ASSESSMENT OF MOBILE SECURITY PLATFORMS

Germán Retamosa and Jorge E. López de Vergara
Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain

Keywords: Mobile Security, Mobile Malware, Security Platforms, Windows Mobile, Symbian OS, iPhone OS.

Abstract: Mobility is one of the most important features in users' communications. The increasing progress carried out by mobile communications has changed the concept of mobile phone. Nowadays, each new device offers newer technologies and services adapted to user requirements. However, each added improvement has a set of threats, which have to be taken into account for ensuring the confidentiality, integrity and availability of user and system data. In this paper, we analyze some important mobile security platforms, such as *Symbian OS*, *Windows Mobile* and *iPhone OS*. Then, we compare these platforms giving some advantages and drawbacks of each one and providing some conclusions about this study.

1 INTRODUCTION

The progress carried out by mobile phone manufacturers is a present reality, and theirs trends are growing as the days go by. Furthermore, this evolution has some advantages like newer features adapted to current user requirements. However, it also implies some difficulties, which have to be taken into account, such as the creation of a trustworthy mobile platform, which ensures user privacy and security. The trustworthiness of these systems must check the three main features on any computer security: confidentiality, integrity and availability. However, ensuring user privacy is a difficult task, which has to be carried out throughout the software lifecycle, from analysis to development stage. For this reason, mobile manufacturers have created security divisions for implemented platforms, which protect users from different attacks. But all these security mechanisms are not completely effective, either because of the effort that it implies or simply due to design failures. The goal of this study is to analyze and evaluate the trustworthiness of some important mobile security platforms.

The rest of this paper is divided into three sections: the first one depicts an analysis of some relevant mobile security platforms such as *Symbian*, *Windows Mobile* and *iPhone OS*. Then, the second section uses these obtained results for describing an evaluation process, which sums up with some positive and negative aspects of each studied platform. Finally, the third part finishes with some final conclusions obtained from the evaluation process.

2 MOBILE SECURITY PLATFORM ANALYSIS

The wide mobile security range includes several types of mobile operating systems. Each one has its own features and examples, described below (Heath, 2006):

Open Platforms enable the access to native interfaces, and usually provide an SDK (*Software Development Kit)* for accessing to these interfaces and, develop third-party applications. Despite all these advantages, these systems are the most vulnerable due to the wide range of access to system capabilities and, for that reason, are the main target of any attacker. *Symbian OS*, *Windows Mobile (Compact Edition)* are examples of this type of systems.

Layered Execution Environments are considered closed environments which deny access to native interfaces, although sometimes enable access on some restrictive circumstances, named sandboxed environments. Moreover, these systems usually provide an SDK (*Software Development Kit)* for developing third-party applications. *iPhone OS* and *Google Android* are examples of these types of systems.

Closed OS denies access to native interfaces, and usually does not provide an SDK (*Software*

Retamosa G. and E. López de Vergara J. (2009).
ASSESSMENT OF MOBILE SECURITY PLATFORMS.
In *Proceedings of the International Conference on Security and Cryptography*, pages 127-132
DOI: 10.5220/0002261101270132

Development Kit) to develop third-party applications and so, the security platform is not known for any attacker, unless reverse engineering techniques can be used. For these reasons, it is very difficult to develop malware on this type of systems. An example of this type is the *LG Proprietary OS* (LG Electronics, 2009). In response to previous classification, layered execution environment and closed platforms seem to be more secure than open platforms but in fact, both are also exposed to many attacks such as *denial of service*. The following study focuses its efforts on analyzing open platforms and layered execution environments for providing user some security knowledge about these platforms.

2.1 Symbian OS

Currently, *Symbian OS* (Symbian, 2009) is the most known and used operating system on the mobile market. *Symbian OS* is a single-user and multitask operating system with EKA2 (*EPOC Kernel Architecture 32bits)* architecture and some specific features (Savoldi and Gubian, 2008):

- Modularity.
- RTOS (*Real-time Operating System)* nanokernel.
- Priority-based pre-emptive multitask.

Single-user feature reduces considerably the complexity of security platform, achieving a lightweight platform, which satisfies common mobile restrictions, such as effectiveness and efficiency. Furthermore, the security policies used on these systems are based on capabilities list. The following list enumerates some rules, which set up the basis of *Symbian OS* security platform.

- Each process must have resources and access privileges list, which allow the access to some system capabilities, which are invariants during their execution.
- Each process must have its own virtual address space protected by the operating system.
- Each application signature gives access to only one system capability, although the operating system allows multiple signed applications.
- Each process can only use shared libraries (DLL, *Dynamic Load Library*) with at least the same level of trust.

