Monitoring Smartphones for Anomaly Detection

Aubrey-Derrick Schmidt
DAI-Labor
Technische Universität Berlin
aubrey.schmidt@dai-labor.de

Frank Peters
DAI-Labor
Technische Universität Berlin
frank.peters@dai-labor.de

Florian Lamour
DAI-Labor
Technische Universität Berlin
florian.lamour@dai-labor.de

Sahin Albayrak
DAI-Labor
Technische Universität Berlin
sahin.albayrak@dai-labor.de

ABSTRACT

In this paper we 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 intrusion detection system (IDS) on this kind of mobile device still is not feasible, due to capability and hardware limitations. We 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.

Keywords

Monitoring, Smartphone, Anomaly Detection, Mobile Device

1. INTRODUCTION

Mobile phones get more and more popular. Since August 2006, more mobile phones than inhabitants are registered in Germany [3]. As the capabilities of these devices increase, they are not simple voice centric handsets anymore; rather they provide mobile computing power that can be used for several purposes. Especially *smartphones* represent a possibility of moving appropriate applications from the PC to mobile devices, as they mostly provide large bandwidth wireless network access, office tools and the possibility of installing third party programs. But with the increase of capabilities, more and more malicious software (malware) targeting these devices emerge. We believe that the evolution of malware for mobile devices will take a similar direction as the evolution of PC malware. This will lead to the same problems, like missing signatures for unknown threats and new appearing malware at high frequency.

This paper introduces an approach how to monitor smartphones in order to extract values that can be used for *remote* anomaly detection. Anomaly detection does not need signatures in order to work, which allows the detection of new and unknown malware. Therefore, it has to be learned what is the normal behavior of a user and device in order to be able to distinguish between normal and abnormal, possibly malicious actions. The extracted features are sent as vector to a remote system, taking the responsibility for extended security measures away from the probably unaware user. These vectors can be used for methods from the field of artificial intelligence, like Artificial Immune Systems (AIS) [5] or Self-Organizing Maps (SOM) [2], in order to detect abnormal behavior.

In Section 2 we explain what a smartphone actually is and where it can be used. In Section 3 we show how to build a monitoring client running on a smartphone and give explicit examples on values that can be extracted from Symbian OS devices. In order to be able to learn, what is normally done on smartphones, we map actions excerpted from a study on mobile phone usage to different use cases and specify testing scenarios on them. Examples of these, together with the corresponding monitoring results, are given in Section 4. Finally in Section 5, we conclude the paper and point out the future work. Note that a list of further features measurable on Symbian OS is given in the Appendix.

2. SMARTPHONES

In this paper we use the expression *smartphone*[1] in order to describe a mobile device that mostly unifies functionalities of a cellular phone, a PDA, an audio player, a digital camera and camcorder, a GPS[2] receiver and a PC.

Smartphones often use PC-like *QWERTY* keyboards in order to increase typing speed and sometimes PDA-like pen displays for improved data and command handling. Mechanisms were developed that additionally improve text input, like T9, which means "Text on 9 keys" and represents predictive text technology. Smartphones use different techniques for creating wireless connections for communication purpose:

MOBILWARE 2008, February 13-15, Innsbruck, Austria
Copyright © 2008 ICST 978-1-59593-984-5
DOI 10.4108/ICST.MOBILWARE2008.2492

[1]In the sense of this work, we will use the expressions *smartphone*, *mobile phone* and *mobile device* equivalently.
[2]Global Positioning System

- GSM[3] represents the second generation (2G) of mobile end-to-end communication, mainly used for voice calls and services like SMS[4]

- GPRS[5] in combination with 2G is often described as 2.5G, as it provides voice and packet data

- W-CDMA[6] was designed as replacement of GSM and is used in the FOMA[7] system (JP) and UMTS[8], being able to transport data at higher speed than GSM.

Additionally, the devices provide Bluetooth, Wireless LAN (WLAN) or IrDA[9] 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*[1], where interesting areas of application were pointed out by Roussos et al. [12]:

- Mobile Phones as information service endpoint, e.g. applied as navigational assistance or location based services.

- Mobile Phones as remote controllers for different devices, like television or HiFi station.

- Mobile Phones as pervasive network hubs to provide wide area connectivity, e.g. for wearable systems that need to communicate in order to transmit health related data.

- Mobile Phones as ID tokens in order to store information used to verify the user and information.

