Formalization of the Access Control on ARM-Android Platform with the B Method

Lu Ren, Wei Wang, Xiaodong Zhu, Yujia Man, Qing Yin
State Key Laboratory of Mathematic Engineering and Advanced ComputingZhengzhou, China, 450001
miranda.lu.rn@gmail.com

Abstract. ARM-Android is a widespread mobile platform with multi-layer access control mechanisms, security-critical in the system. Many access control vulnerabilities still exist due to the course-grained policy and numerous engineering defects, which have been widely studied. However, few researches focus on the mechanism formalization, including the Android permission framework, kernel process management and hardware isolation. This paper first develops a comprehensive formal access control model on the ARM-Android platform using the B method, from the Android middleware to hardware layer. All the model specifications are type checked and proved to be well-defined, with 75% of proof obligations demonstrated automatically. The results show that the proposed B model is feasible to specify and verify access control schemes in the ARM-Android system, and capable of implementing a practical control module.

1. Introduction
Recently, mobile devices have changed our lifestyle, like the BYOD (Bring Your Own Device), mobile payment, etc. The security of embedded system thus becomes increasingly important, where the access control policy plays a significant role. ARM-Android is a popular mobile system architecture equipped with multilevel access control mechanisms, such as the Android permission framework, kernel sandbox and hardware isolation. Although Android has some controversial features because of the application-level permission and incomplete access control policy, the current trusted execution technology, like ARM TrustZone, provides hardware isolation environment for the system. With the combination of hardware and software protection prevalent in both industry and academia, most studies aim at the practical technique and system design. Xu et al. [1] implemented a TrustZone-based prototype system, KNOX [2] is a real-world product of Samsung, utilizing the TrustZone as the root of trust. However, there is no common access control model on the ARM-Android platform, supporting the practical technologies theoretically. As a formal method, B method has great advantages in system modeling and development, ensuring the high security of software systems.

This paper seeks to formalize the access control mechanism on ARM-Android platform, using the B language to describe the policies in different layers. A comprehensive formal model is constructed with the ability to visualize and auto-prove security properties progressively, which supports the model enhancement and consistency valuation as well. Based on Atelier B, the results illustrate that all the formal specifications are well-defined and 75% (535 out of 718) proof obligations are automatically proved, the rest to be proved interactively.
Our main contributions are:
  i). We develop a formal Android permission model with the B method, specifying the inter-
     component communication and access control operation to maintain primary security properties.
  ii). We design a process control model in the kernel and hardware layer, describing the process
     access management with the hardware-based isolation mechanism.

2. Background

2.1. Android overview
Android is an open source software stack including the Linux kernel, the middleware and the
application layer[3].

Android Application. An Android application consists of various components, which should be
instantiated by the system at runtime. Android isolates one application execution from others and
provides the ICC (Inter Component Communication) mechanism. The AndroidManifest.xml file of an
application gives some basic information, e.g., package attributes, an identifier, contained components
and app-related permissions.

Android permission mechanism. Diverse permissions are applied for the resource protection,
comprising the identifier, permission group and protection level. The permission-based security
mechanism is enforced at both the application installation time and running time. When installed, an
application requests a set of permissions to be granted by the user. At runtime, the system performs
permission check on instantiated components and control ICCs. Android 6.0 (or higher) supports
dynamic permission revocation at runtime [4].

Android kernel access control. The Linux kernel of Android provides many core services such as
the process scheduling, sandbox isolation and inter-process communication, involving the UID-based
and GID-based access control mechanisms. However, these kind of discretionary strategies will result
in privilege escalation and permissions leakage. Since version 4.3, Android integrated the SELinux
(Security-Enhanced Linux) and achieved mandatory access control based on the domain-type
mechanism. Every process and file should be associated with a security context, and the defined
policies in the system are fine-grained. When a process accesses target resource, all control conditions
must be satisfied.

2.2. TrustZone isolation
ARM Trustzone is a hardware-based security technology, with the secure and normal world strictly
separated. The software within the processor resides in only one of the isolated zone, and the secure
resource cannot be accessed by normal world components. The TrustZone architecture sets a secure
monitor mode and brings a rigorous mechanism into the secure world in order to control the context
switch between two worlds, triggered by the SMC (Secure Monitor Call) instruction or other
exceptions [5]. In general, ARM-Android takes advantage of the TrustZone isolation environment to
robustly protect sensitive assets in the secure world, thus preventing possible attacks which are from
the normal processor core.

2.3. B method
B method is a formal method based on the set theory and first order logic, and B development covers
the entire system development cycle. There are two relevant activities: writing formal specifications
and proving them. The first stage is to build abstract machines, which will be refined and finally
transformed into fault-free concrete models. During modeling process, the second activity performs
numerous type checks and theorem demonstrations to prove the correctness of specifications. After
proved, the model can be coded in to C or Ada language. There are a number of mature tools
supporting the B method, such as Atelier B and ProB [6].

