A Rich Client-Server Based Framework for Convenient Security and Management of Mobile Applications

Stephen Badan[1], Julien Probst[1], Markus Jaton[1], Damien Vionnet[2], Jean-Frédéric Wagen[2], and Gérald Litzistorf[3]

[1] University of Applied Sciences of Western Switzerland - HES-SO
HES-SO / HEIG-VD, 1400 Yverdon, Switzerland
markus.jaton@heig-vd.ch
[2] HES-SO / EIA-FR, 1705 Fribourg, Switzerland
jean-frederic.wagen@hefr.ch
[3] HES-SO / HES-GE, 1202 Genève, Switzerland
gerald.litzistorf@hesge.ch

Abstract. Contact lists, Emails, SMS or custom applications on a professional smartphone could hold very confidential or sensitive information. What could happen in case of theft or accidental loss of such devices? Such events could be detected by the separation between the smartphone and a Bluetooth companion device. This event should typically block the applications and delete personal and sensitive data. Here, a solution is proposed based on a secured framework application running on the mobile phone as a rich client connected to a security server. The framework offers strong and customizable authentication and secured connectivity. A security server manages all security issues. User applications are then loaded via the framework. User data can be secured, synchronized, pushed or pulled via the framework. This contribution proposes a convenient although secured environment based on a client-server architecture using external authentications. Several features of the proposed system are exposed and a practical demonstrator is described.

Keywords: mobile security, smartphone, rich client, client-server, secure framework, authentication, transient authentication, theft detection.

1 Introduction

Smartphones are increasingly popular : their performance and recent enhancements in user interface technologies (such as touchscreen, intuitive behavior, ease of use) have popularized their use. Becoming indispensable companion devices, they also keep sensitive information. This covers personal information, such as contacts, addresses, phone calls, to do lists and other data. Usually there are also business data stored on the mobile phone, such as the professional contacts of a bank advisor, or a confidential list of customers data for example. The increasingly popular email clients (whether push or pull) result in the storage of confidential

A.U. Schmidt and S. Lian (Eds.): MobiSec 2009, LNICST 17, pp. 23–34, 2009.

messages. The loss of a mobile phone is not usually considered too critical. Most mobile devices feature easy back-up possibilities on standard personal computers; but the data contained in a lost or stolen smartphone is usually left unsecured, and might be examined by unauthorized or even malicious persons.

Many authors (e.g., [1], [2], [3]) and at least one company, Research In Motion (RIM) [4], have recognized the potential security threat of such devices and proposed interesting solutions. Focusing on readily available solutions, RIM proposes Blackberry. Blackberry devices are very popular in business areas, as the information kept on the device is ciphered and is usually impossible to exploit without proper authentication of the user. Thus, a lost Blackberry device will usually not be readable. The Blackberry solution is however criticized by some potential users, i.e. bank professionals and politics, especially. The problem with the Blackberry solution is mainly the existence within the network of proprietary Network Operations Centers (so-called NOC's) whose functionality is widely opaque to the users. Although the data is fully protected throughout the whole network, the implementation of the security features relies widely on proprietary code located in the NOCs and in the Blackberry phone. This is considered as a possible backdoor by professional users [5].

This contribution proposes an alternative architecture for the implementation of a highly secured client-server infrastructure for mobile applications. The proposed system is called here OSMOSYS (Open and Secure Mobile SYStem). This system features:

- An end-to-end security scheme without opaque components. The code can be independently audited and validated by the entity using the system.
- A convenient management of users, user groups and mobile devices is performed centrally, via a security server.
- Application validations and updates performed remotely on the mobile client.
- Introduction of a so-called companion device which insures periodic user authentication without the need of any manipulation from the user and, thus, offers a protection against loss or theft.
- An advanced development environment allowing the creation of fully secured custom applications.
- A set of useful applications for the secured management of personal data, like e-mail clients, contact list, meetings and notes managers.
- The choice of various devices from several smartphone manufacturers (HTC, LG, SAMSUNG, SONY ERICSSON, ...).

The next section will briefly review the state of the art of some existing solutions to offer security to smartphone users. The requirements for a framework securing user applications are then discussed in Section 3. The main authentication and security features are also explained. The chosen client-server architecture of the framework is then detailed in Section 4. An implementation is described in Section 5 to field test the OSMOSYS framework. Finally, this contribution is concluded in Section 6.