Smartphones can be applied to these fields, not only, but partly because they have standardized operating systems installed. Following Canalys [4], there are three main competitors in this field: Symbian OS from Symbian Ltd. [14], Windows Mobile from Microsoft [10], and Research In Motion (RIM) with the Blackberry devices, with 78.7%, 16.9%, and 3.5% market share on the smart mobile device market, respectively. These operating systems enable the installation of third party applications, which allows customization of the device according to the software needs of the user. But this customization brings the danger of being infected by malware along. According to Peter Gostev from Kaspersky Lab. [6], from June 2004 to August 2006, 31 new families of malware for mobile devices with 170 variants were recognized, where *Caribe* was the first worm to hit the mobile community (Symbian OS).

Jamaluddin et al. described in [8] how easy it is to write a Trojan horse capable of sending SMS messages to premium services. The usage of this malware is visualized in Subsection 4.5.

[3]Global System for Mobile Communications
[4]Short Message Service
[5]General Packet Radio Service
[6]Wideband Code Division Multiple Access
[7]Freedom of Mobile Multimedia Access
[8]Universal Mobile Telecommunications System
[9]Infrared Data Association

3. THE MONITORING CLIENT

Intrusion detection can be separated into two fields: signature-based misuse detection and anomaly detection. As the devices are monitored for anomaly detection, it is important to monitor device data that enables differentiation between normality and anomalies. In [13] Eugene Spafford et al. point out that host-based approaches, direct data collection techniques and internal sensors are preferable to network-based approaches, indirect data collection techniques and external sensors. This was taken into account when designing our monitoring client.

3.1 The Client Design

We propose three main components for the monitoring client: User Interface, Communication Module and Feature Extractor.

The User Interface enables command entering, like changing server or port. It can be used to visualize the state of the monitoring client, e.g. sending or buffering, or even to indicate anomaly detection results.

The Communication Module is responsible for managing connection states and sending or buffering the feature vectors, which is shown on Figure 1. If the client cannot connect due to signal loss, it starts buffering until a connection can be established. If a connection is not possible before the buffer is filled, it adds the last extracted vector and removes the first.

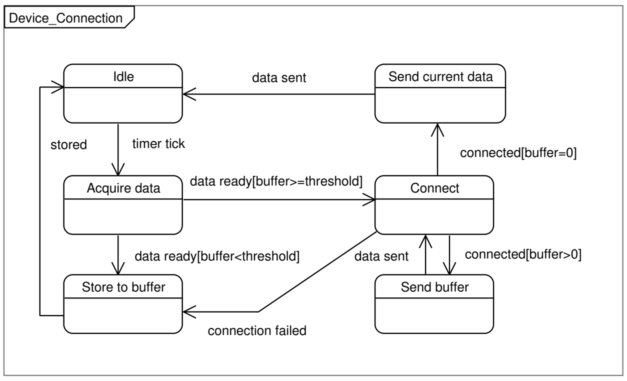

Figure 1: The possible connection states of the monitoring client

The Feature Extractor has several different components gathering and computing features. Features describe the state of the monitored device. They represent various measurements and observations of resources and other hard- and software components. If no direct interfaces are provided by the operating system, features are extracted by using algorithms or methods, which provide approximated results. This is done with care, as additional encumbering of the already limited device possibly distort the monitoring results.

3.2 The Symbian Client

The monitoring client was developed in Symbian C++ version S60 3rd with Nokia Carbide.vs and consists of the three proposed components. *The User Interface* can be used to change server port and address, to start, stop or move the

Figure 2: The Nokia E61 smartphone running the monitoring client.

client into the background. Further user information can be inserted in order to control access to the remote server. For reasons of program stability and to prevent interference, the GUI is running in a thread separate from the other components. Further work may even remove the user interface to a separate application, since there is no need to tie up GUI resources for an application running in the background. *The Communication Module* uses SOAP[10] Webservices on top of TCP/IP in order to communicate with the server. As we found out, sending data — or even just remaining in ready-to-send mode — is rather expensive in terms of battery power. To prevent the rapid depletion of the power source, first, all data is stored locally and sent in bulk after reaching a certain threshold level. *The Feature Extractor* is triggered to fetch new monitored data every thirty seconds[11], which is stored locally and later, upon reaching the threshold, sent to the server using the appropriate service.

