A Secure Software Architecture for Mobile Computing

Hassan Reza, and Narayana Mazumder
School of Aerospace Science, University of North Dakota

Grand Forks, USA 58201

reza@aero.und.edu

Abstract. *Mobile computing is considered as low-end computing systems. It currently dominates almost all aspects of our lives from mobile banking to mobile purchasing to mobile skypeing, etc. With the increasing demand of smartphones in today's world, making the system secure is very important. Software architecture plays an important role to achieve both functionalities and quality (e.g., security) of a system. This paper surveys the software architecture of five of the leading mobile operating systems such as Android (Linux), Blackberry, iPhone (Mac OS X), Symbian, and Windows Mobile. Our survey study indicates that Android is the most promising of al and identifies security issues with Android. The paper then proposes improved software architecture to resolve these issues. More specifically, our proposed software architecture attempts to enhance the security of Android without comprising it performance.*

Key Words: *Software Architecture, Security Engineering, Mobile Computing, Android, Operating Systems.*

1. Introduction

The demand for Smartphones is increasingly growing at an exponential rate; it is expected to soon pass laptops' sale as a device for mobile computing [9]. Initially, cell phones were only used to perform simple tasks such as making phone calls and/or exchanging text messages. Currently, cell phones are becoming more sophisticated by providing functionalities that once expected from laptop and/or desktop computing systems. For example, using cell phone, callers can now interact with system using spoken language, brows internet, exchange emails, chat online and social network medias, use navigation systems, etc. Further, new and unimaginable capabilities can be incorporated in the mobile computing at any time and rate using new apps.

Performance and security are perceived paramount qualities for the modern mobile computing systems.

Mobile systems (e.g., smartphones) do not have the computing capabilities attributed to desktops or laptops. As such, one cannot implement everything in a smartphones that can be implemented in full-fledged traditional computing systems. Some on the limitations of smartphones are as follows:

- Smartphones run in battery and has significant power constraint.
- Smartphones doesn't have the same CPU power as a computer.
- Smartphones have limited RAM as compared to a computer.
- With the implementation of 3G network speed, the internet download and upload speed of the cellular networks has shown great improvement.
- Screen size plays a major role and is a very important issue which needs to be taken into consideration while building an operating system.

In the next section, a brief background of the surveyed operating systems is given together with their comparison table. Using analysis result, two of the leading operating systems are selected and their architectures are explained in section 3. The shortcomings and security risks of the selected architecture are discussed in section 4. Sections 5 and 6, discuss our proposed enhancements together with their architectural solutions.

2. Background and Comparison

Blackberry: Blackberry was introduced by Research In Motion (RIM) in 1999 which started

978-0-7695-4654-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ITNG.2012.122

as a simple two-way pager, but it quickly became one of the most widespread of the mobile devices [6].Blackberry software platform has a layered architecture. It was the first mobile operating system for smartphones.

Symbian: In 1998, Nokia, Ericsson, Panasonic, and Samsung got together and collaboratively created single operating system to run their devices [4].

iPhone: Apple announced iPhone at the MacWorld expo in San Francisco in January 2007 [1]. The iPhone was the first smartphone which had huge popularity amongst the general users and it quickly snatched some of the market share from RIM and Symbian and it quickly became a status symbol [4].

Android: Currently**,** Android is the most popular operating system out of the several Linux based mobile operating systems (e.g.,Maemo) [6]. In November 2007 Google released the operating system Android under the Open Handset Alliance with the goal of being an open source platform for software development on mobile platform [1]. Android software architecture is based layered architectural style [20] [21]; Android provides a custom built virtual machine (Dalvik) for the applications to run; it also acts as the middleware between code and the main operating system [4]. Table 1 summaries our survey results.

