Handset Malwares Threats Feature Extraction Based on Behavior Analysis

Marwa M. A. Elfattah (Corresponding author)
Computer Science Dep, Faculty of Computers and Information, Helwan University, Cairo, Egypt
E-mail: marwa_8_80@yahoo.com

Aliaa A. A. Youssif
Computer Science Dep, Faculty of Computers and Information, Helwan University, Cairo, Egypt
E-mail: aliaay@helwan.edu.eg

Ebada Sarhan Ahmed
Computer Science Dep, Faculty of Computers and Information, Helwan University, Cairo, Egypt
E-mail: ebadasarhan@yahoo.com

Received: November 28, 2011 Accepted: December 12, 2011 Published: March 1, 2012
doi:10.5539/cis.v5n2p79 URL: http://dx.doi.org/10.5539/cis.v5n2p79

Abstract

The malware threat for mobile phones is expected to increase with the great functionality enhancement of mobile phones. Despite the nowadays malware high abilities, there are a lot of challenges that facing the mobile threat containment process.

From this perspective, this work introduces a novel effective solution for discovering handset malwares threats. The work proposed a new behavior based technique for mobile application analysis, which is based on exploiting the application DLL usages, in order to extract values that can be used in a malware detection process. The technique is highly expected to be able to detect zero day viruses that have the similar functionalities as existing ones. Also, since these DLL functions are easy to be extracted from the executable files, the approach is computationally efficient.

Keywords: Mobile malware, Feature extraction, Behavior feature extraction, Dynamic linked library DLL

1. Introduction

Nowadays, mobile handsets have become more intelligent and complex in functionality, much like PCs, not only that, but they also have become more popular than PCs. This resulted in an increasing number of criminals who wants to exploit the mobile capabilities for illegal gains (A. Bose & G. Shin., 2006; A. Bose, 2008; A. Bose, X. Hu Kang *et al*., 2008; H. Kim *et al*., 2008; J. Cheng *et al*., 2007). Moreover, the existence SMS/MMS and BlueTooth interfaces as new technology for data transmission has quickly become new infection vector for viruses, which makes the mobile handset susceptible to get infected even when it is disconnected from the internet.

Mobile malwares are categorized into different families or classes based on their functionalities. A completely new virus which is not a variant of any existing type starts a family of its own. All virus variants in a family share a common malicious core behavior. Historically, the first proof-of-concept of mobile malware was "Cabir", which was proposed in 2004 targeting Symbian OS. After that, mobile malware evolved rapidly during the first two years (2004-2006) of its existence. Between 2006 and 2009, the amount of malware for mobile devices trebled. This shows that the growth rate demonstrated between 2004 and 2006 has been maintained.

In spite of nowadays malware's ability of stealing and transmitting the contact list and other data, locking the device completely, giving remote access to criminals, sending SMS and MMS messages etc, some of mobile handset users treat mobile handset malware as a problem which has not happened yet, or believe that it is not an

issue which really concerns them (A. Bose & G. Shin., 2006; A. Bose, 2008; A. Bose, X. Hu Kang *et al*., 2008; G. Chuanxiong *et al*., 2004; H. Kim *et al*., 2008; J. Cheng *et al*., 2007). On other hand, the task of facing mobile handset malware is not easy, but it has a lot of obstacles. One of the most important obstacles is the poorness of the mobile resource, as, a mobile handset has limited processing power and storage capacity, unlike resource-rich PCs, a malware handling framework should not consume too much of the device resources, including CPU, memory, and battery power.

Therefore, there are many approaches for preventing mobile handset from being infected by malware. Figure 1 shows the most three major categories of those approaches. The simplest of them is to only trust and install digitally signed applications (A. Bose, 2008). This ensures that the software has undergone a standard testing procedure as part of being signed. For example, the Symbian Signed framework derives a unique application certificate from its root certificate issued by its Certificate Authority (CA) for signing an application. When a signed application is installed, the Symbian installer program verifies that the signature is valid before proceeding with the installation. This ensures that the software has not been tampered during its distribution and has undergone a standard testing procedure as part of being signed (A. Bose, 2008).

