Malware Detection on Mobile Devices

Asaf Shabtai
Deutsche Telekom Laboratories at Ben-Gurion University and
Department of Information Systems Engineering, Ben-Gurion University, Israel
shabtaia@bgu.ac.il

Abstract— **We present various approaches for mitigating malware on mobile devices which we have implemented and evaluated on Google Android. Our work is divided into the following three segments: a host-based intrusion detection framework; an implementation of SELinux in Android; and static analysis of Android application files.**

Keywords- Mobile Devices, Machine Learning, Malware, Security, Android, SELinux

I. INTRODUCTION

Smartphones have evolved from simple mobile phones into sophisticated yet compact minicomputers. Mobile devices become susceptible to various new threats such as viruses, Trojan horses and worms, all of which are well-known from the desktop computers. Our research has explored several perspectives of the challenge posed by malware to mobile devices. We mainly focused on Android-powered mobile devices. The Android framework, Google's new software stack for mobile devices, includes an operating system, middleware and key applications. In [1] we provide a comprehensive security assessment of the Android framework and security mechanisms incorporated into it. We conducted a methodological qualitative risk analysis that identifies high-risk threats to the framework and any potential danger to information or to the system resulting from vulnerabilities that have been uncovered and exploited. Our security assessment of the Android framework indicates that malware penetration to the device is likely to happen, not only at the Linux level but in the application level (Java) while exploiting the Android permission mechanism and thus exploring methods for protecting the Android framework is essential. A collection of applicable security solutions for mobile devices in general and Android in particular is presented in [2]. First, it is highly important to incorporate a mechanism that can prevent or contain potential damage deriving from an attack on the Linux kernel layer. Also, better protection should be added for hardening the Android permission mechanism or for detecting misuse of granted permissions. Consequently, we assign the highest priority to SELinux access control mechanism along with an Intrusion Detection and Context Aware Access Control capabilities. We place at a lower priority data encryption, SPAM-filtering and selective Android permission mechanisms. Remote management, VPN and authentication are recommended as solutions for corporate customers.

II. INTRUSION DETECTION FRAMEWORK

In [3-6] an innovative host-based intrusion detection system for detecting malware on mobile devices was developed and evaluated. The framework (figure 1) relies on a lightweight agent (in terms of CPU, memory and battery consumption) that continuously samples various features on a device, analyzes collected data using machine learning and temporal reasoning method and inferences the state of the device. Features belonging to groups such as Messaging, Phone Calls and Applications belong to the Application Framework category and were extracted through APIs provided by the framework; features belonging to groups such as Keyboard, Touch Screen, Scheduling and Memory belong to the Linux Kernel category.

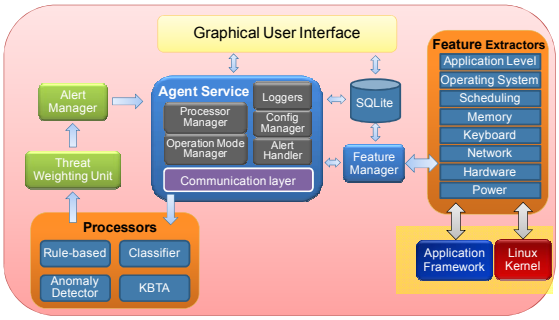

Figure 1. The Host-based Intrusion Detection framework.

A. Malware Detection using Anomaly Detection

In [3] we the intrusion detection framework continuously samples various system metrics and applies machine learning (ML) methods to infer the state of the device, i.e., whether the collected data is normal (benign) or abnormal (malicious). The main assumption is that system metrics (e.g., CPU usage, number of sent packets through the Wi-Fi, number of running processes, battery level) can be employed for detecting previously un-encountered malware by examining similarities to system metric patterns induced by known malware. Since no malicious applications are yet available for Android, we developed four malicious applications, and evaluated the ability to detect new malware based on samples of known malware. We evaluated several combinations of classification and anomaly detection algorithms (Decision Tree, Naïve Bayes, Bayesian Networks, k-Means, Histogram and Logistic Regression); feature selection methods (Chi-Square, Fisher Score and Information Gain); and a number of top features selected (10, 20 and 50). The purpose of this study is to understand how a detection algorithm, a particular feature selection method, and number of top features combine to differentiate between benign and malicious applications which are not included in the training set, when training and testing are performed on different devices and to find specific features that yield maximum detection accuracy. Empirical results suggest that the proposed framework is effective in detecting malware on mobile devices in general and on Android in particular (accuracy 87.4% with false positive rate 0.126).

978-0-7695-4048-1/10 $26.00 © 2010 IEEE
DOI 10.1109/MDM.2010.28

B. Using the KBTA Method

In [4, 5] we examine the applicability of detecting malware instances using a light version of the Knowledge-based Temporal Abstraction (KBTA) method [6] that can be activated on resource-limited devices. Using KBTA, continuously measured data (e.g., number of running processes) and events (e.g., software installation) are integrated with a temporal-abstraction knowledge-base; i.e., a security ontology for abstracting higher-level, meaningful concepts and patterns from raw, time-oriented security data, also known as temporal abstractions (Figure 2).