Figure 1 shows the *Symbian OS* security architecture, which is divided into four different tiers.

- TCB, Trusted Computing Base.
- TCE, Trusted Computing Environment.
- Signed software.
- Unsigned software.

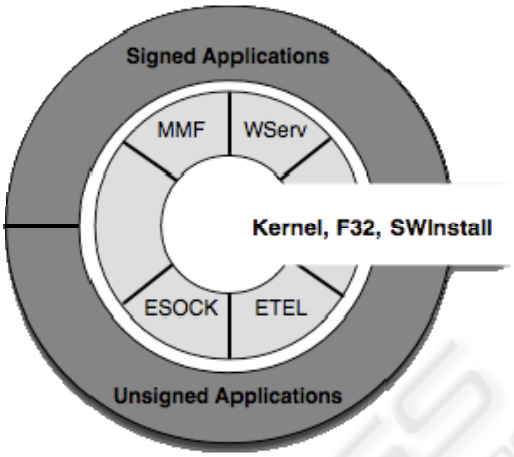

Figure 1: *Symbian OS* Security Architecture (Heath, 2006).

TCB is the core component of the security platform and so, the most trusted part, which controls the lowest level of security mechanisms. It includes the components described below.

- Kernel: Assign privileges on process creation.
- File Server (F32): Load program code on process creation.
- Software Installer (SWInstall): Install SIS (*Symbian OS Install Script)* files, including software validation.

TCE consists of a set of servers with specific features. Each server is independent from the rest, and needs a set of specific privileges to access it. **Signed software** is a set of trusted applications which have enough rights to access TCE interfaces, and **unsigned software** is a set of untrusted applications which does not have rights to access TCE interfaces. *Symbian OS* capabilities are a set of privileges installed on the system. In particular, it has twenty capabilities divided into three groups: TCB, user and system. Finally, the last component of the *Symbian OS* security platform is the data caging, which is used to protect the integrity and confidentiality of the system. There are three special containers protected by data caging: `\sys`, `\resource` and `\private`.

Figure 2 describes the process followed by the security platform on most of the software installation process. Firstly, SWInstall verifies the presence of signatures in the application. If so, TCB and TCE enumerate the signature or signatures and then, classify them into the different system servers. If not, the application is unsigned, so it will not have privileges.

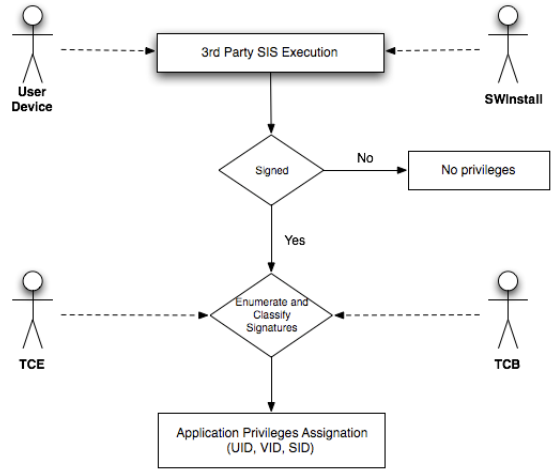

Figure 2: *Symbian OS* Execution Process.

2.2 Windows Mobile

Windows Mobile (Microsoft, 2009) is technically a version of *Windows Compact Edition CE*. This system is based on the *Win32 API* and *.NET Compact Framework*, and it is optimized for several mobile processors, such as *XScale* or *ARM* (*Advanced RISC Machines*). However, it is not a homogeneous platform because it is divided into three different models according to the different type of devices existing on telecommunications market, *Standard SDK* for Smartphones, *Professional SDK* for PDAs (*Personal Digital Assistant*) and *Classic SDK* for PDAs with no phone. Taking this disposition into account, this security platform is the sum of a set of security policies, roles and certificates. *Windows Mobile* supports three different types of security policies defined on system registry (`HKLM\Security\Policies\Policies`), which set up the access permissions to the file system (Microsoft, 2007)(Moreno, 2008):

- Privileged or Kernel Mode. It allows read and write access to the file system.
- Normal or User Mode. It allows full access to personal data areas and denies access to protected zones.
- Blocked. No access at all.

Security roles are the components responsible to manage access to system resources, and it is especially used for provisioning purposes. Furthermore, certificate management is one of the most important features because it provides confidentiality to user data and manages the execution of an application and the assignation of its respective access rights. All certificates are saved on certificates stores, each one with different privileges and access rights. Finally, *Windows Mobile* defines five security levels, which are listed below (Microsoft, 2007) (Moreno, 2008):

- Security Off (only for testing purposes).
- One-tier prompt (see Figure 3).
- Two-tier prompt (see Figure 4).
- Mobile2Market locked.
- Locked.