However, given the vast number of mobile applications available on the Internet, especially peer-to-peer sites, one cannot expect all applications to be signed with a certificate. An application that has been self-signed cannot be trusted to be free of malicious code. Moreover, even when an application is signed by a trusted CA, a malicious program can still enter the system via downloads (e.g., SMS/MMS messages with multimedia attachments), and it may exploit known vulnerabilities of an unsigned helper application.

Signature-based detection is another will known approach for handling mobile handset malwares (D. Venugopal & Hu. Guoning, 2008; H. Kim, J. Smith & G. Shin, 2008). It is relays on static file signature which make it vulnerable to simple obfuscation, polymorphism, and packing techniques. To identify malwares, signature-based detection systems compare the contents of a file to a dictionary of malware signatures. There are well developed signature-based techniques for malware detection on the PC domain, but it requires considerable effort to adapt these techniques for mobile handsets.

Unfortunately, as shown in Figure 1, signature-based detection techniques are not suitable for mobiles because those techniques require a new signature for every single malware variant (A. Bose, 2008; A. Bose, X. Hu Kang *et al*., 2008; H. Kim, J. Smith & G. Shin, 2008). However, mobile handsets have severe resource constraints in terms of memory and power. Also, it is challenging to distribute virus signatures files to the mobile handsets in a timely manner.

An alternative to signature-based methods is the behavioral-based detection (A. Bose, 2008; G. Chuanxiong *et al*., 2004; Hu. Guoning *et al*., 2007; L. Xie *et al*., 2010; Q. Yan *et al*., 2010; X. Zhang *et al*., 2009), which has been emerged as a promising way of preventing the intrusion of spyware, viruses and worms. In behavioral-based detection techniques, the behavior of an application is monitored and compared against a set of malicious and/or normal behavior profiles. The malicious behavior profiles can be specified as global rules that are applied to all applications, as well as fine-grained application-specific rules.

Behavioral detection is more flexible to deal with polymorphic worms or code obfuscation, because it assesses the effects of an application based on more than just specific payload signatures. Considering the fact that a new malware variant is usually created by adding new functionality to existing malware or modifying obsolete modules with fresh ones, this abstraction is effective for detecting previously-unknown malware variants that share a common behavior exhibited by previously-known malware. A typical database of behavior profiles and rules should be smaller than that needed for storing specific payload signatures of many different classes of malware. This makes behavioral detection methods particularly suitable for handsets.

One common problem with behavioral detection, however, is specification of what constitutes normal or malicious behavior that covers a wide range of applications, while keeping false positives and false-negatives low. Another one is the reconstruction method of potentially suspicious behavior from the applications, so that the observed signatures can be matched against a database of normal and malicious signatures. Heuristics on what behavioral should be monitored and how to monitor and collect behavioral vary (Figure 1).

Thus, an effective handling framework should be able to deal with diverse types of malware and malware variants, keeping both false-negatives and false-positives below a certain acceptable threshold; moreover, it should not consume the device resources, but the overhead which is required for malware handling should be kept to a minimum. In response to those requirements, this paper introduces a new approach for analyzing mobile device applications, in order to extract values that can be used for deciding which of those applications has normal behaviors on the device and which has abnormal ones, and thus, if that application should be treated as a malware or it is considered useful.

The rest of this paper is organized as follow: in section 2, the previously done researches related to that works are introduced. Section 3 details the proposed approach for analyzing, and describes the conducted experiments including its reached results. And finally, the paper is completely concluded in section 4.

2. Related Work

Analyzing malicious executables is not a new problem; consequently, a number of approaches have been already proposed. In this section, we review set of those approaches and list some of major technologies and methodologies for malware analyzing.

Some of researchers attempted adjust signature-bases detection algorithm to fit the mobile device. Deepak and Guoning (D. Venugopal & Hu. Guoning, 2008; Hu. Guoning, & V. Deepak, 2007) suggested a system that reduces the number of signatures by using a common signature for a malware and its variants. They used hash table and sub-signature matching techniques to handle the mobile resource limitation issue. Signature-based detection methods still suffer from a lot of weaknesses. Signatures are created using static information, thus being vulnerable to simple polymorphism. Also, it is challenging to distribute virus signatures files to the mobile handsets in a timely manner. Even though, only one signature is required for a malware and its variants, the amount of those signatures is still more than the amount of malware behavior signatures, which are changed rarely.

