedure 2 SMS malware

Figure 11: Pictures of smartphone users taken and sent by malware. The users were unaware that the picture was taken.

Getting such Symbian OS malwares run "in the wild" is not easy, fortunately. But there is an approach that can be used to address single persons. For signing Symbian OS installation files developer can use developer certificates, which only require the identification numbers (IMEI) of the targeted devices. By pressing "$*\#06\#$" everyone with access to a mobile phone can determine this number and then is able to create a personalized installation file. This gives an attacker the opportunity to infect at least a small number of devices in range.

Regarding the legal issues relating to this kind of malware and attack, it is not not harmless to develop such kind of software. Since 7. August 2007 Germany enacted *new* laws related to information security [27]. Following §202 a,b, and c of this criminal code, several restrictions were made regarding unsolicited data access, e.g. §202a states that you can be confined for up to three years if you actively gain access to foreign data. Furthermore, using, developing, and providing tools that *allow* access to foreign or secured data may cause a similar punishment. At this point we want to state that all the presented self-written malwares were developed for research purpose. Only predefined devices were targeted and the source code will never be released.

A summary of the threats presented in this section can be found on Table 9. Merging these threats resulted in a fictive scenario presented in the next section.

3.4.1 Future Scenario: "A Nightmare on Privacy"

In this scenario we show how dangerous an example mal- and spyware for Symbian OS smartphones using the mentioned malicious functionalities together with possibly upcoming exploits can be. At this point, we point out again that [14] most of the data on a smartphone consists of private information, which can be a lucrative target for spammers, phishers, and similar people, who will likely use the data for abuse or extortion.

At first, the spyware has to be distributed to numerous victims. This can be done using exploits (worm) or by convincing the victim that the application he is installing is something different than intended, e.g. a game. Once installed, the malware can use various techniques in order to hide from the user view, like running as daemon process which can only be detected by an ordinary user through installing a third party process viewer. As every single certified application has a unique identifier number, the malware knows which processes to kill, in order to prevent the detection, e.g. through a virus scanner. If the device is designed to provide video calls, the malicious application starts capturing pictures from the victim until a good quality is achieved. This is done by sending the picture to a remote server which runs a face recognition system that indicates the arrival of a *good* picture.

Then, the malware extracts every usable information from the device that is accessible: IMEI, IMSI, network provider name, IDs of the installed applications, phone book, files, SMS, MMS, email messages[59]. All of these are sent stealthy, using time slots of already opened connections. The remote server processes the data and builds usage profiles and social networks relating to the extracted contacts. These social networks consist of nodes and edges where the nodes represent a person with all information, profiles, and files that were extracted. An edge represents a link between two nodes and consists of a tuple basing on two email addresses, phone numbers, or other contact information. These social networks help to organize information relating to the victims which can be used to abuse or extort users more coordinated. If the attackers knows most of the friends of a victim, it is not that difficult to generate a new message from formerly captured messages that looks really authentic. This kind of message could convince the user, e.g. to download more malicious software or to send money to a friend.

Table 10 shows profiles that can be generated from the extracted values. Profiles can be used manifoldly, e.g. a *user profile* gives every gathered demographic information on one single person, a *location profile* represents the observed locations of a user referring

[59] several further can be found in Appendix B

Table 10: Possible profiles

Profile	Description
User	Indicates extracted information relating to the observed person: age, gender, ethnic, name, profession, interests, user documents, email/internet identities with passwords (if logged)
Usage	Indicates application/phone/internet usage with corresponding time
Location	Shows location with corresponding time, which can be transfered into a movement map
Relation	shows all contact information including email addresses, phone numbers, names

to his monitored device. If this device has an attached or included GPS module, the accuracy is about 5 meters to the real position of the device. Burglars can use this to safely rob a house and a jealous boyfriend can check whether his girlfriend went to the cinema as told or not. A *usage profile* can indicate visited webpages, used applications, or dialed calls, which can help to identify user preferences and service usage (e.g. information hotline) and a *relation profile* gives information on people related to an observed person. These relation information can be used to attack these people or the observed victim itself.

In Figure 12, a small sample network is shown, where the sixth node is selected and the corresponding information are given. Several possible flags indicate information on the person, e.g. the *friends* flag shows which nodes are contacted often (through any kind of medium). Imagining a several times greater network, huge coordinated attacks could be started. These attacks could use results coming from graph-theoretical methods, e.g. optimal radio/GSM-base station distribution. This may prevent friends warning each others from believing in faked messages. A similar approach would be to send friends of a person in a foreign country an email including some *malicious* pictures and a note “here are my vacation pictures”. These pictures may infect the corresponding hosts with mal- and spyware, which in turn can be used again to extract information and to extend the network.

At this point, extracting information should not be too difficult. Most people are not aware of any smartphone threats and trust their device without any suspiciousness. Furthermore, banking card and other PINs are saved in the contact list which can be detected by searching for four-number entries.

The value of the gathered information can be increased by buying and comparing further email address and user:password libraries, as not only an email address can identify a user, but also a unique password[60] that was captured by a key logger or through an infected web page. In the worst case, with this technique the user information can be enriched in a way that almost every created fictive identity can be assigned to its real owner, e.g. a.a@site.nex + “myuniquepass” and fake123@othersite.nex + “myuniquepass” $\rightarrow$ one ID.

[60] or identity, citation, profile

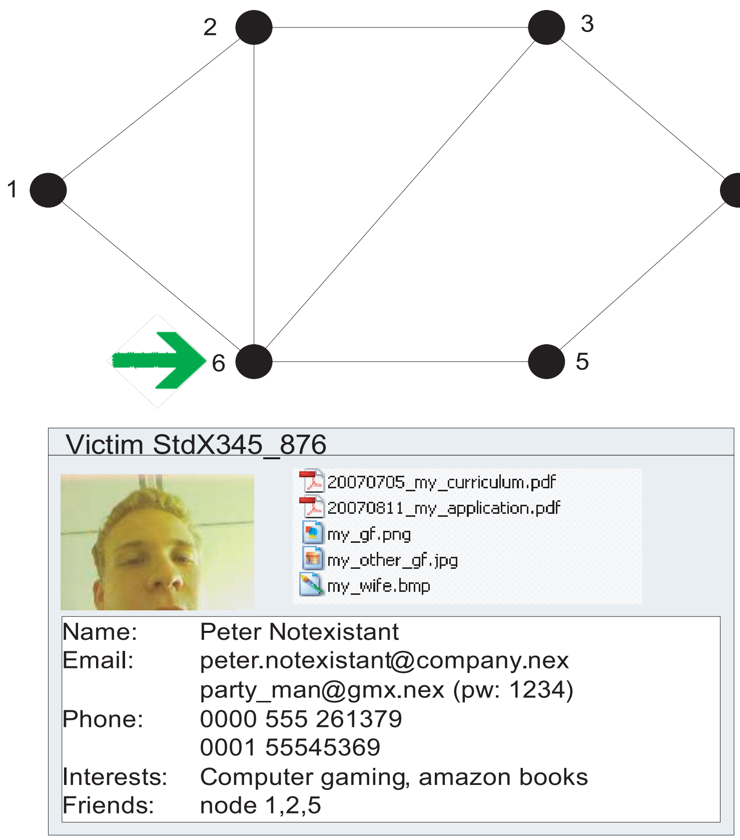

Figure 12: Small sample network with picked node

The more information the attacker gathers, the easier it gets for him to start further attacks that will really hurt a person. People can get blackmailed, as the attacker could know that someone is having a second girlfriend, a serious disease, or was kicked out from his job due to sexual harassment. Bank data can be abused to steal money. Moreover, spam can then be extended to SMS messages, MMS messages, and phone calls which could result in more people believing the misleading information already known from current phishing emails.