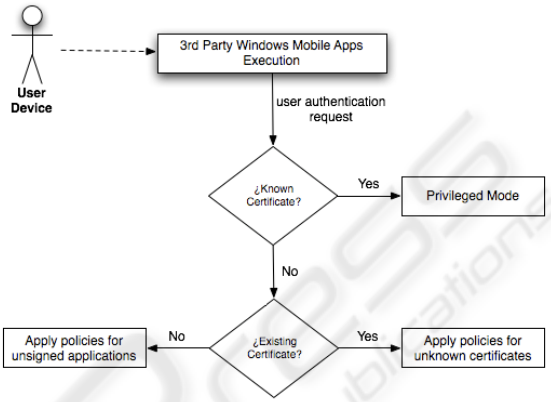

Figure 3: One-tier Prompt Procedure.

Figure 3 shows the installation process followed by the *one-tier* security platform. This type of security platform is used on the *Professional SDK* and the *Classic SDK*. Firstly, the certificate manager checks for presence of certificates in the application. If certificates cannot be found, it will apply security policies for unsigned applications, elsewhere certificates get validated. It will apply privileged mode capabilities if the certificate is known, otherwise it will apply security policies for unknown certificates.

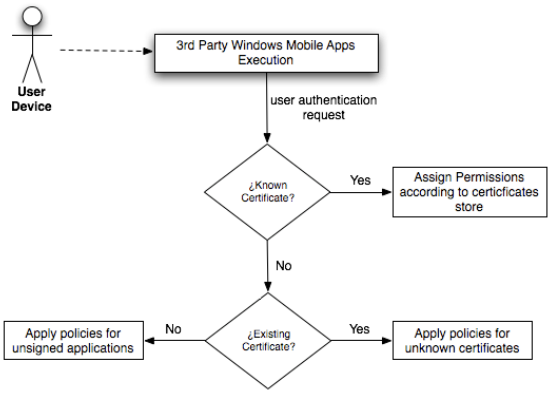

Figure 4: Two-tier Prompt Procedure.

Figure 4 shows the installation process followed by the *two-tier* security platform. This type of security platform is used on the *Standard SDK*. This procedure is analogous to the *one-tier* installation process, but instead of assigning privileged mode

capabilities for known certificates, it assigns permissions according to the certificates store.

2.3 iPhone OS

iPhone OS (Apple, 2009) is the last analyzed operating system. It is widely known as the system designed by *Apple Inc.* for its main mobile phone, *Apple iPhone*. This system is a single-user and multitask smaller version of *OS X* enhanced for ARM processors and based on the Mach kernel. Its origins are relatively newer with respect to other mobile operating systems such as *Symbian OS* or *Windows Mobile*. Figure 5 shows the main architecture of *iPhone OS* and its security services. The main architecture is divided into five different and isolate tiers, which are listed below, although this study will only analyze the security platform included in the Core Services (Apple, 2008):

- User applications.
- Cocoa Touch.
- Media.
- Core Services.
- Core OS.

In addition, there are some security services, which define the security platform (Apple, 2008):

- CFNetwork.
- Keychain Services.
- Sandboxing Services.

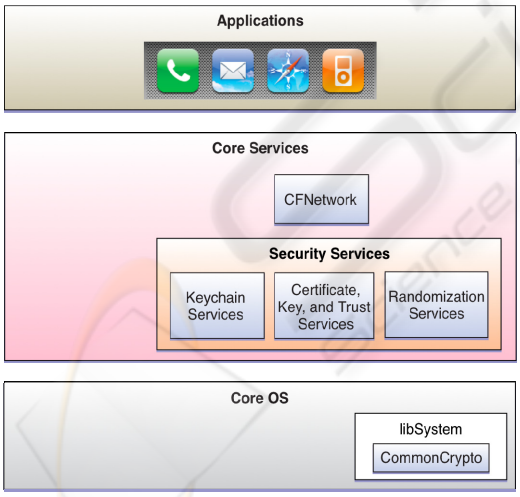

Figure 5*: iPhone OS* Architecture (Apple, 2008).

CFNetwork is the component responsible for establishing secure communications between *iPhone* and any external device connected to Internet throughout TCP/IP by using protocols such as SSL (*Secure Socket Layer*) or TLS (*Transport Secure Layer*). **Keychain services** are responsible for keeping secure user and system data storage by ensuring the confidentiality, integrity and availability. These features are based on encryption/decryption interfaces and certificate management. Its usage is very important: it protects the personal data from any possible attacker.