3. Android permission model
3.1. Model structure

On the basis of realistic Android application framework, the model structure is shown in Fig.1. The permission and component machine are in the base layer, with the application system model above. Taking the ICC security into the consideration, we extract the necessary elements of entities and take them as the variables in different abstract machines.

With all static items elicited through the AndroidManifest.xml file, the status attribute of a component is defined to describe the communicating state dynamically, containing six constants: unavailable(un), free(fr), requiring(rq), waiting(wt), verifying(vf), calling(call), called(called). Therefore, the application invocation is regarded as the communication process with state migration under a control network. Assuming that all components in the system can only have one running instance, we use the 2-tuple (caller, callee) to indicate the status of communicating pairs, and then express the status transition process.

![Figure 1. Structure of Android Permission formal model](image)

3.2. Policy extraction

More than ten primary authorization policies in the Android middleware are obtained according to the new Android official document, and then reduced into six logical expressions [4]. For example, record the delegated permission set of an application A as $ps_A$, and A requests the permission $P$ which belongs to the group $gp$. $A$ is granted $P$ if and only if

i) The user delegates the permissions in $gp$ to $A$ explicitly through the notification interface. or

ii) Some of permissions in $gp$ are already contained in $ps_A$.

3.3. Formal permission framework

We establish the basic abstract machine Permission, Component and Perm_Sys based on the model structure in Fig.1, wherein the B method-based framework Perm_Sys includes Component and Permission and uses the variables and operations in them.

The overall operations in Perm_Sys, consist not only of manipulations on an application such as the installation, startup and permission assignment, but of the component communication simulation. There are ten crucial access-related operations defined in the formal model: four kinds of requests (Out_Req_1, In_Req_1, Out_Req_2, In_Req_2), a successful access (Succ_Access), the finish action (EndAccess), the dynamic permission revocation (Revoke_ac), the modification and restoration of component’s status (Modify_ac, Restore_st, Clr_voidcall). These operations also concern the state transitions of a component. The state is migrated only if the invocation context complies with access control policy. Once the permission check fails, the component remains in its original state. Fig.2 presents the state transition process.
4. PROCESS MODEL BASED ON ARM TRUSTZONE

4.1. Model elements
We abstract out three fundamental entities in the kernel and hardware layer to characterize access actions of a process. As highlighted in Fig.3, the process object consists of the identifier, security domain, execution and access status. The identifier represents the unique PID, which can be used in the traditional discretionary access control mechanism. The resource object is composed of one identifier, security domain, available state and buffer capacity. Specifically, the security domains introduced in the process and resource entity indicate their location in the hardware layer, either the secure world or normal world. Both the process and resource model need observe the rules defined in AC_Policy. The item is as fine-grained as the mandatory access control policy, only allowing the specific process performs whitelist operations on the certain resource.

The entire system machine is built on these basic models for accessing and communicating simulation of the process, including the process scheduling and hardware world switching, etc. There are four constants indicating realistic process execution states: ready, blocked, run and final. Each process has four access statuses: pre-access (pre), wait for accessing (wait), accessing (acing) and post-access (post).

Figure 2. State transition of a component in Perm_Sys
4.2. Context switch in TrustZone
The access control on the TrustZone architecture largely relies on a secure context switch between two partitions under the monitor mode. Therefore, the access actions involving two different worlds and transition operations in the hardware layer need to be characterized. Concretely, the hardware domain attribute referring to current processor operating environment is set up, in order to ensure that it always match the security domain of the process and resource entity. Moreover, the process interrupt should be specified in the system.

According to the TrustZone whitepaper [7], we define eighteen operations for the context switch formalization. Take the hardware transition from the normal world to secure world for an instance. When the processor is executing in normal world, the dedicated instruction and exceptions cause it to switch into the monitor mode. The programs within monitor mode save the current world state and restore the status of secure world, that is, isolate the normal world and enable the hardware attribute as “secure” in the formal model, making all non-secure programs cannot access secure assets.

4.3. Formal process model with isolation
The formal process model details the resource access of a certain process in all possible situations, including the cross-world request, the secure context switch and so on. Because the process execution state is relevant to the access status, it is necessary to demonstrate the migration of both two states in the same access-related action. A process may have the following six central operations: the request (Pre_Access), a proper access (Pro_Access), waiting for occupied assets (Wait_Access), re-accessing the allowed resource (Re_Access), a failed access (Fail_Access) and the finish (End_Access). Fig.4 shows the primary state migration under operation conditions.

Generally, the access operations cover two-level control strategies: the security domain based and policy-based mechanism, corresponding to TrustZone hardware isolation and mandatory access control in the kernel respectively. When the context switch occurs in the hardware, the access process in switched world will be rebuilt.
5. MODEL EVALUATION

Our model development is based on the Atelier B4.3, all security properties translated into either invariants or operation conditions. Atelier checks the type of all model components and proves the specifications are well-defined. Table 1 and Table 2 summarize the size of formal Android permission framework and process model, with the number of POs (Proof Obligations) and the rate of automatic proofs.

