Runtime Security Verification for Mobile Devices

Mobile devices, such as smartphones and tablets, are gaining popularity and increasingly being used as the main devices for communication, to store sensitive personal information, to access the web and to conduct e-commerce activities. Android devices are increasingly being targeted by malwares. This shift of attacks to mobile devices bring in new challenges in operating system security that are not present in more traditional systems such as Windows and Linux, and require developments of new techniques to deal with the challenges. This project aims to develop runtime verification techniques, i.e., detection of attacks and anomalous behaviours at runtime, and to implement security monitors for mobile devices. More specifically, we aim to develop architecture independent and flexible security policy specification languages, and automated generation of security monitor codes to enforce policies specified in those languages. We have conducted a preliminary case study on the Android operating system, and shown that it is effective in mitigating some previously unknown attacks on Android systems.

As a major case study, we have designed and implemented a security extension of the Android OS with a custom security monitor. The Android OS is built on top of Linux kernel, so at the most basic level, it inherits most of the security architecture of Unix/Linux. Applications in Android, however, do not run directly on Linux. Rather they run inside a virtual machine, called Dalvik Virtual Machine (DVM), that are insulated from the rest of the system. This sandboxing and the Linux security features Android inherits address the traditional operating system security issues. However, most of the interesting issues arise on the level of applications, and a separate security mechanism is required to deal with those. The Android application framework provides several API to access certain functionalities of the device, such as making phone calls, sending SMS, querying GPS location information, or unique device ID (such as the IMEI number), etc. Access to these functionalities is controlled via the permission mechanism of Android. Despite the various security mechanisms implemented in Android, it is still vulnerable to various attacks. We shall focus mostly on attacks that target the permission mechanism at the application level. A main weakness in the Android permission mechanism lies in the so-called permission leakage problem. In Android, an app can provide a “service” to another app. Through this provision of services, an app can “leak” certain capabilities to an unprivileged app. For example, an app which has access to the ability to make phone call could act as a proxy for another app. This leads to the problem of privilege escalation, i.e., an app obtains a permission it was not granted by exploiting other apps.

We have extended Android security mechanisms by adding a custom security reference monitor in the Android OS, both in the underlying Linux kernel and in the Android application framework. This modified Android OS is called LogicDroid, and its architecture is given in Figure 1. The reference monitor in LogicDroid guards access to certain objects in the (operating) system that a system owner would like to protect from unauthorized uses. In the context of Android, these resources could be access to private data such as contact database, location information, or functionalities such as the ability to make phone calls, SMS or opening a connection to the internet. A runtime verification framework should allow one to specifiy unambigiously the security policy to be implemented, and an enforcement mechanism for that policy. In this work, we follow an established approach to runtime verification using linear temporal logic (LTL), which is commonly used in runtime verification. Linear temporal logic extends the (classical) propositional logic with temporal operators that allow one to specify temporal relations between events in a system, something which is crucial in detecting malware behaviors in Android.

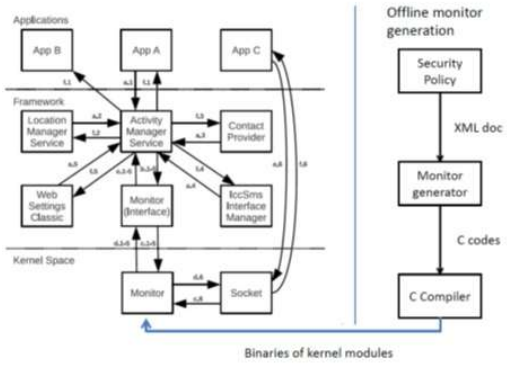

Figure 1. LogicDroid architecture.

Principal Investigator	Asst. Prof. Alwen Fernanto Tiu *(SCSE)*
Researcher	Mr. Nguyen Thanh Nam
Student	Ms. Du Xiaoning Mr. Hendra Gunadi
School / Department	School of Computer Science and Engineering