SAM: A Secure Anti-Malware Framework for the Smartphone Operating Systems

Md Shahrear Iqbal and Mohammad Zulkernine
School of Computing
Queen's University, Kingston, Ontario, Canada
{iqbal, mzulker}@cs.queensu.ca

***Abstract*—Smartphones have become an integral part of our daily life. Businesses now offer services through smartphones. Users also store sensitive personal information on their smartphones and perform financial transactions. Consequently, security attacks on smartphone platforms have also increased significantly. Traditional desktop anti-virus software are not very effective in smartphones due to the restrictive security model and they are heavily dependent on their definition updates. In this paper, we propose a Secure Anti-Malware framework (SAM) for smartphone operating systems to prevent malicious activities. The core idea of the framework resembles a smart city. The framework acts as the government of the city and treats the applications as citizens. It has components to enforce laws (prevent) and perform policing (monitor and control). It also provides APIs to aid anti-virus software and third-party applications to leverage the functionalities of the framework. Our goal is to design an operating system framework that hinders malicious activities and thus protects user resources.**

***Index Terms*—Smartphone security, Mobile malware, Malware analysis and detection, Anti-malware framework.**

I. Introduction

During the last few years, the use of smartphones has increased significantly. People are using smartphones for different types of tasks, e.g., writing emails, surfing the web, listening to music, watching videos, playing games, performing financial transactions, and creating reports. More importantly, people use their smartphones to control other devices (smart TVs, smart fridges, etc.).

Industries are continuously developing new web and application development technologies to take advantage of the feature-rich smart devices. New payment standards (e.g., Google Wallet and Apple Pay) are also introduced recently. Companies now allow BYOD (Bring Your Own Device) policy, which lets the employees bring and use their own smartphones for accessing confidential business network in office premises.

Smartphone sensors, third-party applications and the availability of the wireless Internet have increased the risk of security and privacy significantly. Different parties (e.g., government agencies and malicious hackers) have their own interests in the data stored in people's smartphones. Third-party application developers collect user information for advertisement or other purposes. Personal or business adversaries may aim at gathering sensitive information.

Smartphone operating systems implement a restrictive security model for the handheld devices. However, they often lack an anti-malware component as a core part of the operating system. The users have to depend on the anti-virus software to protect themselves. As a third-party application, anti-virus software are largely ineffective in smartphones. Moreover, even if an anti-virus software detects a malware, the security model of smartphone operating systems will restrict the anti-virus to act properly against the malware. We believe that we need an operating system which protects the user from malicious activities of the third-party applications and helps anti-virus software to complement the protection. Modern applications for smartphones should also be developed to leverage the new capabilities of the operating system to fight against malware.

In this paper, we propose a framework to make smartphone operating systems malware-aware and capable of protecting their resources. It has components to hinder malicious activities, monitor application behavior, and control activities of the third-party apps. In particular, the proposed Secure Anti-Malware framework (SAM) has the following key features:

1) It provides multiple execution zones with different levels of constraints and enforces fine-grained access control. The zoning system acts as a first line of defense against malware.
2) The framework switches to different security modes based on the context of the phone and protects user resources of one mode from another.
3) The second line of defense consists of a surveillance system to monitor device activities constantly and a number of analysis and detection components.
4) The framework is aware of other smart devices of the same owner and capable of communicating with them for effective computation offloading.
5) The framework provides APIs for leveraging its functionalities for third-party anti-virus vendors and app developers.

The remainder of the paper is organized as follows. Section II illustrates smartphone malware characteristics and existing security approaches. It also identifies some of the challenges in smartphone malware detection. Section III describes a number of related work. In Section IV, we describe our framework along with a deployment view for the Android operating system. Finally, we summarize the paper along with the future work in Section VI.

II. Background

We provide an overview of the architecture of smartphone operating systems in Figure 1. Normally, they consist of a computing platform (e.g., Linux, Darwin) and a middleware. For example, Android uses the Linux kernel and a middleware (written in Java) to provide services to the upper layer applications. Similarly, Apple's iOS uses the Darwin kernel and a middleware.