3.3 The Symbian Features

Table 1: An Excerpt of the Extracted Features

Name	Compl.	Description
RAM FREE	simple	Indicates the amount of free RAM in Kbyte
USER INACTIVITY	simple	Indicates, if the user was active in the last ten seconds
PROCESS COUNT	medium	Indicates the amount of running processes
CPU USAGE	complex	Represents the CPU Usage in percent
SMS SENT COUNT	complex	Represents the amount of SMS messages in the sent directory

Symbian OS[12] provides some programming interfaces for extracting features, e.g. fetching the amount of free RAM or the user inactivity time are one-line commands. But not all areas are covered, especially reading network traffic packets cannot be done by average programmers, as the application programming interfaces are restricted. Some other features need complex method constructs in order to be extracted. We distinguish between three different method complexities: *simple*, *medium* and *complex*. Features that can be called through Symbian C++ interfaces taking only one or few lines of code are categorized as *simple*. Features that need several classes or algorithms to be computed are marked as *complex*. Everything in between is marked as *medium*. Some of the features can be used to identify and manage observed users or devices. In the following, we describe how to compute some of the features shown on Table 1 with pseudo code. Some of these will be used to visualize user activity in Section 4.4 and Section 4.5. In order to present, how Symbian C++ programming looks like, we will show the real call for getting the available RAM:

[10]formerly: Simple Object Access Protocol
[11]due to hardware limitations
[12]tested on Version 9.1 S60 3rd

RAM FREE is a feature that can be easily extracted. All applications need more or less RAM in order to work, so every running program/malware should have impact on this value.

```
User::LeaveIfError(HAL::Get(
        HALData::EMemoryRAMFree, iFreeRamSize));
```

USER INACTIVITY indicates if a button was pressed within the last 10 seconds. If so, a "0" is returned else a "1". This feature uses a function that returns the absolute user inactivity time in seconds. This value is very interesting for giving hints on activities that are not directly caused by the user and happen automatically and/or periodically in the background.

Table 2: Pseudo Code for Indicating User Activity

```
GET UserInactivityTime

IF UserInactivityTime ≥ 10 seconds
    RETURN User is inactive
ELSE
    RETURN User is active
```

The PROCESS COUNT can be easily computed through a while loop that is checking the existence of processes. Each started application should increase the process count at least by one, and so should malware.

Table 3: Pseudo Code for Getting the Process Count (and Further Information)

```
WHILE there are more processes
    INCREASE counter
    FETCH process information from process object
    STORE process information

RETURN the counter
```

The CPU USAGE cannot be read through a given Symbian OS interface. While searching for an approximation, we found a method described by Marcus Gröber [7] that manually checks, whether the CPU is busy or not. This is done by requesting a timer event with low priority 100 times a second. Another request with high priority checks every second how often the low priority request was actually called. The answer can be used to approximate the usage of the CPU, as the more the CPU is busy the less the low priority request will be called. The following code fragment shows the main calls and functions of this method.

Table 4: Pseudo Code for Approximating the CPU Usage

```
CREATE new requesting Active Object with low priority
CREATE new checking Active Object with high priority

SEND low priority time request to CPU 100 times/second
CHECK the number of accepted requests every second
        Return approximated CPU usage every second
```

SMS SENT COUNT like every feature relating to messaging (SMS, MMS, and email) needs some more complex functions to be computed. But once implemented, most of the similar features can be extracted using the same classes. Together with USER INACTIVITY this feature can help to indicate malware sending cost causing messages.

Table 5: Pseudo Code for Getting the Amount of Sent SMS Messages

```
CREATE messaging session
CONNECT messaging session to sent folder
        SELECT SMS sent folder
                RETURN amount of SMS messages
```

3.4 Securing the Monitoring Client

As the communication or even the monitoring client itself can be targeted by malware, it is important to secure the functionality of the client.

Using encryption for the communication channel should be a proper way to secure communications. Securing the application is more complicated. The Symbian OS API provides a method for setting processes to different *critical* levels. On highest level, if the monitoring client process is killed, the device reboots and restarts the process. This functionality was not added yet, as it obviously could lead to denial of service attacks on the device, but at least it would guarantee either that the monitoring agent is running or that the user brings the device to a specialist. Another possibility is checking the running applications, that are clearly identifiable through a unique ID, and as soon as an unknown application is started, this could be compared with an application white list, that includes all allowed programs. If an unknown program was started, the system could kill it or alert the user and system.

