Risk-driven Security Objectives, Controls, and Metrics for an Android Smartphone Application

Reijo M. Savola

VTT Technical Research Centre of Finland, Oulu, Finland
E-mail: reijo.savola@vtt.fi

Markku Kylänpää

VTT Technical Research Centre of Finland, Espoo, Finland
E-mail: markku.kylanpaa@vtt.fi

Habtamu Abie

Norwegian Computing Center, Blindern, Oslo, Norway
E.mail: habtamu.abie@nr.no

Abstract:

Security management in Android smartphone platforms is a challenge. This challenge can be overcome at least partially by developing systematically risk-driven security objectives and controls for the target system, and how to offer sufficient evidence of its security performance via metrics. The target system of our investigation is an Android platform utilized for public safety and security mobile networks. We develop and analyse the security objectives and controls for these systems based on an industrial risk analysis. In addition, we investigate how effective and efficient security metrics can be developed for the target system, and describe implementation details of enhanced security controls for authentication, authorization, and integrity objectives. Our analysis includes implementation details of selected security controls, and a discussion of their security effectiveness. It also includes conceptualization and description of adaptive security for Android platform which can improve the flexibility and effectiveness of these security controls and end-users confidence in service providers.

Keywords: Android; security objectives; security metrics; security effectiveness; risk analysis.

1 Introduction

Nowadays, Android is the world's most widely used smartphone platform. Security management in Android platforms and applications is a challenge especially due to the openness of the system, its popularity and the specific difficulties in version control procedures. Attacks of various types make it possible to compromise an Android device and potentially critical information systems to which it has connections.

Well-designed and managed security metrics increase our understanding of the security level of the target Android system. Metrics should be designed to give efficient input to the main questions addressed by the security decision-making during the full lifecycle. They should be meaningful, measurable and correct. Prioritized security objectives, designed from the risk analysis results, are needed to steer the metrics development. Effective and efficient evidence of configuration correctness, system quality and adequate implementation of security controls help to manage Android security in a systematic way.

Most mobile platforms offer built-in security mechanisms to protect users from various types of malware, but these security mechanisms cannot prevent the rising number of attacks on these platforms. Therefore in the literature various improvements are proposed to the built-in security mechanisms of mobile platforms, specifically to the Android platform due to its market takes-up, and freely available source code [1], [2]. The permission system in Android is predominantly static, that is, users have no means of controlling the runtime behaviour of applications and this lack of dynamic security mechanisms is a design principle [3]. However, many authors argue for the need for customizable security policies for Android devices according to some regulations for many critical applications such as military and governmental applications [4], and for a context-aware adaptive security framework for eliciting context information such as location, time, network, and dynamically adapting the security settings of mobile applications for different situations and user actions [5]. We therefore argue that the successful deployment of mobile applications depends on ensuring security and privacy that need to adapt to the mobile devicesâŁ™ processing capabilities and resource use, which can be met through the development of adaptive and context-aware security for the next generation of digital ecosystems, which will improve end userâŁ™s confidence in service providers [6].

This study is an enhanced study to our earlier contribution published in [7]. Our earlier work contained a proposal of guidelines for security objective and control description, and security metrics development for the target system, and a discussion of preliminary implementation considerations of the resulting security controls of authorization and integrity objectives which have been demonstrated in the form of enhancements to Android. The results were based on a risk analysis presented in [8]. The main enhancements of this study in comparison to [7] are (i) presentation of performance measurements carried out in the demonstration system, (ii) discussion of the effectiveness of the identified security controls, (iii) more detailed analysis of the security controls implemented in the demonstrator, and (iv) more profound analysis of various selected topics handled in [7], (v) Conceptualization and analysis of adaptive security at various levels of an Android system.