This scenario showed that is dangerous to understimate smartphone malware. Most of the used functionalities can be easily implemented using standard APIs. Currently, the only chance to prevent such attacks is to secure the devices by installing security software. Therefore, a summary of possible countermeasures can be found in the next section.

3.5 Countermeasures

Countermeasures, which help to secure a system, can be usually taken by installing certain hard- or software. Three main systems for computers can be identified: firewalls, anti virus software and intrusion detection systems (IDS), where their basic purpose will be explained briefly next on.

Firewalls [28] are so called “white list”-based systems, which means that there is a special list of rules explicitly allowing certain ports to communicate with internal or external peers[61]. If malicious software is able to masquerade as a trusted software using a trusted port, a basic firewall will allow all communication activities. Anti virus scanners use “black lists” in order to detect certain threats included in the black list.

A virus scanner [29] can block viruses, worms and Trojan horses, from infecting the often real time monitored system, whereas only manual scanning can be activated, too. Malware is detected by scanning for and finding a certain string or pattern, also called signature, therefore the malware has to be known by the scanner, otherwise it is not able to detect it.

Intrusion detection systems (IDS) [30] formerly were systems that monitored network traffic in order to log this data, which the network administrator used to detect abnormal behavior. He then started proper countermeasures like closing ports or locking systems. IDS evolved to intrusion prevention systems (IPS) which are able to detect certain abnormal behaviors and take preventive measures automatically.

Up to our knowledge, until today, there is no IDS/IPS commercially available for smartphones so the only possibility to secure a device from attacks is installing a firewall and anti virus software. This leaves smartphones unprotected against new and unknown malware that can not be detected by the signature-based anti virus software. Furthermore, if the user allows network communication for a malware in the firewall rules, because it disguises as a game or something similar, the device will allow all malicious traffic. Hence, from this point of view, smartphones are highly threatened by any kinds of attacks that pass virus scanners and firewalls.

4 Smartphone Malware Research

In this section we present related research in the field of smartphone malware. This includes articles about malware and proposed countermeasures.

One of the first articles on smartphone malware was “Mobile Phones as Computing Devices: The Viruses are Coming!” [2] from David Dagon et al., in which the authors describe the taxonomy of mobile malware at that time (2004). They point out that mostly three security goals are compromised: confidentiality, integrity and availability. Furthermore, they have a special column on battery exhausting attacks, which a virus scanner would not detect.

Another early article on mobile malware from Matt Piercy in 2004 [3] describes the first worm sighted for Symbian OS. This worm is called *Cabir* and disguises itself as an application called “Caribe Security Manager”. If installed, every time the device is started, the worm displays “Caribe” and starts looking for nearby devices with activated Bluetooth interface in order to propagate.

Leavitt et al. [31] provide a first overview on smartphone malware. They present some of the malwares in 2005 and give hints on how to secure devices.

Martin et al. [32] show the three main attacks how to drain the battery from mobile devices. The first are *service request power attacks*, where repeated service requests are made. The second are *benign power attacks*, where the victim is convinced to execute a valid but energy-hungry task repeatedly. The last are *malignant power attacks* which have various kinds.

[61] e.g. TCP/UDP

The problem of securing a mobile device was addressed by several researchers, where several methods were suggested. One approach could be the use of an intrusion detection system for smartphones. Several theoretical papers can be found about this, where Samfat and Molva [33] presented a distributed intrusion detection system for cellular networks that tries to detect abusive behavior like masquerading or eavesdropping in future mobile networks. They use learning algorithms in order to obtain user profiles, which in turn are used as signatures to detect abnormal behavior. This works with features like call length or cell information.

Miettinen et al. [34] designed an intrusion detection framework, which uses host-based and network-based intrusion detection. If an anomaly is detected on a mobile device, the device sends an intrusion alert to a back-end server. This server is able to collect further information from network-based sensors in order to create network related intrusion alarms when necessary. They use a correlation engine in order to correlate the device and network intrusion alarms.

In Schmidt et al. [25] the authors demonstrate how to monitor a smartphone running Symbian OS in order to extract features that describe the state of the device and can be used for anomaly detection. These features are sent to a remote server, because running a complex IDS on this kind of mobile device still is not feasible, due to capability and hardware limitations. They give examples on how to compute some of the features and introduce the top ten applications used by mobile phone users basing on a study in 2005. The usage of these applications is recorded and visualized and for a first comparison, data results of the monitoring of a simple malware are given.

Nash et al. [35] introduce their first steps towards an intrusion detection system handling power-draining attack. Therefore they monitor the power usage of processes and then identify potential battery-exhausting attacks.

Another battery-based IDS is presented by Jacoby et al. [36]. It measures the devices power consumption, which is correlated with the application activity on the device by running a rule-based host intrusion detection engine.

"SmartSiren" represents an approach of Cheng et al. [37], which collects communication data and sends this to a remote server in order to offload the processing burden from resource-constrained smartphones. The data is analyzed for anomalies and if found, the threatened devices are informed

As an alternative to IDS, the suggestions by Hwang et. al. could be used to secure smartphones. In "Securing Embedded Devices" [38] they propose the distinguishment of five layers on an embedded device, like a smartphone, and to put certain security measures on each of these. The pointed out layers are (top-down): protocol level, algorithm level, architecture level, micro architecture level and circuit level.

In the opinion of the author, the approach of Miettinen et al. [34] is the most promising. It regards the resource constrains of smartphones but still allows complex analysis of indicated intrusions. With the increasing capabilities of smartphones, more and more functionality can be moved from the server to the mobile devices. Then, even more complex analysis approaches can be installed to the server side in order to process indicated intrusions. This approach may be supported by the system from Samfat and Molva [33] which would add intrusion detection capabilities to the mobile phone network.

5 Conclusions and Future Work

This technical report presented the still present danger of malware for smartphones. We showed that using publicly available APIs can lead to new malwares that are able to extract various private information as well as to perform harmful action on infected devices. Private information are the number one data on mobile phones, and hence, a loss or modification will harm every affected person. But, as less and less critical malwares appear, security consideration seem to lose their importance. This is a big mistake and the provided scenario showed that underestimating smartphone malware can cause serious problems not only concerning privacy issues.