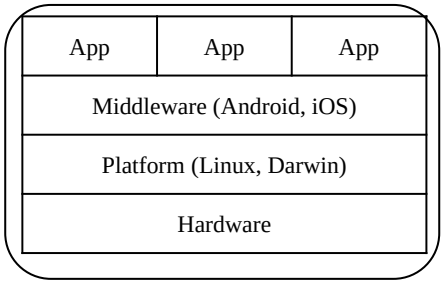

Fig. 1: Smartphone architecture

A. Malware intentions and types of attacks

The most prominent intention of malware writers is the financial gain. Attackers steal personal information, perform email spamming, send premium-rate SMSs, etc., for their own financial benefit. Sometimes, the motivation behind writing a malware may be due to political, industrial, or personal rivalry. There are also targeted malware that are created by governments, security agencies, or other organizations to spy on people. For example, Hong Kong protesters for democracy were targeted by a smartphone spy app. The app spread through SMSs with the message "Check out this Android app designed by Code4HK for the coordination of Occupy Central!" [1].

These malware eavesdrop to monitor SMS, call, or use camera or microphone to record sensitive personal information. They send files located in the SD card or the contact list to the attacker. Different types of loggers are used to steal passwords and other user inputs. Profilers can infer information about a user's behavior by monitoring his or her location, movement history, and app usage. Malware can also misuse the available services of a smartphone and send unintended SMS, call, and/or email from a user's device. Smartphones can also be a part of a botnet and help launch large-scale low noise attacks (e.g., DDoS, click-fraud). In extreme scenarios, malware can gain control of the smartphone and perform life threatening activities. For example, smartphones that control medical devices can be catastrophic if compromised by an attacker [2].

B. Propagation of malware

In the smartphone ecosystem, a common malware propagation method is that the attackers repackage a popular app or its variation and redistribute the app with malicious payloads through official and third-party markets. Billions of people residing in the developing countries do not have online payment facilities and use third-party markets to download apps which are often repackaged by the attackers. Attackers write seemingly useful applications that users download for free. Those applications dynamically load malicious payloads at a later time and infect smartphones.

In Figure 2, we present a malware characterization based on their intentions, methods of propagation and types of exploitation. Types and methods of attacks are also specified in the figure.

C. Existing security measures in smartphone platforms

The most common features in terms of smartphone platform security are app permission and app sandboxing.

App permission. All smartphone platforms have a permission-based system to restrict an app's access to the stored data and available mobile services. These permission-based systems differ in functionality from one platform to another. For example, in Android and iOS, some basic permissions like the ability to access the Internet are given to every app at install time. Users cannot revoke such permissions. All other sensitive permissions require user interaction. Moreover, users can revoke these permissions for any particular app later.

App sandboxing. Sandboxing is a security technique that isolates apps. Sandboxing prevents malicious or malfunctioning apps from damaging the device or other apps. While sandboxing is effective to some extent, it does not prevent apps completely from exploiting system or kernel vulnerabilities.

D. Challenges in smartphone malware detection

Traditional anti-malware software have been proved ineffective for smartphones. It is not easy for a third-party anti-virus software to monitor other applications and system properties in smartphones due to their inherent increased security model (e.g., sandboxing). Zhou et al. [3] showed that common smartphone antivirus software detect only between 20.2% and 79.6% of the analyzed malware and Rastogi et al. [4] claimed that they are vulnerable against transformation attacks.

Malware writers use obfuscation techniques to evade static and signature-based detection. Signature-based techniques are also useless against polymorphic and metamorphic code [5]. As a result, we are trying to make the operating system malware-aware and capable of controlling applications behavior in a finer granularity. The operating system itself will monitor app activities in realtime, perform behavioral analysis, and restrict unauthorized behavior without affecting the power consumption of the device.