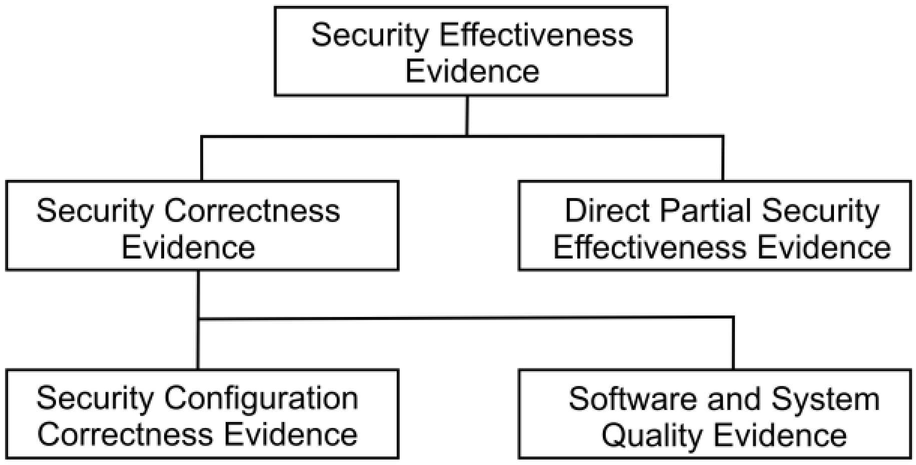
Security Effectiveness Evidence
Security Correctness Evidence
Direct Partial Security Effectiveness Evidence
Security Configuration Correctness Evidence
Software and System Quality Evidence

Figure 1 Example factors contributing to SE [13]

2 Background

2.1 Security effectiveness, objectives, controls and metrics

Adequate *security effectiveness* (SE) level of the System under Investigation (SuI) is the primary concern of security decision-making and consequently, security measurement activities. SE is the assurance that the stated *security objectives* (SOs) are met in the SuI and the expectations for resiliency in its use environment are satisfied, while at the same time the system does not behave in a way other than intended [9], [10], [11]. SOs are high level statements of intent to counter identified threats and/or satisfy the organisational security policies and/or assumptions made [12]. *Security controls* (SCs), the actual security solutions, are developed based on the SOs, taking into account the context and limitations of the actual system and its environment.

Carefully designed security metrics can be used to model the SE, and systematically managed measurements offer effective and efficient input to security decision-making.

Unfortunately, SE can be measured with a relatively high degree of accuracy only during long periods of actual operation of the SuI, when it is exposed to real *security risk occurrence*. However, in this case, the accuracy deteriorates due to the dynamic evolution of the risk landscape. Penetration testing is often used to obtain evidence of SE, but this kind of testing has many limitations compared to a life case, offering only partial evidence.

Because of the major challenges of measuring real security risk occurrence, when tried directly SE measurement can only be partial. Indicators of *direct partial SE*, security correctness, and software and system quality are different factors, high-level evidence, contributing to the overall SE evidence: see Fig. 1 [13]. In addition, there are more specific categories of factors contributing to SE, depending on the system characteristics, context. The categories shown in the figure are common to most systems.

Security correctness is a key factor that contributes to the SE of the SuI due to its concreteness. For example, policy and requirements compliance measurements belong to this category. However, it must be noted that good compliance does not automatically imply

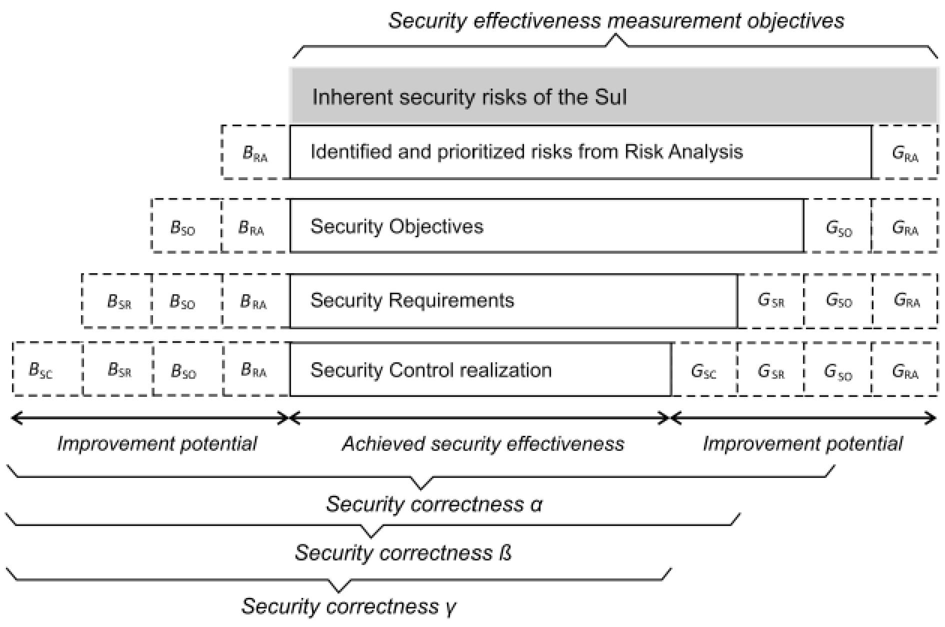

Figure 2 Visualisation of information gaps and biases between RA results, SOs, security requirements, and SCs [14]