Future research has to target this mistake and more possibilities for securing smartphones have to be developed. One approach is the use of intrusion detection systems where not only scientific papers are demanded. Practical solutions are needed and, therefore, our future work will address this problem.

Acknowledgments

I thank the CC Security at DAI-Labor for their support, especially Ahmet Camtepe, Karsten Bsufka, and Frank Peters for the pictures of malware *victims* from CC-SEC at DAI-Labor.

References

[1] Bundesverband Informationswirtschaft Telekommunikation und neue Medien e.V.- BITKOM: Mehr Handys als Einwohner in Deutschland (2006) http://www.bitkom.de/41015_40990.aspx (online visited 2007.10.04).

[2] Dagon, D., Martin, T., Starner, T.: Mobile phones as computing devices: The viruses are coming! IEEE Pervasive Computing **3**(4) (2004) 11–15

[3] Piercy, M.: Embedded devices next on the virus target list. IEE Electronics Systems and Software **2** (December-January 2004) 42–43

[4] Abowd, G.D., Iftode, L., Mitchel, H.: The Smart Phone: A First Platform for Pervasive Computing. IEEE Pervasive Computing (2005) 18–19 April-June.

[5] Roussos, G., March, A.J., Maglavera, S.: Enabling Pervasive Computing with Smart Phones. IEEE Pervasive Computing (2005) 20–27 April-June.

[6] Canalys: EMEA Q3 2006 - Highlight From the Canalys Research (2006) http://www.canalys.com/ (online visited 2007.10.04).

[7] Symbian Software Limited: Symbian OS - the mobile operating system (2007) http://www.symbian.com (online visited 2007.10.04).

[8] Microsoft Corporation: Windows Mobile (2007) http://www.microsoft.com/ (online visited 2007.10.04).

[9] Research In Motion: Blackberry (2007) http://www.blackberry.com/ (online visited 2007.10.04).

[10] Palm: Palm developer network (2007) https://pdn.palm.com (2007.10.04) (online visited 2007.10.04).

[11] ARCchart: (2007) http://www.arcchart.com/ (online visited 2007.10.04).

[12] SUN: Java me at a glance (2007) http://java.sun.com/javame/index.jsp (online visited 2007.10.04).

[13] Technology, T.: Consumer Trends in Mobile Applications - A TNS Technology Briefing for Technology Decision Makers (2005) http://www.tns-global.com/ (online visited 2007.10.04).

[14] InfoWatch and Zoom.CNews: Mobile Device Security 2007 (March 2007) http://www.infowatch.com/ (online visited 2007.10.04).

[15] Szor, P.: Virus Research and Defense. Addison Wesley (2005)

[16] Bundesamt fuer Sicherheit in der Informationstechnik: Mobile endgeräte und mobile applikationen: Sicherheitsgefährdungen und schutzmaßnahmen (2006)

[17] Gostev, A.: Mobile Malware Evolution: An Overview, Part 1 (September 2006) http://www.viruslist.com/ (online visited 2007.10.04).

[18] Shevchenko, A.: An overview of mobile device security (2005) http://www.viruslist.com (online visited 2007.10.04).

[19] Vervata: FlexiSPY (2006) http://www.flexispy.com/ (online visited 2007.10.04).

[20] Symantec: Spyware.FlexiSpy (March 2006) http://www.symantec.com (online visited 2007.10.04).

[21] McAfee: McAfee Avert Labs Unveils Predictions for the Top Ten Security Threats in 2007 As Hacking Comes of Age (November 2006) http://www.mcafee.com/ (online visited 2007.10.04).

[22] Mulliner, C., Vigna, G.: Vulnerability Analysis of MMS User Agents. In: ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference, Washington, DC, USA, IEEE Computer Society (2006) 77–88

[23] Leidner, B.M.: Voraussetzungen für die entwicklung von malware unter windows mobile 5. Master's thesis, Rheinisch-Westfälische Technische Hochschule Aachen (2007)

[24] Racic, R., Ma, D., Chen, H.: Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone's Battery. In: Proceedings of the Second IEEE Communications Society / CreateNet International Conference on Security and Privacy in Communication Networks (SecureComm), Baltimore, MD. (aug 2006)

[25] Schmidt, A.D., Peters, F., Lamour, F., Albayrak, S.: Monitoring smartphones for anomaly detection. In: MOBILWARE 2008, International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, Insbruck, Austria (2008) to appear.

[26] Jamaluddin, J., Zotou, N., Edwards, R., Coulton, P.: Mobile Phone Vulnerabilities: A New Generation of Malware. In: Proceedings of the 2004 IEEE International Symposium on Consumer Electronics. (September 2004) 199–202

[27] Ringstorff, H., Merkel, A., Zypries, B.: Einundvierzigstes strafrechtsänderungsgesetz zur bekämpfung der computerkriminalität (41. strÄndg) (aug 2007) http://www.bgblportal.de/BGBL/bgbl1f/bgbl107s1786.pdf.

[28] Lodin, S.W.; Schuba, C.: Firewalls fend off invasions from the net. Spectrum, IEEE **35**(2) (Feb 1998) 26–34

[29] Szor, P.: 11 Antivirus Defense Techniques. In: Virus Research and Defense. Symantec Press (2005) 425–491

[30] Kemmerer, R.A.; Vigna, G.: Intrusion detection: A Brief History and Overview. Computer **35**(4) (Apr 2002) 27–30

[31] Leavitt, N.: Mobile phones: The next frontier for hackers? IEEE Computer **38**(4) (2005) 20–23

[32] Martin, T., Hsiao, M., Ha, D., Krishnaswami, J.: Denial-of-service attacks on battery-powered mobile computers. In: PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), Washington, DC, USA, IEEE Computer Society (2004) 309

[33] Samfat, D., Molva, R.: IDAMN: An Intrusion Detection Architecture for Mobile Networks. IEEE Journal on Selected Areas in Communications **15**(7) (September 1997) 1373–1380

[34] Miettinen, M., Halonen, P., Hätönen, K.: Host-Based Intrusion Detection for Advanced Mobile Devices. In: AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06), Washington, DC, USA, IEEE Computer Society (2006) 72–76

[35] Nash, D.C., Martin, T.L., Ha, D.S., Hsiao, M.S.: Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. In: PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, Washington, DC, USA, IEEE Computer Society (2005) 141–145

[36] Jacoby, G.A., Marchany, R., IV, N.J.D.: How mobile host batteries can improve network security. IEEE Security and Privacy **4**(5) (2006) 40–49

[37] Cheng, J., Wong, S.H., Yang, H., Lu, S.: Smartsiren: virus detection and alert for smartphones. In: MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services, New York, NY, USA, ACM (2007) 258–271

[38] Hwang, D.D., Schaumont, P., Tiri, K., Verbauwhede, I.: Securing Embedded Systems. Security & Privacy Magazine, IEEE **4**(2) (2006) 40–49

[39] Department of Defense: DoD Dictonary of Military Terms (2001) `http://www.dtic.mil/doctrine/jel/doddict/data/s/04767.html` (online visited 2007.10.04).