However, the access to keychain services and root certificates management are driven by the *iPhone OS* security server daemon. For security reasons, the daemon API (*Application Program Interface*) is not accessible to software developers. **Sandboxing services** are a fine-grained access control system whose main task is to protect the resources of the system. This type of system is implemented with MAC (*Mandatory Access Control*) and every installed application is inside of a sandbox environment. Therefore, with this security countermeasure, the file system is protected against some attacks, or misuse. However, there are some other attacks, such as *buffer/heap overflow*, which can avoid this restriction and it is responsibility of the application designer to create secure applications, which protect this type of attacks or malicious applications (Zdziarski, 2008)(Miller *et al.*, 2007).

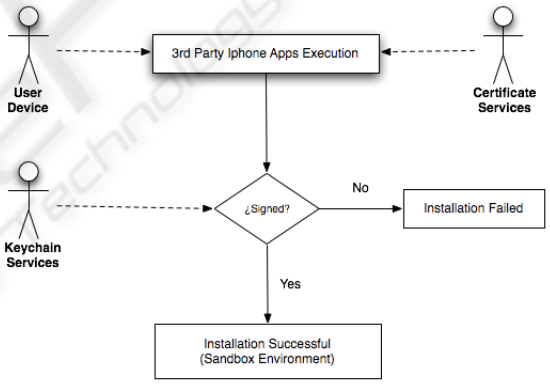

Figure 6*: iPhone OS* Execution Process.

Figure 6 describes the steps followed by the *iPhone OS* security platform during the installation of a *third-party* application. Firstly, the certificate manager and the keychain manager checks for presence of certificates into the application. If so, the installation process will finish inside of a sandbox environment if the keychain manager validates the existing certificate, otherwise the installation process will fail. Finally, the last security feature is a simple data caging procedure. The disk drive is divided into two partitions, and one of them contains the kernel file system with read-only permissions. Therefore, this countermeasure protects the file system from malicious attacks.

3 MOBILE SECURITY PLATFORM ASSESSMENT

In the previous section, three of the most important mobile security platforms were analyzed for better understanding of the topics discussed below. This section will define the most known vulnerabilities and weaknesses of these operating systems by using the following methodology schema:

- Implementation Failures.
  - File System Failures.
  - Native Application Failures.
- Communication Failures.
  - Bluetooth Failures.
  - MMS (*Multimedia Messaging System*) Failures.

3.1 Symbian OS

Nowadays, *Symbian OS* is considered the most attacked operating system on the market, followed by far by *Windows Mobile* (Heath, 2006). This experience is translated into a robust security platform with a really small number of vulnerabilities.

Implementation failures, also known as *bugs*, are one of the most important loopholes in any operating system. However, the wide range of this attack vector exceeds the scope of this assessment, and it will only analyze file system and native application failures, which have been considered the most relevant and harmful inside of a mobile phone. Currently, the number of known *Symbian OS* implementation vulnerabilities is really small or non-existent.

Communications failures are the most recent loopholes in any operating system, because the growing evolution of the mobile phones involves the creation of weak communications protocols adapted to user requirements. **Bluetooth and MMS failures** are the most known *Symbian OS* failures, but they need to use social engineering techniques for exploiting the mobile phone. The problem of the social engineering techniques is that the mobile phone user carries all the liability and the security platform cannot perform any countermeasure. Furthermore, Bluetooth attacks have a wide range of varieties listed below (Gostev, 2008):

- Bluetooth Propagation (*Cabir/29.A*).
- Operating System Infection (*Skulls*).
- Phonebook Access and MMS Propagation (*Commwar*).

3.2 Windows Mobile

Windows Mobile is nowadays one of the most important mobile operating systems. This importance is due to *Microsoft Windows* predominance on the computer market and the current growing number of devices with this platform.

File system failures are the most dangerous case of implementation failures because their presence are extended to all mobile phones with the involved operating system, but they are also the most cost-effective to fix because only one patch release can be applied to all the involved operating systems. A sample of this type of vulnerability is described below:

- Bypassing the code-signing protection in *Windows Mobile* is relatively easy because, once a mobile phone user accepts the installation of a normal application, provided by Visual Studio but used with malicious purposes (`SDKCerts.cab`), any installed application signed with a special certificate (`SDKSamplePrivDeveloper.spc`) will have full privileges (Moreno, 2008).