**Table 1. The number of POs in the permission model**

| Machine   | Code Size | P0s  | Proved | Unproved | Auto-Proof Rate% |
|-----------|-----------|------|--------|----------|------------------|
| Permission| 72        | 12   | 10     | 2        | 83%              |
| Component | 84        | 19   | 16     | 3        | 84%              |
| Perm_Sys  | 245       | 147  | 91     | 56       | 62%              |

**Table 2. The number of POs in the process model**

| Machine | Code Size | P0s  | Proved | Unproved | Auto-Proof Rate% |
|---------|-----------|------|--------|----------|------------------|
| Proc    | 86        | 21   | 17     | 4        | 81%              |
| Res     | 66        | 46   | 39     | 7        | 85%              |
| Pro_Sys | 320       | 473  | 362    | 111      | 77%              |

Typically, the high percentage of auto-proofs contributes to reduce the model complexity. The results show that our machines are well-formed with the correct grammar, and 75% of lemmas (535 out of 718) are demonstrated automatically, which is acceptable and rewarding. We are now working on proving remaining POs manually.
6. RELATED WORK
Formalization of Android permission framework was pioneered by Shin et al. [8] They utilized Coq to formalize the Android permission framework, and analyzed its safety and completeness. Similarly, Betarte et al. [9] enriched the model and formally proved some crucial properties. In the study of Fragkaki et al., [10] an abstract formal permission model was proposed with desired attributes. However, these model are too abstract to keep the practical solution in accordance with them completely. Since the B method is advantaged in developing the safety-critical software, our model can realize an executable module with consistency.

In terms of the trusted access control mechanism based on ARM TrustZone, Xu et al. [1] designed and realized a secure prototype system with multi policy, including the DTE (Domain and Type Enforcement) and BLP (Bell-La Padula) model. Kawamorita et al. [11] developed a separation kernel for embedded system with two formal methods. They employed the B method to devise the key part, which guarantees the efficiency and high-assurance. Chang et al. [12] proposed a usage and access control model on the ARM-Android platform for permission leakage, rather than formal security analysis. Combined the Android software stack with ARM TrustZone, our model is for formalization of the mechanism specifications and invariant properties, providing an extensible framework for security enhancement.

7. CONCLUSION AND FUTURE WORK
For multi-tier access control mechanisms on the ARM-Android platform, this paper develops a formal model with the B method, including the Android permission model and process model with TrustZone isolation. Based on the Atelier B, different kinds of control policies are specified in the system. The evaluation results show that all descriptions of proposed models are type checked and 75% (535 out of 718) proof obligations are proved automatically. Generally, our formal framework is an attempt to design the access control module, and we are already trying to enriching and refining the model by adding more actual structures and actions. We are also interested in applying model checker to test the formal model, which is capable of implementing a practical control module consistent with formal specifications.

8. References
[1] Xu, Y., Pan, W., and Zhang, X., 2008. Design and Implementation of Secure Embedded Systems Based on Trustzone. In International Conference on Embedded Software and Systems, 136-141. DOI=10.1109/ICESS.2008.59.
[2] Kanonov, U. and Wool, A., 2016. Secure Containers in Android. Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM’16. DOI=10.1145/2994459.2994470.
[3] Platform Architecture | Android Developers: 2017. https://developer.android.com/guide/platform/index.html. Accessed: 2017-05-01.
[4] Permission | Android Developers: 2017. https://developer.android.com/guide/topics/security/permissions.html. Accessed: 2017-05-01.
[5] Products | TrustZone – ARM: 2017. http://www.arm.com/products/security-on-arm/trustzone. Accessed: 2017-05-01.
[6] Presentation of the B method | Méthode B: 2017. http://www.methode-b.com/en/b-method/. Accessed: 2017-05-01.
[7] ARM, 2009. ARM Security Technology Building a Secure System using TrustZone Technology (white paper). ARM Limited.
[8] Shin, W., Kiyomoto, S., Fukushima, K., and Tanaka, T., 2010. A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework. In IEEE Second International Conference on Social Computing, 944-951. DOI=10.1109/SocialCom.2010.140.
[9] Betarte, G., Campo, J.D., Luna, C., and Romano, A., 2015. Verifying Android’s Permission Model. Springer International Publishing. DOI= 10.1007/978-3-319-25150-9_28.

[10] Fragkaki, E., Bauer, L., Jia, L., and Swasey, D., 2012. Modeling and Enhancing Android’s Permission System. Springer Berlin Heidelberg. DOI= 10.1007/978-3-642-33167-1_1.

[11] Kawamorita, K., Kasahara, R., Mochizuki, Y., and Noguchi, K., 2010. Application of Formal Methods for Designing a Separation Kernel for Embedded Systems. World Academy of Science Engineering & Technology, 68, 1303.

[12] Chang, R., Jiang L., Chen, W., He, H., and Yang, S., 2017. Enhancement of Permission Management for an ARM-Android Platform. In IEEE International Conference on Computer and Information Technology. DOI= 10.1109/CIT.2016.14.

Acknowledgments
Thanks to project supported by the National Natural Science Foundation of China (No. 61572516).