[40] Schäfer, G.: Network Security (2004) `http://www-tkn.ee.tu-berlin.de/curricula/networksecurity` (online visited 2007.10.04).

A Security in the Common (Information Technology) Sense

One interpretation of the word “security” is the condition of being protected against danger or loss. The department of defense in the U.S.A. defines it as a condition that results from the establishment and maintenance of protective measures that ensure a state of inviolability from hostile acts or influences [39]. Other introduce the term “security goal” to be able to describe single goals that have to be achieved in order to say a computer system or network is secure. These goals are confidentiality, data integrity, accountability, availability and controlled access, which is shown on Table 11 [40]. These goals aim

Table 11: Security Goals

Confidentiality	Data that is transmitted or stored should only be revealed to an intended audience
Data Integrity	Modification of data should be possible to detect and the creator should be identifiable
Accountability	It should be possible to identify the entity responsible for any communication event
Availability	Services should be available and function correctly
Controlled Access	Only authorized entities should be able to access certain services or information

to cope with following threats: masquerading, eavesdropping, authorization violation, loss or modification of data, repudiation, forgery, and sabotage, which are described in Table 12 [40].

Table 12: Security Threats

Masquerading	An entity claims to be another
Eavesdropping	An entity reads information that it is not intended to read
Authorization Violation	An entity uses a service or resource it is not intended to use
Loss or Modification of Data	Data is being altered or destroyed
Repudiation	An entity falsely denies its’ participation in a communication act
Forgery	An entity creates new information in the name of another entity
Sabotage	Any action that aims to reduce the availability and / or correct functioning of services or systems

B Extractable Values from Symbian OS

In Table 13 you can find some more of the extractable information from Symbian OS devices. These information can be accessed by using the given APIs. The following values base on API calls, that were granted using a *developer certificate* that basicially every developer can request. Further values, e.g. mobile network information or very sensitive OS data can be accessed using a *phone manufacturer approved certificate* that only trusted partners of Symbian Ltd. can aquire. The table is has three columns, where the name of the extractable values, the complexity of computing, and a description is given.

Table 13: More Features extracted on Symbian OS devices

Name	Complexity	Description
KEYLOCK STATUS	simple	Is Keylock activated?
USER INACTIVITY TIME	simple	Time in seconds, where user was inactive
BATTERY CHARGE LEVEL	medium	Battery charge level
BATTERY STATUS	medium	Power supply plugged?
CONNECTION DATA	medium	How many connection interfaces are used and which amount of data was sent (e.g. WLAN, 3G, BT, IrDA, ...)
DATE AND TIME	medium	Date and time on the device
DISK DATA	medium	Size, available space
FILE SYSTEM DATA	medium	files
IMEI	medium	Device identification
IMSI	medium	User identification
IP ADDRESS	medium	IPv4 and IPv6 Address if assigned
REMOVABLE DATA	medium	Size, available space
REMOVABLE PLUGGED	medium	Is a storage module plugged?
PROCESSES	medium	Running processes, tasks, and threads
CONTACT LIST	medium	Represents the whole contact list
INSTALLED APPLICATIONS	complex	List of installed applications (IDs, names)
OS DATA	complex	CPU usage, available RAM, RAM size
MAIL DATA	complex	Inbox, Outbox, Sentbox, Draft, receipents, contents
MMS DATA	complex	Inbox, Outbox, Sentbox, Draft, receipents, contents
SMS DATA	complex	Inbox, Outbox, Sentbox, Draft, receipents, contents
LOCATION	complex	Cell and GPS information

C Acronyms

API	Application Programming Interface
BT	Bluetooth
CPU	Central Processing Unit
DoS	Denial of Service
Email	Electronic Mail
FOMA	Freedom of Mobile Multimedia Access
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
IDS	Intrusion Detection System
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IrDA	Infrared Data Association
J2ME	Java 2 Micro Edition
KVM	"Kilobyte" Virtual Machine
MMC	Multimedia Card
MMS	Multimedia Messaging System
OS	Operating System
SDK	Software Developing Kit
SIS	Symbian Installation System
SMS	Short Message Service
TCP	Transmission Control
UMTS	Universal Mobile Telecommunications System
W-CDMA	Wideband Code Division Multiple Access
WiFi	Wireless Fidelity

D Malware List

The following tables represent the malwares that could be excerpted from online virus databases. The table gives names, types, days, months, years, and descriptions of the corresponding malwares. The tables are sorted by the discovery dates.

Table 14: Malware List

Name	Type	D	M	Y	Payload
Palm. Libertycrack	Trojan horse	8	30	2000	Deletes applications and files
Palm. Vapor	Trojan horse	9	22	2000	Deletes applications and files
Palm. Phage	Virus	9	25	2000	Deletes applications and files
Palm. MTX.II.A	Virus	?	?	2001	display messages
SymbOS. Cabir.A	Worm	6	15	2004	replicates via Bluetooth
WinCE. Duts.A	Virus	7	17	2004	appends itself to all non-infected exe
SymbOS. Skulls	Trojan horse	11	19	2004	replaces files disables apps
SymbOS. Cabir.B	Worm	11	22	2004	replicates via bt same as cabir.a only txt different
SymbOS. Cdropper.H	Trojan	11	30	2004	drops
SymbOS. Skulls.B	Trojan horse	11	30	2004	replaces files disables apps and drops
SymbOS. Cdropper.C	Trojan horse	11	30	2004	drops
SymbOS. Cdropper.A	Trojan horse	12	9	2004	replaces files drops disables apps
SymbOS. Cabir.E	Worm	12	14	2004	replicates via bt only txt is different to cabir b
SymbOS. Cabir.D	Worm	12	14	2004	replicates via bt only txt and filename changed
SymbOS. Cabir.C	Worm	12	14	2004	replicates via bt only txt is different to cabir.b
SymbOS. Cdropper.B	Trojan horse	12	22	2004	drops
SymbOS. Cabir.J	Worm	12	22	2004	replicates via bt and creates files
SymbOS. Skulls.C	Trojan horse	12	22	2004	replaces files disables apps
SymbOS. MGDropper	Trojan horse	12	22	2004	replaces files and disables apps drops cabir
SymbOS. Cabir.H	Worm	12	22	2004	replicates via bt
SymbOS. Cabir.G	Worm	12	22	2004	replicates via bt
SymbOS. Cabir.I	Worm	12	29	2004	replicates via bt
SymbOS. Cabir.L	Worm	12	29	2004	replicates via bt
SymbOS. Cabir.F	Worm	12	30	2004	replicates via bt only filename changed
SymbOS. Cdropper.M	Trojan horse	12	30	2004	drops cabir.j
SymbOS. Cabir.K	Worm	12	30	2004	replicates via bt and creates files
SymbOS. Cabir.T	Worm	1	5	2005	replicates via bt only filename changed
SymbOS. Cabir.N	Worm	1	5	2005	replicates via bt only filename changed