III. Related Work

Researchers proposed different types of extensions to enhance the security of the smartphone operating systems. Isohara et al. [6] proposed a kernel-based behavior analysis system to monitor malicious applications. The system collects system call logs from the kernel and then matches activities with regular expressions. The system can be used to inspect some of the malicious behavior of applications. However, it cannot restrict an application from performing malicious acts. Lange et al. [7] implemented a generic operating system framework for secure smartphones called "L4Android". Their

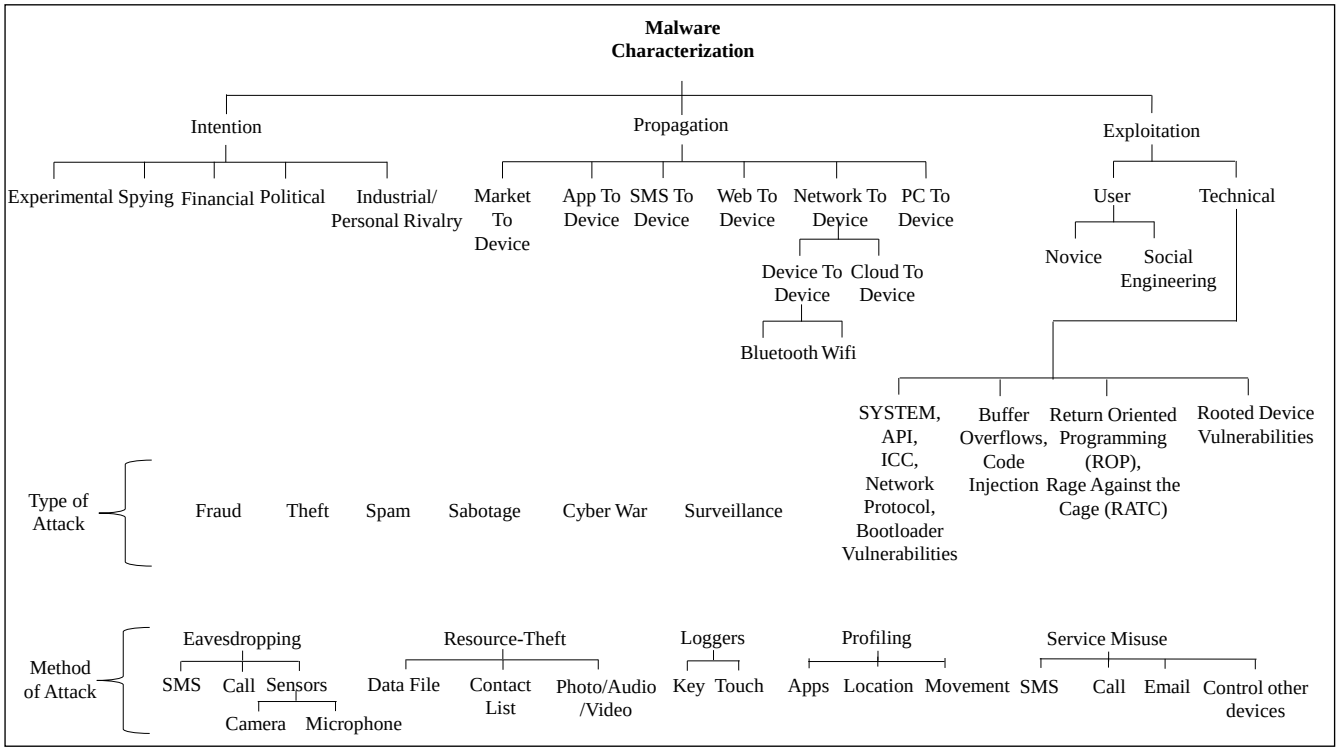

Fig. 2: Smartphone malware characterization

framework hosts multiple virtual machines to separate secure and non-secure applications. Each VM hosts its own version of Android. L4Android mainly focuses on the security of the sensitive applications. Moreover, it relies on the hardware virtualization support, which is not yet practical for smartphones.

Conti et al. [8] proposed CRePE that can enforce fine-grained policies based on the context of the phone. Similarly, Schreckling et al. [9] introduced a real-time user-defined policy enforcement framework for Android. The main drawback of these frameworks are that they require a lot of user control for their operation. A vast majority of the smartphone users are non-technical and unable to define technical policies. In our proposal, the operating system will come with predefined policies which will be sufficient for an average user. There will be options to modify the policies for power users.