No.	Application	Usage
1.	SMS	83%
2.	Games	61%
3.	Camera	49%[15]
4.	MMS Picture	46%
5.	PDA Functions	36%
6.	Internet	31%
7.	WAP	30%[16]
8.	Bluetooth	28%
9.	Email	27%
10.	Video Camera	27%

Table 6: TNS GTI 2005 Top Ten Applications/Services

4. EXPERIMENTS

As our goal is to provide data that enables differentiation between normal and malicious device usage, we need to know first, what actually is normal. TNS Technology released a booklet [15] sourced from the TNS Technology's Global Technology Insight (GTI) 2005, where typical user actions on mobile phones are described. We excerpted actions that we performed on a Nokia E61 smartphone [11] in order to monitor normality. The corresponding software behaviors, visualized as data results, can be found in the Section 4.4.

4.1 TNS GTI 2005 Study

The GTI 2005 bases on data coming from 6807 people aged 16 to 49, in 15 different countries. These respondents used 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 [15], which we used to excerpt the top ten actions, that were introduced in that work. These top ten actions base on the percentage of mobile phone users, who use the corresponding application and can be seen on Table 6.

4.2 Testing Specification

In order to perform the actions, we had to specify testing scenarios where we had to distinguish between different use cases, for example a smartphone user can send *and* receive a SMS message of various size with various recipients. We identified about 40 use cases and specified a testing protocol for each. An example protocol is given on Table 7.

Table 7: The Testing Specification for Multi-player Game - Miniblaster

Preconditions:

- Miniblaster is installed on two devices
- Bluetooth is disabled
- settings in Miniblaster:
    - music/sound enabled
- Two minutes of non-device-usage before testing

Testing:

1. Launch Miniblaster on both devices
2. Start hosting on one device
3. Join game on second device
4. Play two rounds
5. Host exits game with left selection key
6. Second device confirms note and exits
7. Two minutes of non-device-usage

Expected Results:

- Fall of *FREE RAM*
- Raise of *CPU USAGE*
- Bluetooth gets enabled
- Data transfer