2 State of the Art

As stated in the introduction, many solutions to secure data on mobile devices have been proposed. For example, in [1], a wearable token is used to protect a mobile device: this model was called "transient authentication". The solution requires the author's cryptographic file system (named ZIA). Securing applications on mobile devices also received a lot of attention, including from [2] who designed a flexible code security architecture based on "security-by-contract". Unlike [3], these works do not focus on the needs from the security managers of a fleet of mobile devices.

On the commercial side, one of the leader is actually RIM (Research In Motion) with its BlackBerry devices and associated network [4]. Security on those devices relies mainly on strong ciphering (AES), and optionally on a wireless companion device that can be paired with the mobile to detect loss or theft of the smartphone. The main drawback of the Blackberry security solutions resides in their opacity. The hardware used is proprietary: BlackBerry or some Nokia's compatible devices. The support network relies on a small number of Network Operations Centers (NOCs) whose implementation is largely opaque [4].

Another relatively strong contender is Microsoft in its latest Windows Mobile release 6 (WM6 for short) [6]. WM 6 builds on top of the .NET framework to integrate end-to-end security for mobile applications [7]. WM6, in its version 6.1 in particular, integrates security features that make such applications nearly as secure as their BlackBerry counterparts. On the other hand, the .NET framework code, like the Windows Mobile OS, is not open, and cannot be audited by customers. This leaves place for potential backdoors as in the BlackBerry solution.

Another caveat of the WM6 security scheme is its usage of a mobile VPN as secured transport channel : although the VPN is potentially secure by itself, it remains a low level security solution : if somebody can control the mobile device, the VPN might turn itself into an opened door to the corporate network.

It is actually impossible to speak about mobile solutions without mentioning Apple's iPhone which has revolutionized the way people consider their smartphones. The iPhone is clearly a hot piece of technology, but security is actually not on a par with the ease of use [9]. Moreover, the relatively severe restrictions introduced by Apple to the development of applications discourage developers from developing strong security infrastructures on Apple's platform, although it should not be impossible as our recent work tends to show.

Nokia relies entirely on a clever mobile VPN solution for security. The pertinence of a VPN in a mobile environment may however be questioned, as for the WM6's solution. Third party solutions (commercial, or not [8]) allow to cipher the voice calls or the smartphone at the OS level, but this generally involves relatively complex installation procedures on the devices.

Ideally, it is assumed here that a secure application on a mobile device should,

- belong to fleet of devices managed by a security manager,
- be easy to install and should update itself automatically,
- require very few explicit authentication procedures (e.g., password entry),

- be loss and theft-safe,
- expose only the very necessary subset of data required by the users,
- be able to detect if it has been modified or tampered with, and
- keep secure backup copies of valuable data whenever possible on a remote, trusted, location,

To our knowledge, only RIM's BlackBerry offers nearly such functionalities. But if the user or an IT manager needs to audit the code of the whole system, there is currently no solution. This situation might explain why smartphones are proscribed in many governments, banks and companies using very sensitive information.

3 A Framework for Secured Applications

The main goals of the proposed framework is to authenticate the user and the server as well as to secure the data on the mobile phone in the most convenient manner. Indeed, users will not use or disable annoying security features [1].

Strong authentication methods must be conceived and implemented to access the secured applications and their relevant data. Any remote access to or from the application must use bi-lateral authenticated sessions and ciphered transmissions. Any data on the mobile phone must be either securely deleted when the application closes or ciphered when permanently stored on the mobile device.

Theft or loss protection must be insured: in both cases, the applications and their data should become unusable. A way must also be found to protect the applications so they cannot be copied, modified and re-installed to jeopardize the security.

Thus, the framework should offer convenient solutions to the following issues:

- deployment and control of the life cycle of the applications (installation, upgrade, removal),
- configurations of the security parameters,
- user and user group management, and
- management of the required devices: mobile phones and companion devices.

To determine the required functionalities, a brief scenario is imagined in the next sub-section. A use case follows to define the actors and to detail the authentication services.

3.1 A Sample Scenario

Imagine a key account manager, Alice, having to deal with confidential data related to her customers. Starting her workday, she turns on her smartphone and a so-called companion device small and thin enough to be weared or to fit in her pocket.

On her smartphone, she launches the application called OSMOSYS. This application opens and requests her username and password. Behind the scene,

OSMOSYS is also verifying that the companion device can be reached via Bluetooth, and that the smartphone itself is allowed to access the company network.