Name	Type	D	M	Y	Payload
SymbOS.Cabir.O	Worm	1	5	2005	replicates via bt only filename changed
SymbOS.Cabir.P	Worm	1	5	2005	replicates via bt only filename changed
SymbOS.Cabir.R	Worm	1	5	2005	replicates via bt only filename changed
SymbOS.Cabir.Q	Worm	1	5	2005	replicates via bt only txt and filename changed
SymbOS.Cabir.S	Worm	1	5	2005	replicates via bt only txt and filename changed
SymbOS.Skulls.D	Trojan horse	1	5	2005	drops cabir.m disables apps shows image to screen
SymbOS.Cabir.M	Worm	1	6	2005	propagates via bt only txt and filename changed
SymbOS.Lasco.A	Worm Virus	1	10	2005	replicates via bt and file injection bases on cabir.h source
SymbOS.Cdropper.D	Trojan horse	2	1	2005	drops cabir variants and shows messages
SymbOS.Cdropper.E	Trojan horse	2	1	2005	drops cabir.b and lock-nut
SymbOS.Locknut	Trojan horse	2	2	2005	drops cabir variants and replaces files which cause a dysfunctional device
SymbOS.Commwarrior.A	Worm	3	7	2005	replicates via bt and mms
SymbOS.Commwarrior.B	Worm	3	7	2005	replicates via bt and mms does not choose clock for deciding on replication method
SymbOS.Dampig.A	Trojan horse	3	8	2005	drops cabir variants dis-ables apps replaces files
SymbOS.Drever.A	Trojan horse	3	21	2005	disables apps
SymbOS.Drever.B	Trojan horse	3	22	2005	disables app
SymbOS.Skulls.F	Trojan horse	3	22	2005	drops cabir variants and locknut.b replaces files disables apps flashed skull pictures
SymbOS.Drever.C	Trojan horse	3	22	2005	replaces files disbles apps-virus scanners

Name	Type	D	M	Y	Payload
SymbOS.Skulls.F	Trojan horse	3	24	2005	drops cabir variants and locknut.b replaces files disbles apps shows skull
SymbOS.Skulls.E	Trojan horse	3	24	2005	replicates via bt drops variants cabir disables apps
SymbOS.Skulls.H	Trojan horse	3	30	2005	drops cabir variants and locknut.b replaces files disables apps flashed skull pictures
SymbOS.Skulls.G	Trojan horse	3	30	2005	disables apps replaces files
SymbOS.Mabir.A	Worm	4	4	2005	replicates via bt and mms listen on incoming mms and sms and answers with infected mms
SymbOS.Fontal.A	Trojan horse	4	6	2005	replaces files prevents reboot
SymbOS.Hobbes.A	Trojan horse	4	17	2005	replaces files disables apps possibly only phone call works
SymbOS.Locknut.B	Trojan horse	4	18	2005	drops cabir.v and locknut.b prevents boot installs corrupted files
SymbOS.Cabir.V	Trojan horse	4	29	2005	replicates via bt only filename is changed
SymbOS.Cabir.Y	Trojan horse	4	29	2005	replicates via bt only name changed
SymbOS.Skulls.I	Trojan horse	5	5	2005	drops cabir variants and locknut.b replaces files disables apps
SymbOS.Skulls.K	Trojan horse	5	9	2005	drops cabir.m replaces files disables apps
SymbOS.AppDisabler.A	Trojan horse	5	18	2005	disables apps
SymbOS.Skulls.J	Trojan horse	6	13	2005	drops appdisabler.a which drops cabir.y and locknut.b disables apps replaces files
SymbOS.Singlejump.C	Trojan horse	6	15	2005	disables files drops singlejump.b uses modified variant of cabir to replicate
SymbOS.Fontal.B	Trojan horse	6	22	2005	raplaces files prevents reboot disables apps

Name	Type	D	M	Y	Payload
SymbOS.Skulls.M	Trojan horse	6	22	2005	replaces files disables apps
SymbOS.Doomboot.A	Trojan horse	7	7	2005	replaces files prevents reboot drains power through sending commwarrior.b via bt prevent reboot
SymbOS.Doomboot.B	Trojan horse	7	14	2005	replaces files prevents reboot
SymbOS.Skulls.L	Trojan horse	7	14	2005	replaces files drops cabir variants disables apps
SymbOS.Doomboot.C	Trojan horse	7	21	2005	replaces files prevents reboot
SymbOS.Cabir.U	Worm	7	27	2005	replicates via bt
SymbOS.Blankfont.A	Trojan horse	8	10	2005	replaces files
SymbOS.Cabir.Z	Trojan horse	8	31	2005	replicates via bt only filename changed
SymbOS.Fontal.C	Trojan horse	9	7	2005	replaces files disables apps prevents rebooting
SymbOS.Doomboot.D	Trojan horse	9	7	2005	prevent reboot replaces files
SymbOS.Skulls.N	Trojan horse	9	16	2005	replaces files disables apps
SymbOS.Doomboot.E	Trojan horse	9	19	2005	prevents reboot replaces files
SymbOS.Doomboot.G	Trojan horse	9	22	2005	drops commwarrior.a+b fontal.a replaces files prevents rebooting
SymbOS.Cardtrap.A	Trojan horse	9	22	2005	copies windows malware to mem card replaces files disables apps
SymbOS.Skulls.O	Trojan horse	9	22	2005	drops fontal.a and commwarrior.b replaces files disables apps
SymbOS.Doomboot.F	Trojan horse	9	22	2005	drops skulls.d cabir.m fontal.a replaces files prevents reboot
SymbOS.Appdisabler.D	Trojan horse	9	23	2005	replaces files disables apps