good SE, and proper Risk Analysis (RA) or at least the use of best practices is required. The quality of the RA has a crucial role in the definition and maintenance of SOs.

The reference requirements used in security decision-making are characterized to be based on (i) security risk, and (ii) best practices or regulations. Risk-driven security requirements directly assume the presence of RA, while the latter ones do not. Security can only be managed to some extent based on best practices.

A severe problem in using best practices is that sometimes SOs and SCs are biased from the needs raised by the original RA. When the SE level is goal of security measurement, sufficient risk knowledge is needed. In practice, there are various gaps and biases. Fig. 2 visualises the gaps and biases between RA results, SOs, security requirements and SCs. First of all, there is information gap between RA results and SOs because in an RA it is not possible to identify and prioritize all the actual risks. Difficulties in understanding the SuI or risk situation can cause bias too. Further gaps and additional bias are introduced when developing SOs, requirements and the actual SC realization. As shown in the figure, security correctness measurements in practice are always an approximation of SE. In practice, due to the gaps and biases, SE can be achieved only asymptotically. Fig. 2 illustrates the achieved effectiveness as the intersection set of RA results, SOs, security requirements, and SCs. The gaps and biases can be reduced in making them more evident via metrics, and by adequate reactions to this evidence during the course of the security engineering or management process [14].

Security requirements, mentioned in Fig. 2, are an intermediate practical phase between SOs and SCs. In order to simplify the analysis, we have omitted investigation of this phase from this study. Fig. 2 visualizes also some alternative reference concepts for security correctness.

2.2 Basis for metrics: iterative risk analysis

Sufficiently detailed SuI-specific security risk knowledge is essential to the effective design of SOs and security metrics. The RA should be carried out in an iterative way. Each cycle should integrate the up-to-date SuI and security risk information. The initial phase of the RA is often conducted when product requirements are defined, the second phase when the product is being specified, and the third phase when the product is under design and verification. The risks should be prioritized, taking into account the integrated impact resulting from the estimated severity and probability of risks.

To enable the analysis presented in [7], [8] and this study, a RA was performed in co-operation between Android experts with Elektrobit Wireless Communications Ltd., and security researchers from VTT Technical Research Centre of Finland. The results of the RA were reported earlier in [8]. The process started with a risk identification expert brainstorming wherein participants were divided into two teams âŁ“ both comprising persons from each organisation with enough expertise in Android technology and security challenges and solutions. The risk sets were later combined via removal of duplicates and merging of risk categories. Finally, each riskâŁ™s probability and the severity of its consequences were rated. Consequently, the first meeting produced a list of risks, probabilities, and severity estimates for consequences, alongside a set of threats and attacks [8].

The âŁ˜rawâŁ™ prioritisation results were ordered in view of expert opinions. Expert opinion was needed at this point, because the risksâŁ™ abstract nature calls for case-by-case decision-making on whether severity or probability is more important. In general, severity was stressed a bit more because public safety and security (PSS) mobile networks are safety- and security-critical. It should be noted that risk prioritisation is not unambiguous, and small changes in system assumptions, especially in the system use scenarios, can change it [8].

2.3 Risk-driven security metrics development by hierarchical SO decomposition

As a basis of security metrics development, we use the hierarchical SO decomposition approach, originally described in [15]. Fig. 3 visualizes the decomposition principle [16]. Basic Measurable Components (BMCs) are leaf components of decomposition that clearly manifest a measurable property of the SuI [15]. Often the term Base Measure (BM) is used for BMC. BMCs can be seen as the core concept for security metrics. The actual security metrics, Derived Measures (DMs) are developed based on the BMCs. The high-level BMCs that can be deduced the sketch in Fig. 3 are Authentication Mechanism Reliability, Authentication Mechanism Integrity, Authentication Identity Structure, Authentication Identity Uniqueness and Authentication Identity Integrity [15]. The BMCs of Fig. 3 can be further decomposed, taking better into account specific system characteristics, resulting to a number of sub-nodes. The number of nodes in the hierarchy grows much larger in practice: for example, authentication decomposition incorporating more than 100 BMCs. An example of a more detailed authentication decomposition is shown in [14].