A set of user applications is then presented to her: an agenda, a to-do list, and a contact list. She opens up her agenda. Nothing has changed since yesterday. But she must prepare herself for an afternoon meeting. Working at her temporary working place in the customer premises, she must go make a few photocopies. She leaves her smartphone on the desk. Despite the fact that this could be considered a security risk, this is not a problem, because the OSMOSYS framework detects that the Bluetooth companion is not within 1 or 2 meters, it locks all applications on the mobile phone and displays a lock on the screen. When Alice returns with her photocopies, the lock disappears and she can use her application as usual.

During lunch, a new appointment is pushed on her agenda, Mr. X must see her at 1pm before the meeting and requests a confirmation call. She calls to confirm. But now being late for her appointment, she forgets her phone on the table at the cafeteria. Fortunately, a colleague brings it to her at Mr. Xs office.

When she tries to use her mobile phone again, she notices that OSMOSYS closed itself to prevent any unauthorized access. This is because the mobile was no longer in range of the companion. All data on the smartphone have also been wiped out. To restore her contacts and meetings, Alice restarts Osmosys, logs onto the server using her user name and password and all her personal data are then automatically restored on the device.

A moment later, when the meeting takes place, she meets a new VIP to be entered on her contact list. At the end of the meeting, on her way back to the hotel, her mobile phone and purse are stolen from her bag. She realizes this only an hour later in the hotel because a message was waiting for her. The phone has been detected as stolen when it lost its connection with the companion., thus OSMOSYS cannot be restarted anymore. In any case, Her SIM card has also been blocked. She must go the premises of their subsidiary to destroy the companion and pick up a new phone and a new companion. A few manual steps in the presence of the local administrator and she will be able to continue her stay and work.

This scenario is not intended to point out all the requirements of a secure framework. Nor does it mention all possible ways to authenticate a user. Furthermore, theft detections via the help of a Bluetooth companion is only one way, although a very promising one. Furthermore, different users might require different security levels.

The OSMOSYS framework aims to offer a convenient way to manage these levels for each user. However, to ease the work of the administrator, different users will be managed as part of different user groups. Security parameters and settings will be defined for each user group.

The reader can imagine the above scenario from the administrator's view point to identify some requirements on the sever part of the OSMOSYS framework. A secure connectivity to the security server, e.g., via SSL, has been implied in the above scenario. Short loss of connectivity is not an issue, but the

user name/password authentication requires in our mind a connection to an authentication sever and the client application must authenticate the server.

OSMOSYS deals with this inconvenience using a so-called "robust session" concept. It is up to the security policy of the company to decide whether or not a session may persist in absence of a server connection, and how long the session is allowed to last without any connection to the company's security server.

The next sub-section defines the use case describing the functionalities required to enable scenarios such as the one above.

3.2 The Use Case

There are two main actors: the end user and the security administrator.

The end user uses the applications after being identified and authenticated. Then, existing applications can be upgraded or new applications can be installed.

The security administrator manages:

- the user and user groups,
- the applications (new installation and upgrade),
- the mobile devices,
- the security parameters for each user groups,
- the authentication companion (typically the Bluetooth companion, but possibly a secure card, a list of numbers or of printed code, etc.),
- the remote wipe of the applications and all relevant data.

Evidently the security manager must also be authenticated to use the security server which is located in the company's intranet. To simplify this authentication, we consider here only a central security server. Conventional security solutions such as a restriction to a physical log-in on the security server (Administration console in Figure 1) can solve the main security issues.

The user authentication can be based on several non-exclusive methods. In the considered framework, three authentication methods have been currently defined:

- formal: based on a user name and associated password; as an option the picture of a "turing number" can be used to deter non-human hacking,
- material: typically based on the unique identifier of the mobile phone (International Mobile Equipment Identity: IMEI),
- external: typically using a Bluetooth companion. However, other examples can be devised from a simple paper list of numbers, to secure card displaying response code (in numeral form or to be read by the camera of the phone).