Name	Type	D	M	Y	Payload
WinCE.Brador.A	Trojan horse	9	23	2005	backdoor
SymbOS.Appdisabler.E	Trojan horse	9	23	2005	drops cabir.b replaces files disables apps
SymbOS.Cardtrap.B	Trojan horse	9	23	2005	drops doomboot.a copies windows malware to memory card replaces files disables apps
SymbOS.Skulls.P	Trojan horse	9	26	2005	drops mabir.a prevents rebooting replaces files disables apps
SymbOS.Singlejump.D	Trojan horse	9	26	2005	drops cabir variants replaces files disables apps prevent rebooting malware renamed to onehop.d
SymbOS.Skulls.Q	Trojan horse	9	27	2005	drops commwarrior.b and cabir variants replaces files disables apps
SymbOS.Appdisabler.F	Trojan horse	9	27	2005	replaces files disables apps
SymbOS.Appdisabler.G	Trojan horse	9	29	2005	replaces files disables apps drops cabir variants
SymbOS.Cardblock.A	Trojan horse	10	3	2005	deletes files set password to memory card
SymbOS.Skulls.R	Trojan horse	10	4	2005	drops mabir.a replaces files disables apps
SymbOS.Fontal.C	Trojan horse	10	4	2005	replaces files disables apps prevents rebooting
SymbOS.Cardtrap.C	Trojan horse	10	7	2005	drops components of doomboot.a
SymbOS.Commwarrior.C	Worm	10	14	2005	replicates via bt mms memory card
SymbOS.Cabir.V	Worm	10	24	2005	replicates via bt only filename is changes
SymbOS.Cardtrp.D	Trojan horse	11	9	2005	replaces files disables apps drops malwares as doomboot component
SymbOS.Doomboot.M	Trojan horse	11	10	2005	replaces files prevents rebooting drops caommwarrior.f

Name	Type	D	M	Y	Payload
SymbOS.Doomboot.N	Trojan horse	11	10	2005	replaces files prevents rebooting
SymbOS.Locknut.C	Trojan horse	11	10	2005	replaces files disables apps prevents rebooting drops cabir.b
SymbOS.Skulls.S	Trojan horse	11	10	2005	drops cabir.f replaces files disables apps
SymbOS.Skulls.T	Trojan horse	11	11	2005	replaces files disables apps drops locknut.c
SymbOS.Cardtrap.G	Trojan horse	11	11	2005	drops windows malware to memory card drops doomboot components
SymbOS.Cardtrap.F	Trojan horse	11	14	2005	replaces files disables apps prevents reboot
SymbOS.Skulls.U	Trojan horse	11	14	2005	drops locknut.a and doomboot.a components drops cabir.b cabir.x locknut.c mgdropper.a replaces files disables apps
SymbOS.Skulls.V	Trojan horse	11	18	2005	replaces files disables apps drops mgdropper.a locknut.a doomboot.a cabir.b cabir.x
SymbOS.Pbstealer.A	Trojan horse	11	21	2005	reads private information and send this via bt (contact data)
SymbOS.Doomboot.P	Trojan horse	11	28	2005	replaced files prevents reboot
SymbOS.Drever.D	Trojan horse	11	28	2005	replaces files disables apps
SymbOS.Ruhag.C	Trojan horse	11	28	2005	replaces files disables apps
SymbOS.Cardtrp.H	Trojan horse	11	28	2005	installs to memory card replaces files disables apps
SymbOS.Fontal.G	Trojan horse	11	29	2005	replaces files disables apps prevents reboot
SymbOS.Doomboot.I	Trojan horse	11	29	2005	replaces files disables apps prevents rebooting
SymbOS.Fontal.D	Trojan horse	11	29	2005	replaces files disables apps drops commwarrior.b

Name	Type	D	M	Y	Payload
SymbOS.Fontal.E	Trojan horse	11	29	2005	replaces files disables apps prevent reboot
SymbOS.Fontal.D	Trojan horse	12	2	2005	replaces files disables apps prevents reboot
SymbOS.Hidmenu.A	Trojan horse	12	3	2005	replaces files
SymbOS.Pbstealer.B	Trojan horse	12	4	2005	read provate information and sends this via bt
SymbOS.Pbstealer.B	Trojan horse	12	5	2005	reads private information and sends this via bt
SymbOS.Doomboot.Q	Trojan horse	12	5	2005	replaces files disables apps prevents rebooting
SymbOS.Bootton.C	Trojan horse	12	7	2005	replaces files disables apps prevents rebooting
SymbOS.Cardtrap.I	Trojan horse	12	12	2005	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrp.K	Trojan horse	12	12	2005	replaces files disables apps installs windows malware to mem card
SymbOS.Cardtrap.J	Trojan horse	12	12	2005	reaplces files diables apps installs windows malware to mem card
SymbOS.Cardtrap.L	Trojan horse	12	12	2005	replaces files disables apps drops windows malware to memory card manipulates private data (deletes calendar and phonebook)
SymbOS.Singlejump.I	Trojan horse	12	13	2005	replaces files disables apps drops doomboot components
SymbOS.Skulls.O	Trojan horse	12	13	2005	replaces files disables apps drops fontal a and commwarrior.b
SymbOS.Skulls.P	Trojan horse	12	13	2005	replaces files disables apps drops mabir.a cabir variants doomboot and fontal components
SymbOS.Cardtrap.M	Trojan horse	12	14	2005	replaces files disables apps installs windows malware to mem card

Name	Type	D	M	Y	Payload
SymbOS.Skulls.Q	Trojan horse	12	14	2005	replaces files disables apps drops commwarrior.b doomboot compnents
SymbOS.Cardtrap.N	Trojan horse	12	14	2005	replaces files disables apps installes windows malware to mem card
SymbOS.Bootton.D	Trojan horse	12	14	2005	drops doomboot.a cabir.g replaces files disables apps
SymbOS.Dampig.B	Trojan horse	12	15	2005	drops cabir disables apps replaces files
SymbOS.Cabir.W	Worm	12	15	2005	replicates via bt only filename changed
SymbOS.Cardtrap.O	Trojan horse	12	15	2005	replaces files disables apps install windoews malware to mem card
SymbOS.Doomboot.R	Trojan horse	12	15	2005	replaces files disables apps prevents rebooting
SymbOS.Cabir.W	Trojan horse	12	15	2005	replicates via bt only filename changed
SymbOS.Dampig.C	Trojan horse	12	16	2005	replaces files disables apps drops malware
SymbOS.Cardtrap.P	Trojan horse	12	16	2005	replaces files disables apps drops windows malware to memory card
SymbOS.Bootton.B	Trojan horse	12	25	2005	replaces files prevents reboot
SymbOS.Bootton.A	Trojan horse	12	25	2005	replaces files disables apps
SymbOS.Singlejump.F	Trojan horse	12	28	2005	replaces files disables apps prevents rebooting sends singlejump.b to bt devices in range
SymbOS.Singlejump.G	Trojan horse	12	28	2005	replaces files disables apps drops doomboot.a components sends doomboot.a to bt devices in range
SymbOS.Singlejump.H	Trojan horse	12	28	2005	reaplces files disables apps prevents rebooting sends cabirdropper to device in bt range