4.3 Technical Set Up

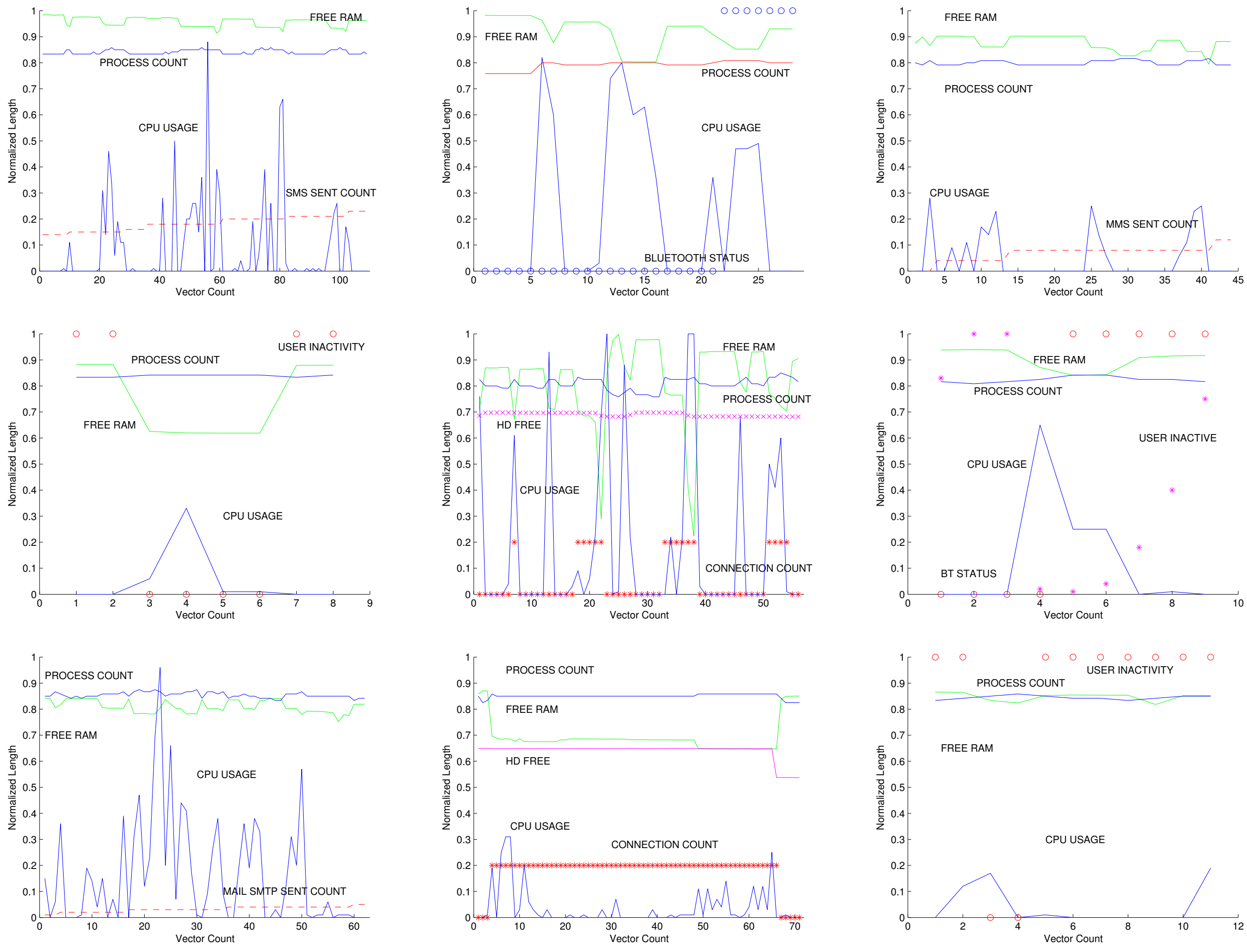

Figure 3: Actions monitored on a Nokia E61: u.l. sending SMS messages, u.c. simple/3D/multiplayer gaming, u.r. sending MMS messages, c.l. PDA function reading .PDF file, c.c. internet browsing, c.r. Bluetooth data transfer, b.l. sending email, b.c. mp3 download, b.l. PDA function creating new calendar entry

We used the Nokia E61 smartphone for testing, which is running Symbian OS 9.1 and has a *QWERTY* keyboard. It supports most of the conventional techniques and protocols used in current smartphones, for example WCDMA and WLAN. A 64Mbyte storage card is plugged, which allows storage of various files, like videos, which then can be viewed on the 320 × 240 pixel display [11]. The installed Symbian-*C*++ monitoring client was triggered for sending a feature vector every 20 seconds to our test server with public IP-address and attached database. This is done using a Webservice via UMTS-connection. The feature vector, that was sent, has a size of less than 8 Kbyte and contains about 50 features.

4.4 Results

On the Figure 3 you can see the usage of most of the top ten applications: Figure 3 shows on the upper left the usage of the Short Message Service. It is separated into four parts: sending empty message, writing and sending a 150-character message, writing and sending a 300-character message and writing and sending a 150-character message with multiple recipients. The upper center shows the usage of three different kinds of games: a simple game called Miniblaster, a more complex game named Sky Force Reloaded and Miniblaster in multi player Bluetooth mode. The upper right visualizes sending an empty MMS message, writing and sending a 150-character MMS message and writing and sending a MMS message with attached picture. The center left represents the usage of PDA functionalities; in detail it is reading a .PDF file. Browsing the internet can be seen on center graph, where different links were clicked and a picture was downloaded. The center right refers to sending an image to a paired Bluetooth device. The bottom left displays sending of various emails. On the bottom center graph, we used the internet to download an 8 Mbyte MP3 file, which was played afterwards. Finally, the bottom right graph represents the making of a new entry into the calendar.

What we can see, although the number of vectors varies on the different figures, is that each application affects the corresponding features in a different way, for example gaming produces much more CPU utilization than creating and sending MMS messages. This encourages the attempt to apply anomaly detection to the field of malware detection.

4.5 Malware Data

As we mentioned before, we use the malware proposed by

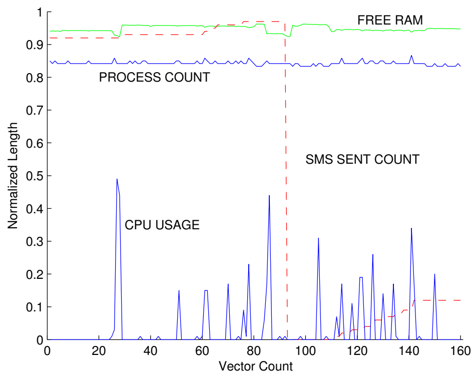

Figure 4: Malware activity on a Nokia E61

Jamaluddin et al. [8] in order to monitor example malicious behavior. If activated, this malware sends a SMS message every time the key "2" is pressed. In Figure 4, every time the SMS SENT COUNT increases, an increase of processes and CPU busyness and a decrease of available RAM can be observed. At vector count 96 we determined, that a Nokia E61 device only can hold 100 SMS messages, which lead to the deletion of these.