A number of behavior-based solutions already exist. From which, is the early efforts of Carsten Willems *et al*. (Carsten Willems *et al*., 2007), who developed CWSandbox (C. Willems, T. Holz & F. Freiling, 2007). They observed malware behavior by executing the malware in the sandbox. The analysis process can be done by taking an image of the complete system state before malware execution and comparing it to the complete system state after execution, or by monitoring the malwares actions during execution with the help of a specialized tool, such as a debugger. The CWSandbox is executed in a virtual environment so that the system can easily return to a clean state after completing the analysis process. But, this approach suffers from some drawbacks, such as, slower execution and device overloading.

X. Zhang, *et al*. (X. Zhang *et al*., 2008; 2009) proposed a mandatory access control (MAC) system to strictly controls the interactions between subjects (e.g., services or processes) and objects (e.g., files, sockets, etc.), which are differentiated based on the labels assigned to them. In the system an agent with a shim - a layer of code placed in between existing layers of code - can monitor all attempts to access critical files and stop them. The kernel-level mechanisms are trusted. But kernel-level solutions are too difficult to be implemented.

H. Kim *et al*. (H. Kim *et al*. 2008) and Liu L *et al*. (L. Liu *et al*., 2009) have designed a malware-detection framework based on power monitoring. Those works have shown that power anomaly is an effective indicator for suspicious activities. Identifying the causes of these activities is still a challenge for power-based malware detection as the power consumption for normal behavior is yet to be accurately quantified. Another challenge is that existing mobile handsets is not able to provide sufficient precision for power consumption measurement without involving extra measuring devices like an oscilloscope (Q. Yan *et al*., 2010).

Liang Xie *et al*. (L. Xie *et al*., 2010) assumed that malware attacks always cause anomalies in process states and state transitions. Such anomalies are reflected through malware function (API) calls, usages of system resources, and requests for system services. Also they noted that, each cell phone user has his own unique and private operational patterns. So, they adopt function call-trace techniques and human intelligence techniques in the context of cell phones to identify process misbehavior. The drawbacks of this system are that, the learning operation of the user interaction pattern is done during the mobile operation, and takes some of time, in which the mobile could be infected. On other hand, if another person has tried to use the device, the system may give false virus alarm. Also, observing function API calls means running the application – virtually or actually –, that is not effective, as mentioned before.

D. Venugopal *et al* (D. Venugopal *et al*., 2006) proposed an algorithm to monitor each DLL function that a process attempts to load. This information is then compared against lists of authorized and unauthorized functions. As will be illustrated next, dividing DLL functions between authorized and unauthorized lists is a disturbing operation, as well as, there are too many DLL functions. A lot of them could be used for malicious target, and could be also used for serviceable target.

3. The Proposed Approach for Mobile Malware Analysis

In response to the great dangerous of mobile handset malware, and the importance of finding a solution to limit this danger, this paper proposed a new approach that help in preventing mobile handsets from being infected by malware. The proposed approach is based on extracting the mobile application feature using its DLL import set, which are considered to be a good reference for the application behavior, as will be explained later.

3.1 The Power of Using Dynamic Link Library (DLL) Imports as a Behavior Feature

Most of mobile malware studies have explained that, it is better for the intelligent malware detection method to identify a malware based on its functionality or behavior. Most viruses in the mobile domain demonstrate common functionalities - e.g., deleting system files, sending MMS messages, stealing and transmitting the contact list and other data, etc. To implement these functions, a program needs to use certain DLLs.

It is well known that, a dynamic link library (DLL) is a collection of small programs, which can be called upon when needed by the executable program that is running. A DLL file is not executable file by itself, but it contains commands and/or data that are common to a certain task, and which is imported by another executable. Each executable file has an import table (M. Pietrek, 2002), which contains a list of DLLs and a list of functions from that DLL the module wishes to link to. So, the executable import table can be consider as a good reference to defining the functions needed to be accessed by the executable, therefore, it can be consider as a good reference to what the task of the executable is. Accordingly, the list of DLL functions used by a virus indicates the behavior of the virus in terms of its functionality. Therefore, the application imported DLL functions can be used as features to indicate the existing of virus.