Smalley et al. [10] implemented the mandatory access control (MAC) in Android. They showed that the mandatory access control is able to thwart some of the well-known malware attacks reported in the literature. Backes et al. [11] introduced Android Security Framework (ASF) that allows a third-party to easily attach code-based security modules to the Android operating system. The modules are loaded to the ASF during boot time and enforced accordingly.

In summary, all these frameworks do not make an operating system aware of the maliciousness of the apps and they add features which may or may not be directly related to malware. SAM particularly addresses the protection against malware by making the smartphone operating system an effective anti-malware.

IV. SAM Anti-malware Framework

In our framework, we treat each device as a smart city. In the smart city wheel proposed in [12], a smart city has smart people, smart environment, smart government, smart economy, smart living, and smart mobility. Similar to a smart city, the phone will have smart apps, smart execution environment, smart framework manager, smart access control, smart protection, and smart mobility. The framework manager will act as the government and other components will prevent malicious activities as well as monitor, report and control applications. Figure 3 shows a representation of SAM which is analogous to the smart city wheel. In the following subsections, we describe the details of different components of SAM. The components are: framework management, prevention, monitor and control, API for apps, and API for anti-virus.

A. Framework management

The framework manager of SAM coordinates between different components (prevention, monitor and control, etc.) of the framework. In addition, the manager does the following tasks.

Maintain the device context. The manager uses smartphone sensors and applies an algorithm to detect the physical context (e.g., home, office, unknown) of the phone. This information is used with a number of other information to maintain the context of the device.

Maintain a list of trusted devices. The manager maintains a list of trusted devices of the same owner. Users can also add a device of a different user if the device contains our framework.

Offload computation. The manager tries to entertain the request of executing offloaded computations from the trusted devices (as resources permit). We show an example in Figure 4. Since the smartphones have power constraints, malicious activity detection components can offload some of their computations to the nearby smart television (same owner) with a permanent power source.

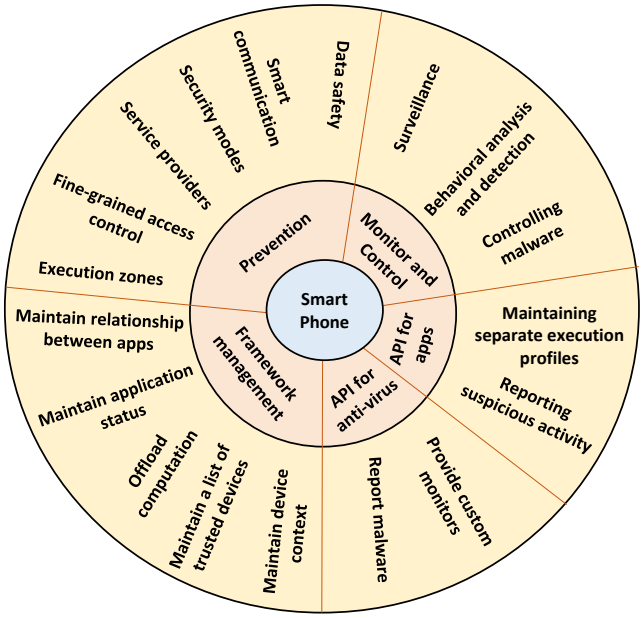

Fig. 3: Smart city analogy of SAM anti-malware framework

Maintain application status. The manager maintains the status of each application based on the behavior. When an application becomes available in the market, it will be treated as a new immigrant in the device if installed by the user and will have limited accessibility to the sensitive resources and services. After some days (e.g., 60 days) of benign activity, the application will be moved to the trusted zone with more privileges and the framework will share the information in the market. Users can also manually move an app to any zone disregarding SAM warning. By this process, SAM helps the ecosystem to maintain a white-list of benign apps.