Native applications failures are the existing implementation failures in each system application and excluding *third-party* applications. These failures are less dangerous than the previous type, but the profitability to fix these failures will depend on the number of existing application instances. If a majority uses an application, the released patch will be cost-effective. Some samples of these failures are the numerous cases of *denial of service* in *ActiveSync* and PIE (*Pocket Internet Explorer*) (Dunham, 2009).

Bluetooth and MMS (Bluetooth, 2009) (MMS, 2003) are considered one of the main communication failures in any operating system, but in fact, it is commonly used for spreading malicious applications. However, Bluetooth is vulnerable to *driver-level* attacks and *BD_ADDR Spoofing* attacks (Moreno, 2009) and MMS has some vulnerabilities via shellcode-based applications in MMS Client and Composer (Dunham, 2009).

3.3 iPhone OS

As we discuss in the previous section, *iPhone OS* is a single-user operating system. However, the only system user is the root account and all the applications are running with no restrictions.

Implementation failures are the only existing failures in the *iPhone OS*, because the communications protocols are very restrictive and

theirs user interfaces are too limited. Currently the most known implementation failures are the file system failures and they are described below (Zdziarski, 2008).

Root account involves that any system failure or malicious application instantly gets a privileged mode. For this reason, Apple Inc. restricts extremely the third-party installation process.

Static addressing means that all the application instances have the same memory address, instead of the randomization methods used in the majority of mobile operating systems.

Static systems means that currently, there are only two phone models of Apple iPhone based on the same hardware and software. This feature is useful for an attacker because all the malicious programs will have the same pattern.

Finally, earlier firmware versions of the *iPhone OS* (1.1.4 – 2.0) had some native applications failures in the Safari WebKit interfaces. These failures exploit the mobile phone with *denial of service* attacks.

4 CONCLUSIONS

The obtained conclusions from this study show that the maturity of the operating systems is a relevant point to take into account because it is inversely proportional to the number of vulnerabilities and therefore, it can be used such as a security estimator. As we discuss in the previous section, *Symbian OS* is the most attacked platform and it has the smaller number of vulnerabilities. In fact, *Symbian OS* has the most secure execution procedures, followed closely by *Windows Mobile*. Moreover, both platforms have a normal level of security on installation procedures based on code-signing and certificates management. On the other hand, *iPhone OS* is the least secure operating system because their execution procedures are too permissive, and these techniques waste all security efforts placed on the installation procedures.

Finally, Table 1 shows a resume of the main differences between each studied security platform.

In future works, we are going to research about advanced methodologies, which enable the automatic security assessment of any mobile operating system. In addition, to improve the existing procedures, we will research on different mobile malware classification techniques.

Table 1: Mobile Security Platforms Overview.

	Symbian OS	**Windows Mobile**	**iPhone OS**
Installation Security Level	Average	Average	High
Execution Security Level	High	Average	Low
Certificates Management	Average	High	Average
Communication Protocols Security Level	Low	Low	Average
Security Policies Management	Average	High	Low
Data Caging	High	Average	Low

REFERENCES

Heath, C., 2006. Symbian OS Security Platform. Wiley.

Savoldi, A., Gubian, P., 2008, Symbian Forensics: An Overview. *IIHMSP '08.*

Zdziarski, J., 2008, iPhone Forensics. O'Reilly.

Miller, C., Honoroff, J., Mason J., 2007. Security Evaluation of Apple's iPhone. Independent Security Evaluators.

Apple Inc., 2008, Security Overview.

Microsoft, 2007, Security Model for Windows Mobile 5 and Windows Mobile 6.

Moreno, A., 2008, Windows Mobile Security Model, http://www.seguridadmobile.com/windows-mobile/seguridad-windows-mobile/ejecucion-aplicaciones.html

Moreno, A., 2009, BD_ADDR Spoofing, http://www.seguridadmobile.com/bluetooth/seguridad-bluetooth/BD_ADDR-spoofing.html.

Symbian Software Ltd., 2009, http://www.symbian.com/index.asp

Microsoft, 2009, http://www.microsoft.com/windows-mobile/ es-es/default.mspx

Apple Inc., 2009, http://developer.apple.com/iphone/

LG Electronics, 2009, http://www.lge.com/

Dunham, K., 2009. Mobile Malware Attacks and Defense. Elsevier.

Bluetooth SIG, 2009, https://www.bluetooth.org/ apps/content/

Gostev, A., 2008. Mobile Virology Introduction. Hakin9.

MMS, 2003, 3GPP MMS Specification, http://www.3gpp.org/ftp/Specs/html-info/23140.htm