The new approach was applied for detecting malware on Google Android powered-devices. Evaluation results demonstrated the effectiveness of the new approach in detecting malicious applications on mobile devices (detection rate above 94% in most scenarios) and the feasibility of running such a system on mobile devices (CPU consumption was 3% on average).

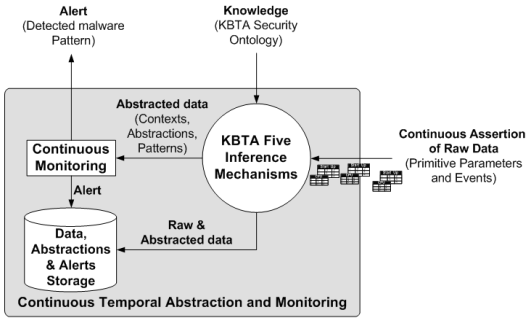

Figure 2. The KBTA process

III. SECURING ANDROID USING SELINUX

In [7] we propose the implementation of SELinux in Android in order to harden the Android system and to enforce low-level access control on critical Linux processes that run under privileged users. By choosing this route, we can protect better the system from scenarios in which an attacker exploits a vulnerability in one of the high privileged processes. We ported SELinux to Android and evaluated it on an HTC G1 device. Our experiment indicates that running SELinux on Android-powered devices is feasible and that an enhanced security level can be achieved by providing a proper policy. This enhanced security has the desirable property of not disturbing a regular user in any noticeable way. In fact, the user need not even be aware that SELinux has been applied.

While SELinux is known for its cumbersome policy maintenance, we created a "targeted" policy which focuses on confining known critical processes (i.e., system processes and daemons with high privileges); this results in a small-sized policy file. A targeted policy leaves unknown applications subject to other existing access control mechanisms such as the Android's application permission mechanism and the POSIX users mechanism.

IV. STATIC ANALYSIS OF ANDRIOD APPLICATION FILES

Machine Learning classification algorithms were employed to more effectively detect unknown malware on PC platform. With machine learning methods, the application file is represented by static features extracted from the file (e.g., byte sequences OpCodes or features extracted from the executable PE header). Classifiers are then applied to learn patterns in the code in order to classify new (unknown) files. Static analysis consumes much less resources and time, compared to dynamic analysis in which features are extracted from the system while the application is running.

Due to a lack of Android malware, our evaluation [8] focuses on classifying two types of applications: tools and games. We extracted approximately 10,000 features from Android application files including apk features, XML features and dex features. dex features were extracted using a a "dex" file parser which we developed in order to transform contents of the dex file into standard features. We evaluate the ability to detect games and tools applications that *did not appear* during the training phase. We evaluated nine classifiers (Bayes Net, Naïve Bayes, Decision Tree, PART, Boosted Naïve Bayes, Boosted J48, Random Forest and VFI), three feature selection methods (Info Gain, Fisher Score and Chi-Square) and selecting the top 50, 100, 200, 300, 500 and 800 feature. The results show that the combination of Boosted Naïve Bayes and the top 800 features selected using InfoGain yield an accuracy level of 91.8% with a 0.172 FPR. We conclude that features extracted statically from apk files, coupled with ML classification, can provide a good indication about the nature of an Android application without running the application, and can assist in detecting malicious applications.

V. CONCLUSIONS

To provide well-rounded protection, a security suite for mobile devices or smartphones (especially open-source ones such as Android) should include a collection of tools blending various capabilities that operate in synergic fashion. This article provided an overview of some of the most relevant approaches anchored in the area of Machine Learning, anomaly detection, KBTA, as well as access control using SELinux.

REFERENCES

[1] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms", CoRR abs/0912.5101, 2009.

[2] A. Shabtai, et al. "Google Android: A Comprehensive Security Assessment," IEEE Security and Privacy Magazine, http://doi.ieeecomputersociety.org/10.1109/MSP.2010.2.

[3] A. Shabtai, Y. Wiess, U. Kanonov, Y. Elovici, C. Glezer, "Andromaly: An Anomaly Detection Framework for Android Devices," submitted pending revision to Computer & Security.

[4] A. Shabtai, U. Kanonov, and Y. Elovici, "Detection, Alert and Response to Malicious Behavior in Mobile Devices: Knowledge-Based Approach," RAID, 2009.

[5] A. Shabtai, U. Kanonov, Y. Elovici, "Intrusion Detection on Mobile Devices Using the Knowledge Based Temporal-Abstraction Method", submitted pending revision to Journal of systems and Software.

[6] A. Shabtai, Y. Fledel, Y. Elovici, and Y. Shahar, "Using the KBTA Method for Inferring Computer and Network Security Alerts from Time-stamped, Raw System Metrics," Journal in Computer Virology, 8(3): 267-298, 2009.

[7] A. Shabtai, Y. Fledel, Y. Elovici, "Securing Android-Powered Mobile Devices Using SELinux", IEEE Security and Privacy, http://doi.ieeecomputersociety.org/10.1109/MSP.2009.144

[8] A. Shabtai, Y. Fledel, Y. Elovici "Detecting Malicious Applications on Android by Applying Machine Learning Classifiers to Static Features," (Poster) ACSAC, 2009.