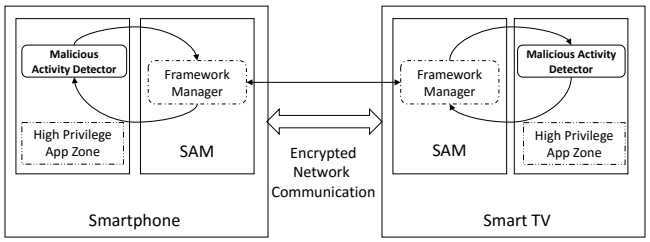

Fig. 4: A detector offloads computation from an Android smartphone to an Android smart TV

Maintain relationship between apps. The manager also maintains relationships (e.g., friends, relatives, and neighbors) between apps based on their behavior and communication. For example, applications from the same author are treated as relatives and if two applications communicate frequently without generating any malicious flags for a certain time, they will be treated as friends. Over time, these relationships may change and affect the interaction between applications.

B. Prevention

To prevent malicious activities, SAM provides the following features.

Execution zones. SAM creates different execution zones for different types of apps. Each zone provides a certain level of privileges. An application can request to move from one zone to another to achieve more privileges. However, SAM decides when such requests should be entertained. By default, SAM creates the following zones: new app zone, trusted app zone, untrusted app zone, high privilege app zone, and restricted zone (for malware).

Fine-grained access control. The zoning system for execution also provides a fine-grained access control for the apps. Power users will be able to move applications from one zone to another zone and fine-tune access control policies of different zones. For example, a user can add a policy to a zone that no applications of that zone will be allowed to use the Internet from 11 PM to 7 AM. The separation of execution zones is analogous to the separation of industrial and residential areas in a smart city. Each area may have their own security policy and a person has to adhere to the policies based on his or her location.

Service providers. There are high privilege system applications that provide services to other applications like the government service providers in a smart city. These services include performing critical low-level tasks and accessing sensitive user or device information. A third-party application can make a request for such a service and after the validation of the request, the high privilege app will perform the task. In this way, SAM removes the necessity to allow a number of critical permissions to the third-party applications.

Security modes. One of the problems of using the same phone for different types of tasks (document processing, banking, entertainment, communication, etc.) is that they are not designed initially for every type of tasks. Each task may require a different security model and a separate device to perform the task securely and effectively. To solve this problem, we introduce the concept of security mode. SAM switches to different security modes based on a decision that considers the context of the phone, the type of the task to be performed and the status of the active and background applications. The users can also select any of the available modes manually. For example, SAM will switch to the "payment mode" if NFC is being used at a POS counter. In this mode, SAM ensures the security of the NFC communication channel. All other apps and third-party services are suspended during this mode.

Secure communication. SAM provides a smart communication system to ensure the use of different languages while communicating (i.e., inter-process communication is encrypted). By default, no process can get any service from

any application. To get services, applications must request using a language that the server application understands. SAM will share the language based on the requester's status and its relationship with the server application. It can also be configured to provide an extra level of encryption (e.g., between Viber calls) while communicating with the trusted devices (the manager maintains a list) to thwart government or other agencies to intercept personal voice calls or SMSs.

Data safety. Depending on the device context, SAM uses different data profiles to protect information from one context to another. For example, resources of an application in an office context will be separated from the resources of the same application in a home or an outdoor context.

C. Monitor and control

SAM has a number of components to monitor, detect and control malware. Here, we describe the components for these tasks.

Surveillance. To detect any malicious activity, we need a surveillance system that monitors different activities and resource usage. The identification of features that can provide evidence of maliciousness is important in this regard. We divide different monitorable features into different groups, such as hardware, sensors, operating system, network, and application behavior. These monitors provide valuable information that can be used by both system apps and third-party anti-virus software.

Behavioral analysis and detection. SAM uses the surveillance data and performs behavioral analysis to detect maliciousness of an application. This detection mechanism can complement the signature-based techniques that are already built into some operating systems.