	Android	Blackberry	iPhone	Symbian	Windows Mobile
Network Scanning	S	Ps	Ps	S	S
Network Interface Selection	Ps	S	S	S	S
Bluetooth I/O	S	S	Ns	S	S
Network Interface Control	S	S	Ns	Ns	S
Background processing / Multitasking	S	S	S	S	S
Energy Monitoring	S	S	S	S	S
Power Saving	S	S	Ps	S	S
Low-level memory management	S	S	S	S	S
Location Sensing	S	Ns	S	S	S
Persistent storage	S	S	S	S	S
Openness	S	Ns	Ns	Ps	Ps
Security	Ps	S	S	Ps	Ps
Cost	S	Ps	Ns	Ps	Ps
Hardware Independence	S	Ns	Ns	S	S
Usability	S	Ps	S	Ps	Ps

S = Satisfied; Ps = Partially satisfied; Ns = Not satisfied;

Table 1: Summary of mobile platform categories

The result from the above table has been used to select the software architecture of two leading operating systems. The selection criteria depended on the availability and popularity of the operating systems. Android was selected as one of the contenders of the research because of the openness and extremely user-friendliness of the operating system. Apple's iOS was selected as the second contender of the research. One of the major reason behind selecting iOS as another contender is due to its current popularity and innovation in the design.

3. Architecture of iOS and Android OS

iOS Architecture: Apple's iOS is a sligthly different than Android, because Apple develops and controls both the end product and the operating system. Therefore, it is difficult to run Apple's iOS in any other device because of lack of access to source code. iOS [13] has a layered architecture and is shown in figure 1.

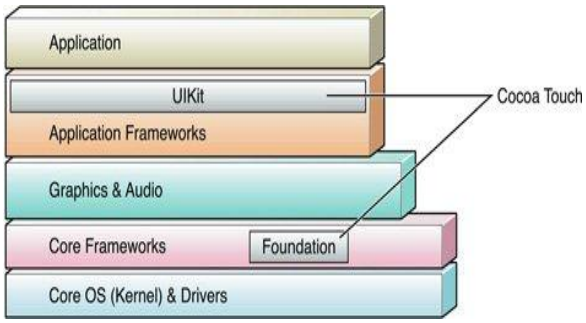

Figure 1: Architecture of Apple's iOS

Apple doesn't allow any third party developer to look into and work in any of the architecture level. The architecture is much closed from the outside world. Developers can only interact through one little window called Cocoa. The main application development language is Objective-C [13].
Since, Apple controls the whole environment – from chip to application to final device, it does not

require a virtual machine. This is one of the major reasons why Apple's iOS cannot be installed in any other device. The operating system is totally dependent on the device. Thus, the portability of the software can be an issue with Apple's iOS [13]. This is the reason why it is becoming less popular nowadays as compared to Android – which can be installed in any device which fulfils the basic requirement of the system [11].

Android architecture: Android is s a software stack (see Figure 2) for mobile devices which includes an operating system, a middleware and key applications. Android SDK provides the tools and APIs necessary to develop an application using JAVA (which is a popular language amongst the developers. The main advantage of Android over Applie's iOS is that it is open-source and is based on *Linux* and is free to be used. Due the presence of the Dalvik virtual machine (discussed later) Android is fast; its hardware is independent, which makes it portable [12].

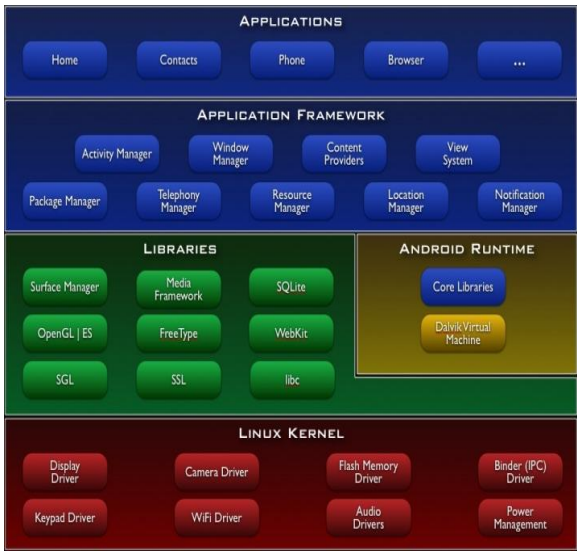

Figure 2: Architecture of Android [12]

Linux Kernel: Android is based on Linux but is not Linux. The kernel of Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. This is the abstraction layer between the hardware and the rest of the software stack [12]. This means that as long as any device satisfies this layer Android can be installed in that device.