As mentioned before, mobile systems suffer from limitations on their resource; they are not rich-resource PCs, so it is imposed on any malware analytical technique taking into account that limitations, and do not consume a lot of the mobile resource. Fortunately, the application analyzing based on extracting its DLL import sets is an easy task, and does not consume much from the processor power. Also, information kept about impacts of DLL import sets on application behavior is not too much, and does not consume much of the memory. Therefore, it could be said that, malware analytical methods which based on analyzing DLL import sets takes into account the limitation of the mobile processors.

Accordingly, this paper has introduced an approach to initially investigate each new mobile application before installing it on the device, to be sure if it has a virus or not. The paper chose to use DLL import set of each application as a good reference in the analysis process of the application behaviors.

3.2 Experiments

We have chosen to conduct our experiments on Symbian platform. Symbian is one of most commonly used mobile operating system (OS) and computing platform designed for smart phones and currently maintained by Nokia. Devices based on this operating system accounted for 29.2% of worldwide smartphone market share in 2011. It integrates the power of computing with mobile telephony, bringing advanced data services to the mass market. Moreover, it enables mobile phones to be a platform for deployment of applications and services (programs and content) developed in a wide range of languages (Java and C++) and content formats. With a flexible and modular implementation, Symbian OS provides a core set of application programming interfaces (APIs) and technologies that is shared by all of its phones.

The experiments are conducted on data sets composed of:

- 30 good application samples, which have been selected to be completely different, that is, to serve as a true representation of the most of mobile applications.
- and 56 malware application samples from 26 different malware families.

Each application sample – good or malware- is a symbian software installation script (sis file).

Figure 2 shows a flow chart that describes the proposed detection system. First of all, as shown in Figure 2, each application sis file is separately investigated, and its executables have been extracted to be analyzed. After that, the next step is that -for each application- each executable is investigated separately, and its dll imports are extracted, that is to be analyzed to define the behavior of each executable.

Then, the DLL imports obtained from an executable file are filtered, that is for extracting DLL functions that are only related to the core functionality, which could be the key for a reliable classification and ensures that it is not over-trained on the existing data. It is important to neglect imports known to occur commonly in all executables including non-virus programs, also, neglect imports that generally do not contribute to the malicious behaviors, and so on.

3.3 Results

As mentioned before, this analysis is mainly used to define the overall behaviors for each application, to well determining which behaviors are considered normal, and which are considered as indicators to malware. Therefore, only by applying the starting step – extracting executable files from the sis application - on the data, it is widely found:

1) Malware sis files could have some too small sized executable files – it could be zero sized - , which do nothing; they are only to overwrite other already existing files, or to overload the system with nothing important. Good sis files do not have so.

2) Malware sis files could have other sis file, but most of good ones do not have so. Those malwares are considered as a carrier to other malwares.

3) Malware sis files could have too much executables, which could be tens or more. It is not reasonable on good applications which do specific well defined task to have too much executables.

After that, by completing the rest of steps, the paper has proposed set of behavior vectors which could be considered as an alarm for a dangerous. Those vectors could be used in analyzing the applications behaviors and to classifying the applications as normal or abnormal. The proposed vectors are:

- B1. Accessing Bluetooth or infrared dlls – for sending critical information or viruses to others.
- B2. Accessing SMS and MMS dlls – for sending critical information or viruses to others, or to consume the device balance.
- B3. Accessing phonebook dlls – for accessing critical information, sending SMS/MMS, or making calls.
- B4. Accessing telephony dlls – for sending critical information, or to consume the device balance.
- B5. Accessing Email dlls – for sending critical information or viruses to others, or to consume the device balance.
- B6. Accessing networking dlls – for sending critical information or viruses to others, or to consume the device balance.
- B7. Accessing fileserver dlls – for accessing critical information, or corrupting the file system.
- B8. Accessing hardware dlls – for destroying it.
- B9. Accessing unknown dlls – which could be malicious.
- B10. If the executable has no UI dll – good applications usually interact with users.
- B11. If the executable has only UI dlls – do nothing but consuming the power and overload the system.
- B12. Number of too small sized or corrupted executable files included in the sis file.
- B13. Number of executable files included in the sis file.
- B14. If the sis file has other sis files.