Controlling malware. SAM sends all the detected malware to the Jail (the restricted execution zone) and the user is notified. The user can decide to delete the malware or allow it to execute in the Jail. If an application executes inside a Jail, it will still be able to provide its basic functionalities. For example, suppose a malicious game performs click-fraud stealthily and SAM detects that and sends the game to the Jail. Then, the user will still be able to play the game while it is being blocked from accessing the network making it ineffective for the click-fraud attack.

D. API for third-party applications

SAM provides the following two APIs for the third-party application developers so that they can build smarter apps.

Maintaining separate execution profile for different contexts. SAM provides API that helps the developers to create apps that behave differently in different contexts. For example, a developer may want his or her application to behave differently in a home network than in an office or unknown network.

Reporting suspicious activity to the framework. Third-party apps can use a SAM API to report suspicious activity to the framework. This module detects and reports a number of malicious activities (e.g., confused deputy attack) around an application. This information will be used by the malware detection system.

E. API for anti-virus vendors

SAM also provides the following APIs for the anti-virus software vendors.

Provide custom monitors. SAM allows trusted third-party anti-virus software to provide code-based monitors to be executed for them. This API executes the monitors in an appropriate level of the operating system to provide them with their requested features.

Report malware. Using this API, anti-virus apps can access and analyze the surveillance data and detect malware. If any such application detects a malware, the application can report the malware to SAM for taking further controlling action.

V. A DEPLOYMENT VIEW OF THE FRAMEWORK

It is becoming a global trend for the operating system vendors to make the system ubiquitous. A modern OS should run on all available platforms with similar user experience. For example, Android and Windows 10 are now running in desktops, laptops, smartphones, smart watches, smart wearables, and IoT devices. Considering the current trend of the evolution of the modern operating systems, we try to design our framework as generic as possible. We believe, operating systems that are not for smartphones can also be benefited from our framework as malware infection is a global problem. However, we design our framework specifically for the smartphones and we are currently implementing it for the Android operating system. Here, we provide a deployment view of SAM for the Android operating system.

In Figure 5, we show the components responsible for different tasks of SAM. The secure communication and the surveillance component are deployed in both the kernel layer and the framework layer (denoted by K). Most of the other components are a part of the Android framework (the middle layer). Each application is assigned the "New App Zone" at install time by the modified package installer. The permission checker checks privileges associated with a zone before allowing an activity. The application status and relationship manager decides when to move an application to another zone. The zone manager moves an application to the restricted zone upon request of a detector component. Other components in this layer are: context management, trusted device management, computation offloading management, security mode management, policy management, and high privilege app service provider.

In the application layer, SAM provides APIs to create malware detector applications. They will use the surveillance data provided by the framework. Alternatively, our framework supports code-based custom monitoring module that the framework can execute for the malware detector apps. Users will use the zone and policy manager app to create/modify custom policies and assign the policies to zones.