Libraries: The surface manager of Android library takes care of the display of the system and OpenGL is an open-source utility which takes care of graphics of the system. The media libraries are responsible for playback and recording of audio and video formats. FreeType takes care of the font of the texts. The database engine of Android is SQLite and Android's browser is a Webkit browser which is another open-source utility [12].

One of the important parts of the library systems that plays a major role is our research is the Bionic 'Libc' of Android. Libc is a set of C/C++ library system. It is a custom libc implementation of general Linux libc, optimized for embedded use. The advantage of this is that it is smaller in size and has faster code paths that are perfectly suited for mobile devices. This makes the system fast and quick to respond to any request. One of the biggest disadvantages of this custom implementation is that it is not compatible with the standard Gnu Libc (glibc) [12].

Android Runtime: The development language that is used in this section is Java. The core Libraries of Android is very powerful, yet simple and familiar development platform as it is very similar to Linux.

Dalvik Virtual Machine (DVM): This is one of the most important reasons why Android is so fast and suitable for mobile systems. Dalvik virtual machine focuses on two of the most important issues of mobile system: limited space, and limited power. It also increases the portability of the system because it can be implemented in any mobile system and is independent of the hardware. DVM converts all the files into smaller and more optimized (.dex) suitable for mobile systems [12]. One of the interesting features of DVM is that it supports multiple virtual machine processes meaning that if two applications are running at the same time, they may work on two separate virtual machines. This feature can improve the performance of the system.

Application Framework and Application: The activity manager is responsible to keep track of life cycle of any application. For example, suppose a user is running an application and then she opens another application and minimizes the current application. She may forget about the minimized

application later on. Activity manager does a very good job by tracking the idle time of a minimized application and killing it when the system runs out of memory. The content provider takes care of the shared data between applications. Therefore, due to performance (speed), openness, and portability of Android, it can be reasonably expected that it will lead the modern mobile computing [11].

4. Security risks of Android OS

In the previous section we have discussed that the presence of Dalvik Virtual Machine (DVM) improves the performance of Android operating system. With the increasing complexity of the Android OS with each new release [12], it is becoming more capable to perform different tasks which may compromise security.

Security is a very important issue that was not given much considerations We have discussed earlier that the openness of Android makes it more popular amongst the developers and end-users. But, an operating system cannot be made open at the cost of the system's security. For example, Rootkits are malware that modify operating system's data to achieve malicious goals. In some cases, Rootkits can create even more devastating effects by spying on conversations via GSM [2]. Since, Android can be installed in any mobile hardware and for any carrier such as GSM and CDMA. As such, it is more vulnerable to these kinds of threats.

5. Enhancing security of Android OS

Android allows modifications even in the operating system level. The openness of the system can not only be a threat to the system but also can be used to protect the system from malicious intrusion. Since, a third party application developer has access to all the levels of software architecture; they can develop kernel-based low level security tools. In Android, malware detection techniques can be implemented even at the kernel level.

Linux is known for its security; there are security software available for Linux based system. Since, Android is based on Linux; one can test the applicability of existing signature and intrusion detection methods in Android environment. But, each and every method cannot be applied in Android because Android libraries have been modified to a large extent to make them suitable for mobile systems.

Detecting malicious application: The following procedure needs to be performed in order to detect a malicious software/application:

1. Identify executable files linking format executables installed on the Linux side of Android operating system (applications that are pre-installed in the system).
2. Perform a statistical analysis of the function calls by these applications and keep a record of the analysis in a log file.
3. Compare the result to the newly installed applications for detecting significant difference.
4. Create a decision tree to determine the level of suspicion.

In what follows, we attempt to find out if existing tools can be adopted for Android based systems. Since, Android is a Linux based operating system it is possible to incorporate security measures designed for Linux based desktop operating system. But, three important factors need to be taken into consideration: 1)the processing power and memory of a mobile system is very low as compared to a desktop/laptop; 2)mobile systems do not have enough space as compared to a desktop; 3) android's libraries has been highly modified to make it suitable for mobile systems.