Name	Type	D	M	Y	Payload
SymbOS.Pbstealer.C	Trojan horse	1	3	2006	reads private information and sends this via bt
SymbOS.Pbstealer.D	Trojan horse	1	18	2006	reads private information and sends this via bt
SymbOS.Bootton.E	Trojan horse	1	18	2006	replaces files prevents rebooting
SymbOS.Sendtool.A	Trojan horse	1	18	2006	spreads other malware via bt user interaction needed
SymbOS.Cardtrap.P	Trojan horse	1	22	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.R	Trojan horse	1	27	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.S	Trojan horse	1	27	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.Q	Trojan horse	1	27	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.T	Trojan	2	1	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.E	Trojan horse	2	1	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.X	Trojan horse	2	2	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.U	Trojan horse	2	8	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.X	Trojan horse	2	8	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.V	Trojan horse	2	8	2006	replaces files disables apps installes windows malware to mem card

Name	Type	D	M	Y	Payload
SymbOS.Cardtrap.W	Trojan horse	2	8	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.Y	Trojan horse	2	11	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.AB	Trojan horse	2	17	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cardtrap.Z	Trojan horse	2	17	2006	replaces files disables apps installes windows malware to mem card
J2ME.RedBrowser.a	Trojan horse	2	28	2006	abuses messaging system
SymbOS.Cardtrap.AA	Trojan horse	3	6	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Appdisabler.I	Trojan horse	3	7	2006	replaces files disables apps
SymbOS.Commwarrior.D	Worm	3	9	2006	replicates via bt and mms only txt is changed
SymbOS.Mabtal.A	Trojan horse	3	10	2006	drops mabir.a fontal.a locknut.b
WinCE.Cxover.A	Worm	3	15	2006	replicates via MS ActiveSync
SymbOS.Doomboot.S	Trojan horse	3	16	2006	replaces files prevents rebooting
SymbOS.Commwarrior.E	Worm	3	17	2006	replicates via bt and mms
SymbOS.Commdropper.D	Trojan horse	3	20	2006	send commwarrior.e via mms
SymbOS.Cdropper.L	Trojan horse	3	23	2006	drops cabir.ad
SymbOS.Cardtrap.AC	Trojan horse	4	5	2006	replaces files disables apps installes windows malware to mem card
SymbOS.Cdropper.N	Trojan horse	4	6	2006	drops cabir.a
WinCE.Letum.A	Worm	4	8	2006	replicates via MS information reads private data sends itself to captured addresses uses usenet registry entries to propgate in usenet

Name	Type	D	M	Y	Payload
SymbOS.Arifat.A	Trojan horse	4	13	2006	reads private information (user password logger) and sends this via sms
SymbOS.Blankfont.B	Trojan horse	4	16	2006	reaplces files prevent reboot
WinCE.Brador.B	Trojan horse	5	6	2006	
SymbOS.Commdropper.C	Trojan horse	5	17	2006	drops commwarrior.h
SymbOS.Commwarrior.F	Worm	5	17	2006	replicates via bt and mms
SymbOS.Mabtal.B	Trojan horse	5	17	2006	drops mabir.a
SymbOS.Commdropper.A	Trojan horse	5	17	2006	drops commwarrior variants
SymbOS.Bootton.F	Trojan horse	5	17	2006	replaces files prevents reboot
SymbOS.Commwarrior.H	Worm	5	18	2006	replicates via bt and mms
SymbOS.Commwarrior.G	Worm	5	18	2006	replicates via mms and bt reads private information(local contact list)
SymbOS.Commdropper.B	Trojan horse	5	18	2006	drops commwarrior.a+b+c
SymbOS.Cardtrp.AF	Trojan horse	5	19	2006	replaces files disables apps installer windows malware to memory card
SymbOS.RommWar.A	Trojan horse	5	19	2006	replaces files disables apps and buttons
SymbOS.Stealwar.B	Trojan horse	5	20	2006	drops commwarrior.a pbstealer.a rommwar.a
SymbOS.Stealwar.C	Trojan horse	5	20	2006	drops pbstealer.f cabir.k mabir.a commwarrior.b
SymbOS.Stealwar.E	Trojan horse	5	20	2006	drops cabir.a commwarrior.a pbstealer.f
SymbOS.Stealwar.D	Trojan horse	5	20	2006	drops cabir.k pbstealer.f commwarrior.c
SymbOS.Stealwar.A	Trojan horse	5	20	2006	drops pbstealer commwarrior or cabir

Name	Type	D	M	Y	Payload
SymbOS.Cardtrap.AE	Trojan horse	5	21	2006	replaces files disables apps installs windows malware to mem card
SymbOS.Cardtrap.AD	Trojan horse	5	24	2006	reaplces files disables apps installs windows malware to mem card
SymbOS.Commwarrior.I	Worm	5	25	2006	replicates via bt and mms
SymbOS.RommWar.B	Trojan horse	5	25	2006	replaces files prevents rebooting
SymbOS.Doomboot.T	Trojan horse	5	25	2006	replaces files drops commwarrior.l prevents rebooting
SymbOS.RommWar.D	Trojan horse	5	25	2006	reaplces files disables apps and buttons
SymbOS.RommWar.C	Trojan horse	5	25	2006	replaces files prevents rebooting
SymbOS.Romride.B	Trojan horse	6	2	2006	replaces files disables files
SymbOS.Romride.A	Trojan horse	6	2	2006	replaces files disables apps
SymbOS.Romride.E	Trojan horse	6	5	2006	replaces files disables apps
SymbOS.Commwarrior.L	Worm	6	5	2006	replicates via bt and mms
SymbOS.Romride.D	Trojan horse	6	5	2006	replaces files disables apps
SymbOS.Commwarrior.K	Worm	6	5	2006	replicates via bt and mms
SymbOS.Commdropper.D	Trojan horse	6	5	2006	drops commwarrior.e
SymbOS.Romride.C	Trojan horse	6	5	2006	replaces files disables apps
SymbOS.Commwarrior.J	Worm	6	6	2006	replicates via bt and mms
SymbOS.Commdropper.E	Trojan horse	6	6	2006	drops commwarior.d
SymbOS.Romride.F	Trojan horse	6	21	2006	replces files disables apps
SymbOS.Romride.H	Trojan horse	6	21	2006	replaces files disables apps
SymbOS.Romride.G	Trojan horse	6	21	2006	reaplces files disables apps
SymbOS.Dropper.A	Trojan horse	6	22	2006	drops windows malware
SymbOS.Commdropper.G	Trojan horse	6	22	2006	drops commwarrior.m