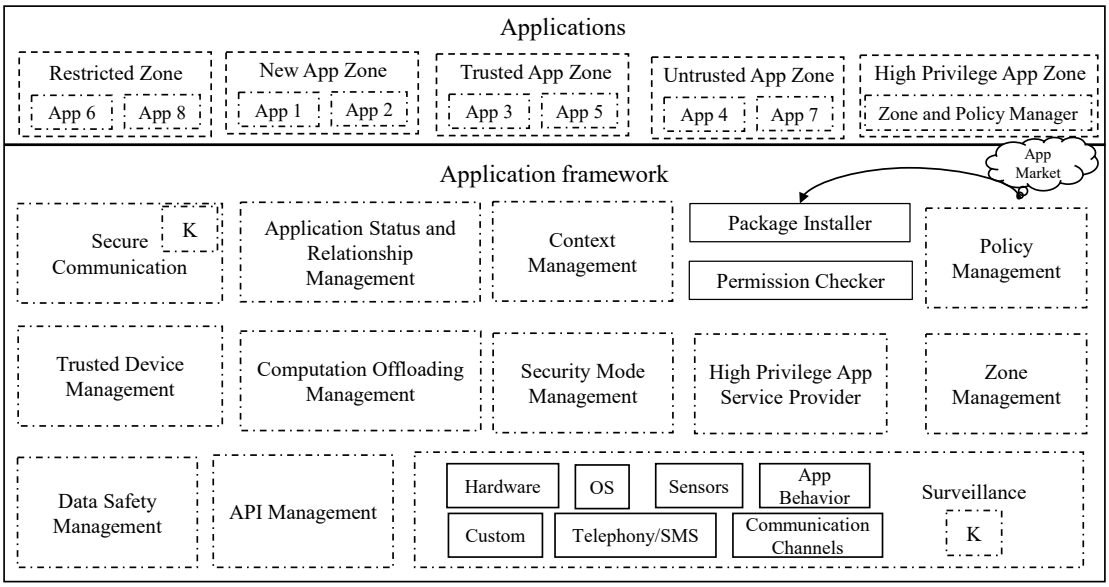

Fig. 5: A deployment view of SAM for the Android operating system

VI. Conclusion

In the desktop computer world, malware are detected and removed by the anti-virus software. However, anti-virus software are mostly ineffective in mobile platforms due to their different security model. Also, they are useless when their database is out of date.

In this paper, we propose a secure anti-malware framework (SAM) for smartphone operating systems to prevent malicious activities of the third-party apps as well as to monitor and control malware. SAM creates a virtual city where applications are the citizens and the framework manager is the government. Inside the device, SAM creates multiple execution zones with different privileges and enforces fine-grained access control through user-configurable policies. Application behavior is analyzed continuously and controlling actions are taken when necessary. SAM makes smartphones aware of other neighboring devices and provides the capability of offloading computation for expensive malware detection. Each application can behave differently in different contexts (e.g., office and home) through the provided APIs and the framework ensures the separation of resources between contexts. The framework also provides multiple security modes and encrypted communication between application components. There is a specific API for the anti-virus vendors to aid them in the detection and controlling of the malware in smartphones. Finally, we describe how our framework can be deployed in the Android operating system. Our goal is to make the framework as general as possible so that the framework can be deployed in any operating system.

Acknowledgment

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Research Chairs (CRC) program.

References

[1] "Hong kong protesters targeted by smartphone spy apps, security company saysa," http://www.theguardian.com/technology/2014/oct/01/hong-kong-protesters-targeted-by-smartphone-spy-apps-security-company-says, accessed: 2015-07-08.

[2] "Look outhes got a phone!" http://www.vanityfair.com/news/2012/12/microcomputers-weapons-smartphone?wcmmode=disabled, accessed: 2014-09-23.

[3] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*. IEEE, 2012, pp. 95–109.

[4] V. Rastogi, Y. Chen, and X. Jiang, "Catch me if you can: Evaluating android anti-malware against transformation attacks," *in IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 99–108, 2014.

[5] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *International Conference on Broadband, Wireless Computing, Communication and Applications*. IEEE, 2010, pp. 297–300.

[6] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for android malware detection," in *Proceedings of the Seventh International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2011, pp. 1011–1015.

[7] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4android: a generic operating system framework for secure smartphones," in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM, 2011, pp. 39–50.

[8] M. Conti, V. T. N. Nguyen, and B. Crispo, "Crepe: Context-related policy enforcement for android," in *Information Security*. Springer, 2011, pp. 331–345.

[9] D. Schreckling, J. Köstler, and M. Schaff, "Kynoid: real-time enforcement of fine-grained, user-defined, and data-centric security policies for android," *in Information Security Technical Report*, vol. 17, no. 3, pp. 71–80, 2013.

[10] S. Smalley and R. Craig, "Security enhanced (se) android: Bringing flexible mac to android," in *Proceedings of the 20th Annual Network and Distributed System Security (NDSS) Symposium*, vol. 310, 2013, pp. 20–38.

[11] M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky, "Android security framework: Extensible multi-layered access control on android," in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 2014, pp. 46–55.

[12] B. Cohen, "What exactly is a smart city?" http://www.fastcoexist.com/1680538/what-exactly-is-a-smart-city, accessed: 2015-03-06.