3.3 The Authentication Services

The use case presented here can easily be extended to combine additional authentication techniques. For example, several PC brands as well as a few smartphones can scan a finger print. This kind of biometric identification could be used in addition to other external authentication. Also, the geographic location (via GPS, cell-ID or other means) can play a role in the settings of the security parameters. For example, within the headquarter area, the periodicity of the formal

re-authentication is set to its maximum, e.g., 24 hours. Another example is the use of the network connection: while roaming under a foreign operator, external authentication could be verified periodically every 5 seconds, but reported only every 10 minutes to the security server; if a WiFi connection is used then reporting will be more frequent, e.g., every 10 seconds. Again, these authentications techniques can be inserted within the external authentication methods.

The examples mentioned above are provided to illustrate some possible combinations of the authentication methods. When one authentication fails, the procedure is to protect the OSMOSYS applications and sensitive data. The chosen procedure depends on which authentication failed. The OSMOSYS framework offers a set of pre-defined procedures: for example, after the failure of the IMEI authentication and depending on the security settings, the user is either warned or not, the OSMOSYS framework is closed and a flag is set on the security server.

Since many security managers will request their own procedures, the implementation of the OSMOSYS framework should be as open as possible to be easily customized.

Beyond authentication and theft detection, the security framework has also to deal with possible attacks. These features will be not be detailed here. Various solutions have been proposed [2,9] and might be used in the future. Currently, a pragmatic approach was taken and the OSMOSYS framework focuses on providing strong authentication, detecting theft and securing all parameters so they can hardly be tampered with. The current solution is to store all security parameters only on a security server and to download them via a secured HTTPS connection. The downloaded values are then ciphered when stored on the mobile phone. It is recalled that the offline solution is also using the same ciphering procedure to store the data.

The end user must also be insured that the applications used are to be trusted. Thus, the OSMOSYS framework provides this service as explained in the following section.

4 Architecture of the Plaform

The architecture is based on a conventional client server scheme; this choice is mainly dictated by the convenience to have somewhere a centralized administrative tool that controls the features mentioned in Section 3.2. This naturally speaks for the introduction of a centralized server, which will also grant the isolation of the mobile access network from the corporate network. The overall client-server architecture is shown in Figure 1. The security and applications servers on the servers' side can obviously run on the same physical machine.

To achieve the goal of being able to check the integrity of an application, and to make sure that a modified application could not access valuable data within the corporate network, an obvious solution is to divide every application into server and client stubs. Defining a very strict, although flexible, interface between client and server part of the application insures that a modified application will not be able to access data outside the scope of the application. To further enhance

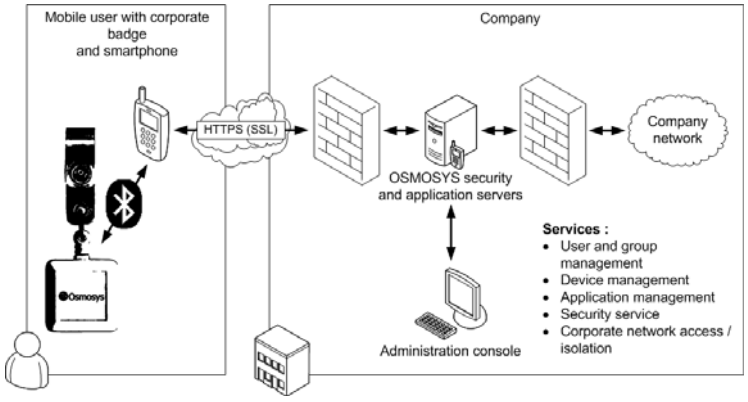

Fig. 1. The client-server architecture of the Open Secured Mobile System (OSMOSYS). A companion device is linked to the client to ease theft or loss detection. The server side is logically composed of a security server and an application server.

security, applications are systematically signed, and authenticated by the server stub. The security and application servers are detailed in Figure 2.

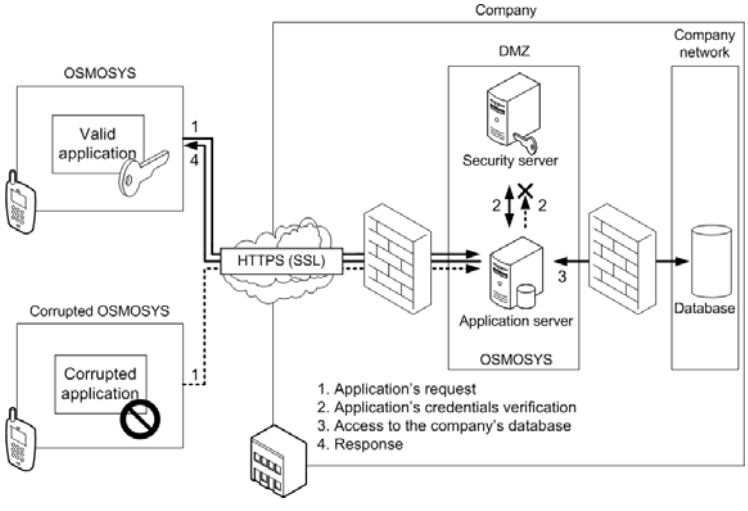

Fig. 2. Details of the client-serveur interaction to access credentials and secure an application: the application server verifies the validity of the requests on the security server before executing them