Name	Type	D	M	Y	Payload
SymbOS.Cardtrp.AG	Trojan horse	6	22	2006	reaplces files disables apps installs windows malware to the memory card
SymbOS.Commwarrior.N	Worm	6	22	2006	replicates via bt and mms
SymbOS.Commwarrior.M	Worm	6	22	2006	replicates via bt and mms
SymbOS.Commdropper.F	Trojan horse	6	23	2006	drops commwarrior.k
SymbOS.Cdropper.F	Trojan horse	6	28	2006	drops cabir variants
SymbOS.Cdropper.K	Trojan horse	6	28	2006	drops cabir.b components
SymbOS.Cdropper.G	Trojan horse	6	28	2006	drops cabir and skulls components
SymbOS.Cdropper.I	Trojan horse	6	28	2006	drops locknut and cabir
SymbOS.Cdropper.J	Trojan horse	6	29	2006	drops cabir.b
SymbOS.Cdropper.O	Trojan horse	6	30	2006	drops cabir.a+b
SymbOS.Cdropper.R	Trojan horse	6	30	2006	drops cabir
SymbOS.Dampig.D	Trojan horse	6	30	2006	drops dampig.a and cabir variants
SymbOS.Cdropper.S	Trojan horse	6	30	2006	drops cabir variants
SymbOS.Doomboot.U	Trojan horse	6	30	2006	replaces files prevents rebooting
SymbOS.Cdropper.P	Trojan horse	6	30	2006	drops cabir variants
SymbOS.Cdropper.Q	Trojan horse	7	2	2006	drops cabir variants
SymbOS.Doomboot.W	Trojan horse	7	4	2006	replaces files prevents reboot
SymbOS.Doomboot.V	Trojan horse	7	4	2006	replaced files prevents reboot
SymbOS.Ruhag.D	Trojan horse	7	5	2006	replaces files disables apps
SymbOS.Ruhag.E	Trojan horse	7	6	2006	replaces files disables apps
SymbOS.Cabir.X	Worm	7	6	2006	replicates via bt only file name changed
SymbOS.Skulls.R	Trojan horse	7	6	2006	replaces files disables appsdrops mabir.a
SymbOS.Commdropper.H	Trojan horse	7	7	2006	drops commwarrior.g
SymbOS.Doomboot.X	Trojan horse	7	7	2006	replaces files prevents rebooting
SymbOS.Mabir.B	Trojan horse	7	8	2006	replicates via mms and bt

Name	Type	D	M	Y	Payload
SymbOS.Doomboot.P	Trojan horse	7	26	2006	replaces files prevents rebooting
SymbOS.Commwarrior.Q	Trojan horse	8	1	2006	replicates via bt mms memory card uses browser
SymbOS.Bootton.G	Trojan horse	8	8	2006	replaces files prevents rebooting
J2ME.Wesber.a	Trojan horse	9	6	2006	abuses nessaging
SymbOS.Blankfont.C	Trojan horse	9	10	2006	replaces files disables apps prevents rebooting
SymbOS.Appdisabler.L	Trojan horse	10	26	2006	reaplces files disables apps
SymbOS.Appdisabler.K	Trojan horse	10	26	2006	replaces files disables apps
SymbOS.Appdisabler.J	Trojan horse	10	26	2006	replaces files disables apps
SymbOS.Keaf	Worm	10	29	2006	reads private information abuses messaging (sends link for downloading itself to all contacts)
SymbOS.Appdisabler.M	Trojan horse	10	31	2006	replaces files disables apps
SymbOS.Appdisabler.N	Trojan horse	11	7	2006	replaces files disables apps
SymbOS.Appdisabler.Q	Trojan horse	11	7	2006	replaces files disableas apps
SymbOS.Appdisabler.O	Trojan horse	11	7	2006	replaces files disables apps
SymbOS.Stealwar.F	Trojan horse	11	7	2006	doprs cabir.a commwarrior.a mosquit.a lasco.a pbstealer.f
SymbOS.Appdisabler.P	Trojan horse	11	7	2006	replaces files disables apps
SymbOS.Cardtrap.AH	Trojan horse	11	7	2006	replaces files disables apps install windows malware to mem card
SymbOS.Romride.I	Trojan horse	11	9	2006	reaplces files causes boot loop
SymbOS.Flerprox.A	Trojan horse	11	9	2006	reaplces files disables apps

Name	Type	D	M	Y	Payload
SymbOS.Romride.J	Trojan horse	11	9	2006	replaces files replaces files causes boot loop
SymbOS.Appdisabler.R	Trojan horse	11	11	2006	replaces files disables apps
SymbOS.Appdisabler.S	Trojan horse	11	29	2006	replaces files disables apps
SymbOS.Appdisabler.T	Trojan horse	12	11	2006	replaces files disables apps
SymbOS.Appdisabler.U	Trojan horse	12	11	2006	reaplces files disables apps
SymbOS.Commdropper.J	Trojan horse	12	22	2006	drops commwarrior.e
SymbOS.Commwarrior.T	Trojan horse	1	15	2007	replicates via bt mms memory card
SymbOS.Commwarrior.h	Worm	1	15	2007	reads private data replicates via mms and bt
SymbOS.RommWar.c	Trojan horse	1	25	2007	no description available
SymbOS.Cabir.AD	Trojan horse	1	25	2007	replciates via bt only filename changed
SymbOS.Cabir.AI	Trojan horse	1	25	2007	replicates via bt
SymbOS.Cabir.AE	Trojan horse	1	25	2007	replicates via bt
SymbOS.Commwarrior.i	Worm	2	11	2007	replicates via bt and mms
SymbOS.Mrex.a	Trojan horse	3	27	2007	no description available
SymbOS.Viver.A	Trojan horse	5	15	2007	abuse messaging
SymbOS.Viver.B	Trojan horse	5	17	2007	abuses messaging
SymbOS.Feaks.a	Trojan horse	5	29	2007	abuses messaging
SymbOS.Appdisabler.V	Trojan horse	5	29	2007	replaces files disables apps
SymbOS.Feak.a	Trojan horse	5	29	2007	no description available
SymbOS.Bootton.H	Trojan horse	6	27	2007	reaplces files prevents rebooting
SymbOS.Bootton.I	Trojan horse	6	28	2007	replaces files prevents rebooting
SymbOS.Fontal.i	Trojan horse	7	31	2007	replaces files disables apps
SymbOS.SHT.a	Trojan horse	8	29	2007	no description available
SymbOS.Skuller.af	Trojan horse	8	31	2007	no description available
SymbOS.Delcon.a	Trojan horse	8	31	2007	no description available
SymbOS.Pbstealer.f	Trojan horse	8	31	2007	abuses messaging read private information
SymbOS.Appdisabler.W	Trojan horse	8	31	2007	replaces files disables apps

Name	Type	D	M	Y	Payload
SymbOS.Appdisabler.x	Trojan horse	10	31	2007	no description available
SymbOS.HatiHati.a	Worm	12	13	2007	abuses mesaging replicates via mmc
SymbOS.Fonzi.a	Trojan horse	1	5	2008	no description available
SymbOS.Killav.a	Trojan horse	1	10	2008	replaces files disables apps
SymbOS.Beselo.a	Worm	1	2	2008	replicates via bt and mms
SymbOS.Cabir.o	Worm	1	23	2008	no description available
SymbOS.Beselo.b	Worm	1	23	2008	replicates via bt and mms
SymbOS.Lasco.b	Worm	1	26	2008	no description available
SymbOS.Acallno.b	Trojan horse	1	26	2008	no description available