Actually, most of previous behaviors vectors by itself may not appear malicious. However, when multiple behavior vectors are correlated together, they form an indication to an existing mobile malware. For example, it is normal to any executable file to have an access for phonebook, SMS, or BlueTooth DLLs, but, it is abnormal for an application with no user interaction to access all of them together, as in the case of CommdropperE malware, which has too much executables, some of them have no UI and try to access BlueTooth, SMS, phonebook, and fileserver DLLs, as shown in Table 1.

The behavioral analysis of malwares shows that the different members of the same malware family almost have the same behavior. Therefore, it is enough to keep track of data about only small set of the family members to recognize other new members. For example, upon examination of Skuller malware family, as shown in Table 2, it is found that most of its members:

a) carry other sis files,

b) have too much executables,

c) a big set of their executables are corrupted,

d) try to send files via BlueTooth, and

e) some of the executables do nothing accept interacting with user.

In principle, a separate DLL import set would be required for each of malware families. However, in the course of this experiment, we discovered that there are common functionalities among different families. Consequently, the amount of data required to be stored about the malicious behavior is greatly reduced, and the proposed approach is able to detect new viruses that have the similar functionalities as existing ones. In addition, since these DLL functions are easy to extract from the executable files, our method is computationally efficient.

Therefore, we could say that, the limitation of mobile handsets processing power and storage capacity is well considered.

4. Conclusion

Due to their flexible communication and computation capabilities, and their resource constraints, mobile handsets are glued victim to malwares. An effective malware handling framework should be able to deal with diverse types of malware and malware variants, keep both false-negatives and false-positives below a certain acceptable threshold, and do not consume the device resources.

Set of researcher have done a lot of work to adjust the existed PC's signature-based detection systems to be suitable for mobiles. But signature-based solutions suffer from a lot of weakness, and the resource poorness of the mobile device make signature based techniques not suitable for mobiles. So, behavioral based solutions are preferred due to their flexibility to deal with polymorphic worms, and to the small amount of data needed to be stored.

Most viruses in the mobile domain demonstrate common functionalities; a program needs to use certain DLLs to implement these functionalities. The list of DLL functions used by a virus indicates the behavior of the virus. Therefore, this paper introduce a new effective behavior based technique for mobile application analysis, which is based on exploiting the application DLL usages, in order to extract values that can be used a malware detection process.

The experiments are conducted on Symbian platform using 30 good application sample, and 56 malware application sample from 26 different malware families. A set of behavior vectors have been proposed to be considered as an alarm for a dangerous. It is found that most of behaviors vectors by itself may not appear malicious; however, when multiple behavior vectors are correlated together, they may form an indication to an existing mobile malware.

The experiments showed that, the different variant of the same malware family almost have the same behavior, so it is enough to keep track of data about only small set of them. Also, there are common functionalities among different families. Consequently, the amount of data required to be stored about the malicious behavior is greatly reduced, and the proposed approach is able to detect new viruses that have the similar functionalities as existing ones. In addition, since these DLL functions are easy to extract from the executable files. Therefore, it could be said, malware analytical methods which based on analyzing DLL import sets is effective and takes into account the limitation of the mobile resources.

References

A. Bose, & G. Shin. (2006). On mobile viruses exploiting messaging and bluetooth services. International Conference on Security and Privacy in Communication Networks, SecureComm, IEEE, PP. 1-10.

A. Bose, X. Hu Kang, G. Shin, & T. Park. (2008). Behavioral detection of malware on mobile handsets, International Conference on Mobile Systems, Applications, and Communications, MobiSys08, pp. 225-238, June.

A. Bose. (2008). Propagation, detection and containment of mobile malware. PhD thesis, the university of Michigan.

C. Willems, T. Holz, & F. Freiling. (2007). Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, 32-39.

D. Venugopal, Hu. Guoning, & N. Roman. (2006). Intelligent virus detection on mobile devices, Fourth International Conference on Privacy, Security and Trust, ACM PST, pp. 1-4.

D. Venugopal, & Hu. Guoning. (2008). Efficient signature based malware detection on mobile devices. *Mobile Information Systems journal*, 4, 33-49.

G. Chuanxiong, J. Wang, & Z. Wenwu. (2004). Smart-phone attacks and defenses, Third Workshop on Hot Topics in Networks, HotNets III, San Diego, CA.