The client application also checks the server's identity before beginning any transactions; beside this check, it also authenticates the user and the companion device. Authentication is always performed with the assistance of the security server. When the server is unreachable, it will depend on the local security

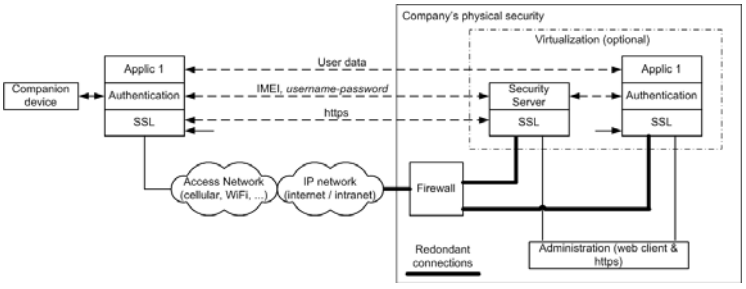

Fig. 3. Architectural view of the OSMOSY framework from a security point of view

settings wether the protected applications are usable or not. The most stringent conditions will require a permanent secure connection to the security server. In this case, a temporary loss of connection due to radio coverage problems for example, would automatically lock the secured OSMOSYS applications until a connection can be re-established by the client.

4.1 Risk Analysis

To analyze the proposed OSMOSYS architecture against the possible threats, the Figure 3 clarifies the client-server interactions. This section presents a brief evaluation of the risks to the OSMOSYS system.

– Write access to the database of the Security Server: This is probably the greater risk, but the security server will usually be physically protected within the company. The access to the servers is then controlled at the physical and administrative levels.

– Unauthorized user: Strong authentications can be based on using material: IMEI (mobile device), formal: User Name, Password, and external (Bluetooth companion). A mix of these authentication techniques can be adapted to the customer's requirements.

– Server authentication: The security server is authenticated via PKI. Therefore, phishing attacks are mitigated via careful PKI implementation and testing.

– Access rights: Each user is only authorized to use his own set of of applications according to his user group profile.

– Stolen mobile: Sensitive user data (contacts, messages, ...) are ciphered at the application level by the OSMOSYS framework. Furthermore, if the mobile is separated from the Bluetooth companion, the OSMOSYS framework will block, and a moment later close, all secured applications.

– Eavesdropping: The confidentiality of all exchanges (mobile-server and administrator-server) are insured via SSL (AES128). The Bluetooth connection to the companion should always be ciphered.

– Data integrity: SSL provides the data integrity service.

– Denial of service (DoS): On the server side, redundancy links and a restriction to intranet access are used to minimized the risks. On the mobile side, the DoS

risks on the access network are considered to be small. The DoS via Bluetooth will currently close the OSMOSYS framework.

– Tracability: Logs are generated on the security server. Logs on the mobile devices were considered only for debugging in order to simplify the mobile phone software and to prevent any security risk.

5 The Open and Secure Mobile System

To validate the concept, a practical system has been set up corresponding to the above specifications. First, an early prototype has been built on a J2ME client, and a JEE server. Although the concept could be effectively validated on this base, J2ME proved to be too restrictive for low-level control, thus a production-ready version has been developed using Microsoft's .NET framework (C# managed code) on the client, but still keeping the JEE environment on the server side.

The OSMOSYS framework is an application, from .NET's point of view, and thus does not rely on Windows Mobile 6 security infrastructure. Instead, it implements its own secured environment at the application level.

