Improving Android Security through Real-time Policy Enforcement

Luke Chircop [⋆1], Christian Colombo[2] and Gordon J. Pace[3]

University of Malta

The use of the Android operating system has become a very popular option with a vast variety of mobile devices. This popularity means that companies and other users are more likely to consider using android devices. Naturally there will be users and companies concerned with how their Android devices are used. Therefore, some sort of device management is required.

Let us consider a coffee distribution company that has employees visiting its customers to showcase new products and take orders. Such a company would need to provide its employees with portable devices containing sensitive data about their products and customers. Therefore, the company would want to limit access to such data to only authorized applications or users. It could also want to disable the android market or not allow untrusted applications from running on the mobile device. Another possible scenario could be that of having parents concerned with how their children use their mobile devices. It is well known that children love to play games, therefore a parent might want to control the amount of hours per day that they could spend playing on their mobile device. They could also want to make sure that the browsers that their children use, filter out bad websites. Parents could also want to control how many messages and phone calls their children make.

Unfortunately such checks or control over mobile devices cannot be achieved with the stock versions of Android. Instead techniques such as runtime verification can be used. This technique consists of introducing a runtime monitor that observes the behaviour of an executing process to be able to determine if it violates any predefined properties.

One common approach that has been implemented by various tools such as Weave Droid [1], RV-Droid [2] and Aurasium [3] focus on application-based monitoring. This approach consists of instrumenting applications that are usually downloaded from the android market with monitoring code before they are installed on the device. The instrumented monitoring code will then be able to observe the applications' behaviour while executing and report any property violations that it encounters. Therefore with this approach, parents could write properties to raise warnings for situations such as when a browser allows its user to request a URL which could potentially be harmful to the user or device. Properties could also be introduced to limit the file/folder access for applications running on the device to safeguard against any attempts to leak sensitive

⋆ The research work disclosed in this publication is partially funded by the *Master it!* Scholarship Scheme (Malta)

data (useful for company mobile devices).

Although it is able to provide a good monitoring framework, it has some limitations. One of which is the fact that a user could easily remove an instrumented application from the device and install an un-instrumented version to be able to bypass all the checks. For companies that provide its employees with devices containing sensitive data this is highly unwanted since the un-instrumented applications could then gain access to sensitive data. Further more, there are some properties that cannot be checked by this approach. For example, let us consider a parent that does not want her children to send more then 100 messages per hour. Such a property cannot always be correctly checked since application-centric monitors observe the behaviour of applications independently of each other. Therefore, in cases where more than one application capable of sending messages is installed on the device, the user would be able to send more than 100 messages per hour without having a property violation being reported.

In our proposal, we are exploring a different approach to runtime verification. Instead of monitoring the applications separately, we are going to be introducing a device-centric approach. This will allow us to observe events at a system-wide level hence allowing us to monitor user behaviour, communication between applications, application installations, etc. To achieve this goal, a tool is going to be developed that will generate a loadable kernel module and event handling code which will be instrumented inside the dalvik virtual machine as shown in figure 1. The instrumented operating system would then need to be installed on the device that requires runtime monitoring.

The loadable module will be loaded by the Android kernel and act as an oracle to determine if a property has been violated or not. Therefore, to be able to check for a property violation such as not allowing a user to send more than 100 messages per hour, it would be required to instrument the send message function inside the dalvik virtual machine. Here for each event fired, the instrumented code would inform the loadable kernel module which would then determine if that action violates the property.

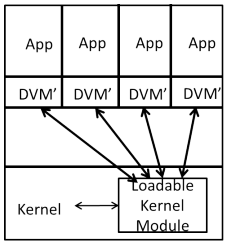

Fig. 1. Instrumented Android OS

To be able to evaluate the proposed solution, we are first going to identify a number of scenarios that can be expressed as device-centric properties. These properties will then be passed on to our developed tool which will automatically generate and instrument a stock Android operating system. The operating system will then be installed on a corresponding device. Once the instrumented operating system is installed on the device, a number of actions will be carried out to be able to determine if the monitoring introduced does observe and detect any property violations whilst allowing good behaviour. Using this approach, we hope to provide the mobile device users with an extra layer of device management helping them have more control of their own device.

References

1. Y. Falcone and S. Currea. Weave droid: aspect-oriented programming on android devices: fully embedded or in the cloud. In M. Goedicke, T. Menzies, and M. Saeki, editors, *ASE*, pages 350–353. ACM, 2012.
2. Y. Falcone, S. Currea, and M. Jaber. Runtime verification and enforcement for android applications with rv-droid. In S. Qadeer and S. Tasiran, editors, *Runtime Verification*, volume 7687 of *Lecture Notes in Computer Science*, pages 88–95. Springer Berlin Heidelberg, 2013.
3. R. Xu, H. Saïdi, and R. Anderson. Aurasium: Practical policy enforcement for android applications. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 27–27, Berkeley, CA, USA, 2012. USENIX Association.