5. CONCLUSION AND FUTURE WORK

In this paper we demonstrated how a smartphone can be monitored in order to transmit feature vectors to a remote server. The gathered data is intended to be used for anomaly detection methods that analyze the data for distinguishing between normal and abnormal behavior. Abnormal behavior can indicate malicious software activity. Furthermore, even unknown malware can be detected, since no signatures are used. In our results we saw, that most of the top ten applications preferred by mobile phone users affect the monitored features in different ways. This strengthens the approach of using anomaly detection in order to detect malware on mobile devices.

Gathering more data from different kinds of smartphones that are running different operating systems, like MS Windows Mobile or Palm OS, will be one of the tasks that we will focus in future. Furthermore, additional malware is needed for increasing the quality of our data sets. If these sets are big enough, we will start to test methods from various fields from computer science, like AI and information retrieval, in order to attempt to detect the malicious activities. A first step towards this can be seen in [9] from our colleague Katja Luther et al., where a biological inspired immune system analysis feature-based network data.

Acknowledgments

This work was funded by Deutsche Telekom Laboratories.

6. REFERENCES

[1] Gregory D. Abowd, Liviu Iftode, and Helena Mitchel. The Smart Phone: A First Platform for Pervasive Computing. *IEEE Pervasive Computing*, pages 18–19, 2005. April-June.

[2] Sahin Albayrak, Christian Scheel, Dragan Milosevic, and Achim Müller. Combining Self-Organizing Map Algorithms for Robust and Scalable Intrusion Detection. In M. Mohammadian, editor, *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA 2005)*, pages 123–130. IEEE Computer Society, 2005.

[3] Bundesverband Informationswirtschaft Telekommunikation und neue Medien e.V.- BITKOM. Mehr Handys als Einwohner in Deutschland. `http://www.bitkom.de/41015_40990.aspx` (15. August 2007), 2006.

[4] Canalys. EMEA Q3 2006 - Highlights from the Canalys Research. `http://www.canalys.com/pr/2006/r2006102.htm` (15. August 2007), 2006.

[5] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Self-nonself Discrimination in a Computer. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 202–212. IEEE Computer Society Press, 1994.

[6] Alexander Gostev. Mobile Malware Evolution: An Overview, Part 1. `http://www.viruslist.com/en/analysis?pubid=200119916` (15. August 2007), September 2006.

[7] Marcus Gröber. Applications for Symbian. `http://www.mgroeber.de/epoc.htm` (15. August 2007).

[8] Jazilah Jamaluddin, Nikoletta Zotou, Reuben Edwards, and Paul Coulton. Mobile Phone Vulnerabilities: A New Generation of Malware. In *Proceedings of the 2004 IEEE International Symposium on Consumer Electronics*, pages 199–202, September 2004.

[9] Katja Luther, Rainer Bye, Tansu Alpcan, Sahin Albayrak, and Achim Müller. A Cooperative AIS Framework for Intrusion Detection. In *Proceedings of the IEEE International Conference on Communications (ICC 2007)*, 2007.

[10] Microsoft Corporation. Windows Mobile. `http://www.microsoft.com/germany/windowsmobile/default.mspx` (15. August 2007), 2007.

[11] Nokia. Nokia E61. `http://www.nokia.co.uk/A4221036` (15. August 2007), 2007.

[12] George Roussos, Andy J. March, and Stavroula Maglavera. Enabling Pervasive Computing with Smart Phones. *IEEE Pervasive Computing*, pages 20–27, 2005. April-June.

[13] Eugene Spafford and Diego Zamboni. Data Collection Mechanisms for Intrusion Detection Systems. CERIAS Technical Report 2000-08, CERIAS, Purdue University, 1315 Recitation Building, West Lafayette, IN, June 2000.

[14] Symbian Software Limited. Symbian OS - the mobile operating system. `http://www.symbian.com` (15. August 2007), 2007.

[15] TNS Technology. Consumer Trends in Mobile Applications - A TNS Technology Briefing for Technology Decision Makers. `http://www.tns-global.com/` (15. May 2007), 2005.