5.1 OSMOSYS Client and Companion

As stated above, the client device is a conventional WM6 smartphone. Installation of OSMOSYS is easy: open Internet Explorer, write the URL of your corporate OSMOSYS server, and click on the installation icon on the displayed page. The network administrator should have first registered a few data such as the IMEI and the Bluetooth address of the companion device. This condition allows to authenticate the smartphone and its Bluetooth companion device from the server side. From this point, OSMOSYS takes all necessary measures to insure that

– the user has the latest OSMOSYS framework version,
– the user can use the applications that are defined for him (his user group),
– the user has the latest software versions of the applications

From now on, the user may use OSMOSYS and the corresponding applications as long as the company's security requirements are met. Figure 4 show the OSMOSYS client on a smartphone after a successful authentication, before any user application has been launched.

5.2 OSMOSYS Server

The server is a conventional JEE application server. The current version has been deployed on a Tomcat server with EasyBeans as EJB container, and the administration dialog is implemented using conventional JSP technology, but this configuration is in no way a must. Any JEE implementation (capable of running servlets and featuring an EJB container) could be used. The server is divided into an application server and a security server; currently both run on the same machine. The role of both servers is well defined:

Fig. 4. Screen shots from the OSMOSYS server (behind) and client (mobile in front)

- The security server deals with authentication and security parameters concerning the users, user groups and devices (Figure 4),
- The application server deals with the server-side of the applications.

Any application built on top of OSMOSYS automatically inherits the security mechanisms built into the framework; if a new application has to be developed, it will be separated into a client part and a server part that share a common communication mechanism. All security and communication issues are kept hidden by the framework : the applications have no way to short-circuit the framework, so they are commited to use OSMOSYS's rules defined by the framework.

As every application has a client and a server part, it is very unlikely that a client could be modified and still work with the server counterpart, provided its signature could have been kept unchanged or faked by some way.

Special care has been taken to allow the full control of the whole system from the server's administration console. It is very unlikely that an end user has to install anything on the smartphone before using OSMOSYS. It is however possible that for security reasons, some companies would require to uninstall some or all pre-installed applications. It remains to be investigated whether or not this uninstallation can be performed remotely from the security server.

6 Conclusion and Future Work

Some important requirements to secure applications and sensitive data on smartphones have been presented. A rich client-server architecture was proposed to authenticate the user, detect loss or theft with the help of an additional companion device, and to secure the use of end-user applications. The security features include strong authentication, data ciphering, and management of the life cycle for any customized application. The framework named OSMOSYS was described to offer these various security services.

An implementation of OSMOSYS using Windows Mobile 6 demonstrated the feasibility of our approach. The security server has been audited by the last author and no flaw has been found. The OSMOSYS product is currently available as a public service for evaluation and testing (see www.securemesa.ch). At the time of writing, a private bank in Geneva is evaluating the service under realistic conditions. Preliminary results of this field trial will be presented at the conference.

References

1. Nicholson, A.J., Corner, M.D., Noble, B.D.: Mobile Device Security Using Transient Authentication. IEEE Transactions on Mobile Computing 5(11), 1489–1502 (2006)
2. Desmet, L., Joosen, W., Massacci, F., Naliuka, K., Philippaerts, P., Piessens, F., Vanoverberghe, D.: A flexible security architecture to support third-party applications on mobile devices. In: Proceedings of the 2007 ACM Workshop on Computer Security Architecture. CSAW 2007, Fairfax, Virginia, USA, pp. 19–28. ACM, New York (2007)
3. Wu, H., Grgoire, J., Mrass, E., Fung, C., Haslani, F.: MoTaskit: a personal task-centric tool for service accesses from mobile phones. In: Proceedings of the 1st Workshop on Mobile Middleware: Embracing the Personal Communication Device. MobMid 2008, Leuven, Belgium, pp. 1–5. ACM, New York (2008)
4. The BlackBerry solution allows users to stay connected with wireless access to email, corporate data, phone, web and organizer features, `http://www.blackberry.com/`
5. Hoffman, D.V.: Blackjacking: Security Threats to BlackBerry Devices, PDAs and Cell Phones in the Enterprise. Wiley, Chichester (2007)
6. Windows Mobile 6, `http://msdn.microsoft.com/en-us/library/bb158486.aspx`
7. Understanding the Windows Mobile security model, `http://technet.microsoft.com/en-us/library/cc512651.aspx`
8. Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring integrity on mobile phone systems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies. SACMAT 2008, Estes Park, CO, USA, pp. 155–164. ACM, New York (2008)
9. Dunham, K. (ed.): Mobile Malware Attacks and Defense. Elsevier, Amsterdam (2009)