The antivirus tool that was selected is Clam AntiVirus which is an open source (GPL) anti-virus toolkit for UNIX [16]. Although all of the parameters were not tested but for the most part, it works.

Netfilter is an open source firewall tool that has been selected for our purpose. It hooks inside the Linux kernel. This is possible to be implemented in Android OS as it is an open-source OS [17].

Chkrootkit has been chosen for this work. It scans for signs of worms, rootkits and Linux Kernel Module (LKM) Trojans. It inspects binaries, checks logs for suspicious interfaces and looks for hidden files [18].

Snort is a well-known Linux tool which is used for intrusion detection. It analyzes the traffic and packet logging on IP networks; it scans each and every port and checks for other suspicious behavior [19].

```
_edata = y
|  gethostbyname = y
|  |  sigaction = y: normal
|  |  sigaction = n: malicious
|  gethostbyname = n
|  |  fork = y
|  |  |  strerror = y
|  |  |  |  getgrgid = y: malicious
|  |  |  |  getgrgid = n: normal
|  |  |  strerror = n: malicious
|  |  fork = n: normal
_edata = n
|  exit = y: malicious
|  exit = n
|  |  fprintf = y: malicious
|  |  fprintf = n
|  |  |  uname = y: malicious
|  |  |  uname = n
|  |  |  |  execv = y: malicious
|  |  |  |  execv = n
|  |  |  |  |  malloc = y: malicious
|  |  |  |  |  malloc = n
|  |  |  |  |  |  putchar = y: malicious
|  |  |  |  |  |  putchar = n
|  |  |  |  |  |  |  memmove = y: malicious
|  |  |  |  |  |  |  memmove = n: malicious
```

Figure 3: Decision tree; 'y' means it appears in the static table of an executable; 'n' means that it does not.

6. Proposed Software Architecture

In the previous sections, we noticed that the Android's architecture has been designed to provide fast respond. For example, the inclusion of virtual machine and the fact that many virtual machines can be activated simultaneously, improves the performance significantly [12]. In this section we attempt to extend the architecture of Android to incorporate some security measures. Finally, we propose and modified layered architecture which makes the system fast and secure.

One way to make a system more secure is by incorporating some of the security aspects at architectural level [10] [21]. Though some of the above mentioned applications can be implemented in the Android operating system to boost security by detecting threats and mitigating against them, it is more cost effective to incorporate security mechanisms at architectural level [21] [10]. But we cannot slow down the performance of a system in order to make the system more secure.

In this section, we propose revised software architecture to enhancing the security without compromising performance of Android by implementing intrusion detection techniques in the system itself. Figure 4 shows the software architecture without security improvement.

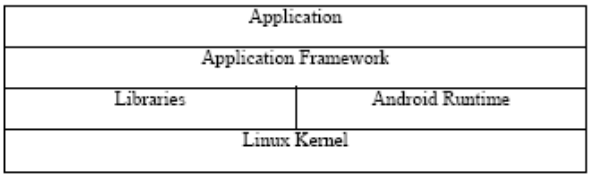

Figure 4: Skeleton of Proposed Architecture

Figure 5 shows the revised but improved software architecture of the Android to improve security. As can be seen from the figure, the proposed architecture includes a monitoring and a detection client security feature, and a control daemon.

It has a database interface which allows the third party applications to access the database. The application framework level includes all the functionalities needed to monitor any application's activity.

A detection manager is included at this level; it constantly monitors the behavior of all the running applications, which might include a malicious one. It also checks for any malicious behavior by any third-party application or from network.

The control daemon is responsible for constantly checking the status of the detection manager. The detection manager on the other hand, extracts information from Linux kernel and files systems that are stored as reference and compares it with the behavior of the third party application (discussed in 'Detecting Malicious Application'). For example, if it finds any anomalies and suspicious behaviors, it informs the control daemon about it. One needs to note that the detection manager itself does not do anything to the malicious application (if found). There are two major reasons behind this. First, if the detection manager itself tries to stop the malicious activity, it has to stop itself from monitoring the activities of other possible threats. Secondly, it will

significantly hamper the performance of the system.