H. Kim, J. Smith, & G. Shin. (2008). Detecting energy-greedy anomalies and mobile malware variants, The International Conference on Mobile Systems, Applications, and Communications (MobiSys), ACM/USENIX, pp. 239-25.

Hu. Guoning, & V. Deepak. (2007). A Malware signature extraction and detection method applied to mobile networks, Performance, Computing, and Communications Conference, IPCCC 2007. *IEEE Internationa*, 19-26.

J. Cheng, S. Wong, H. Yang, & Lu. Songwu. (2007). Smartsiren: virus detection and alert for smartphones, The International Conference on Mobile Systems, Applications, and Communications. *MobiSys*, 258-271, ACM.

L. Liu, Yan, G., Zhang, X., & Chen, S. (2009). VirusMeter: preventing your cellphone from spies, Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, pp. 244-264. http://dx.doi.org/10.1007/978-3-642-04342-0_13

L. Xie, X. Zhang, J. Seifert, & S. Zhu. (2010). PBMDS: A behavior-based malware detection system for cellphone Devices. 3rd ACM conference on wireless network security,WiSec10, ACM, pp. 37-48.

M. Pietrek. (2002). Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format, MSDN Magazine.

Q. Yan, R. H. Deng, Yingjiu Li, & Tieyan Li. (2010). On the potential of limitation-oriented malware detection and prevention techniques on mobile phones. *International Journal of Security and Its Applications*, 4(1), 21-30.

X. Zhang, Aciicmez, O. Latifi, A. Seifert, & S. Jose. (2008). A rusted mobile phone prototype, Consumer Communications and Networking Conference, CCNC 2008. 5th IEEE, pp. 1208-1209.

X. Zhang, L. Xie, A. Chaugule, T. Jaeger, & S. Zhu. (2009). Designing system-level defenses against cellphone malware, 28th IEEE International Symposium on Reliable Distributed Systems, pp.83-90.

Table 1. CommdropperE malware behavior vectors

	Files count	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14
CommdropperE	1							y						20	
	1						y	y		y	y				
	1	y	y	y			y	y			y				
	1									y	y				
	3							y			y				
	2							y		y	y				
	4							y			y				
	1											y			
	3									y	y				
	1										y				
	2							y							

Table 2. Skuller malware variant behavior vectors

	Files count	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14
SkullerA													66	66	
SkullerB	9											y	56	67	y
	1							y		y					
	1	y						y							
SkullerC	2	y						y					76	79	y
	1									y	y				
SkullerCB	2											y	6	8	
SkullerD	1	y						y					16	18	y
	1									y	y				
SkullerD2	1	y						y					13	16	
	1							y			y				
	1											y			
SkullerE	1	y						y					16	18	
	1											y			
SkullerF	3	y						y					26	36	y
	2							y			y				
								y	y	y	y				
							y	y				y			
				y			y	y	y						
			Y		y			y		y					
SkullerH	1	y						y					13	16	
	1							y			y				
	1											y			
SkullerM													62	62	
SkullerR	2	y						y					86	90	y
	1							y			y				
	1											y			
SkullerV	1	y						y					16	18	
	1											y			