In Figure 5, the control daemon usually remains quiet and does not interfere with any of the system's activities. However, as soon as it senses any abnormality from the detection manager, it becomes active. Depending on the risk and maliciousness of the activity it then stops the malicious access to data. In case, an activity is severely malicious it immediately stops the applications access to the libraries and instantly blocks the database interface layer's door that can only be accessed by the control daemon. This feature, in turn, ensures that the data remains relatively secure.

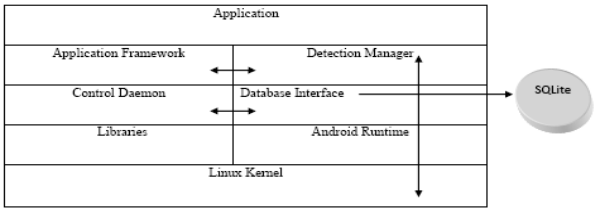

Figure 5: Modified architecture of Android

7. Conclusion and Future Work

In this work, we discussed the importance of low-end computing, namely, mobile computing. Toward this goal, we studied and analyzed five leading mobile phone operating systems. Our study has shown that Android is a promising platform because of its openness and portability. We also discussed some security issues related to Android operating system. Toward this end, we proposed an enhancement that works with the software architecture of Android.

The results in this study have not been empirically validated yet. Therefore, future work will attempt the practicality of our hypothesis by experimenting our theories on real mobile computing systems.[17]

References

[1] M. Anvaari, and S Jansen. Evaluating Architectural Openness in mobile Software Platforms. *ESCA 2010,* Copenhagen, Dennmark,2010, pp. 85-92.

[2] J. Bickford, R. O'hare, A. Baliga, V. Ganathapy, and L.Iftode. Rootkits on Smart Phones: Attacks, Implications and Opportunities. *HotMobile* Rutgers University, Annapolis Maryland, 2010, pp. 1-6.

[3] T. Gronli, J. Hansen, and G. Ghinea, Android vs. Windows Mobile vs Java ME. *PETRA 2010*, Brunel University, 2010, pp. 37-44.

[4] S. P. Hall, and E. Anderson. Operating Systems for Mobile Computing, *Consortium Computing Sciences in College: Rocky Mountain Conference*, 2009, pp. 64-71.

[5] A. Litke, K. Zotos, and G. Stephanides. Energy consumption Analysis of Design Patterns. *Applied Informatics*, Greece, 2005, pp. 86-90.

[6] E. Oliver, A Survey of Platforms for Mobile Networks Research. *Mobile Computing and Communications Review*, December 2008, pp. 56-63.

[7] C. Thompson, J. White, and D.C. Schmidt. Analyzing Mobile Application Software Power Consumption via Model-Driven Engineering. *ACM MobiSys*, 2010, pp.1-15.

[8] N. Vallina- Rodriguez, P. Hui, J. Crowcroft, and A. Rice. Exhausting Battery Statistics. *Mobiheld,* 2010, pp.9-14.

[9] *http://mobilewebsitewatch.com/page/2/*

[10] P. Clements, R. Kazman, and M. Klein. Evaluating Software Archiecture: Methods and Case Studies. Addison Wesley, 2002.

[11] Canalys Report. http://www.canalys.org

[12] Android. *http://code.google.com/android/*

[13] iPhone. http://developer.apple.com/iphone/

[14] R. Quinlan. Induction of decision trees, Machine Learning, vol. 1(1), pp. 81–106, 1986.

[15] K.A. Yuksel, O. Kiraz. Enhancing security of Linux-based Android devices, Sabanci University, Istanbul pp. 4-8.

[16] Clam Antivirus. http://www.clamav.net/

[17] Netfilter. http://www.netfilter.org

[18] ChkRootkit http://www.chkrootkit.org

[19] Snort http://www.snort.org

[20] M. Shaw and D. Garlan. Software architecture: perspectives on an emerging discipline, Pearson, 1996.

[21] H. Reza, E. Grant: Quality-Oriented Software Architecture. *ITCC-2005* (1), pp:140-145, 2005.