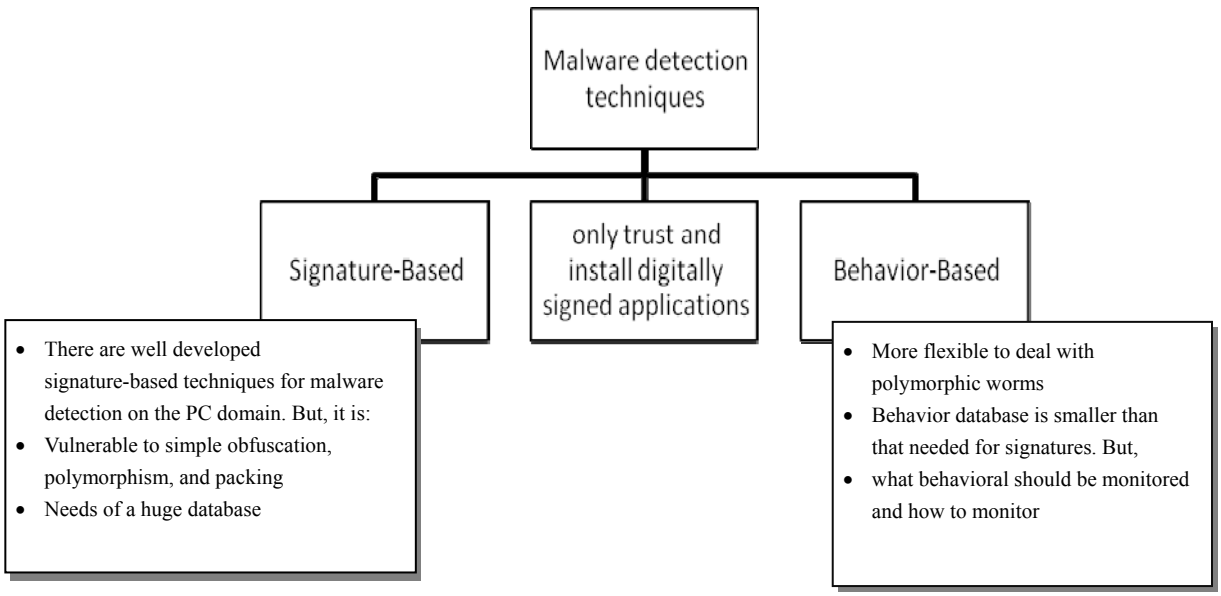

Figure 1. Mobile malware detection techniques

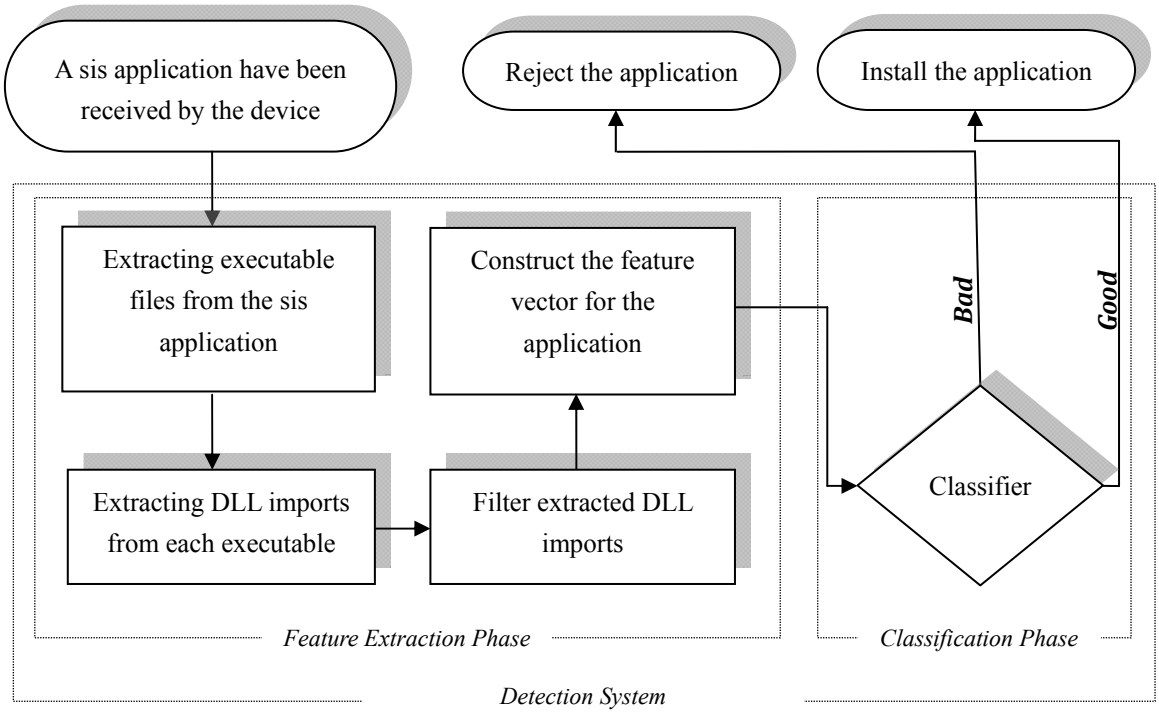

Figure 2. Mobile malware detection system