3 System under investigation

The target SuI of the study is briefly introduced in the following.

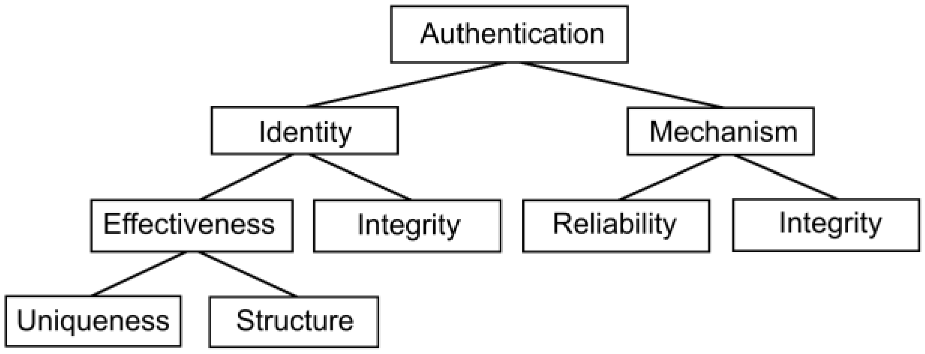

Figure 3 A simplified example authentication decomposition based on [16]

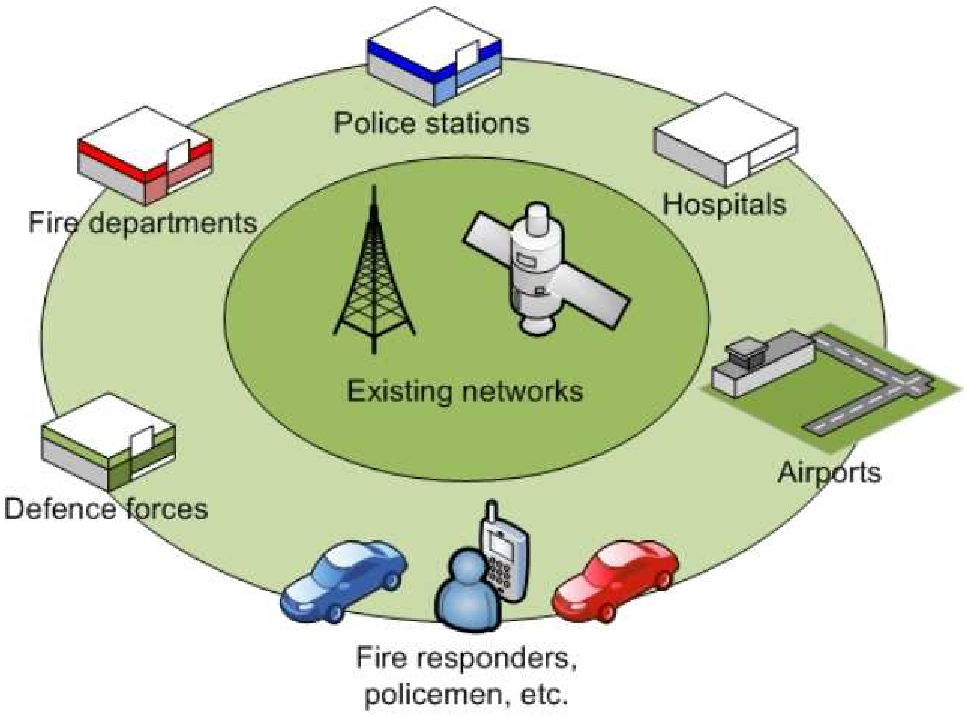

Figure 4 Use of the SuI [8]

3.1 Target System in General

The target SuI in this study is a public safety and security (PSS) mobile network system using the Android platform. The main services and characteristics of PSS mobile networks are high availability, group communication and encryption services [17]. PSS mobile networks utilize a mobile infrastructure that is very similar to cellular networks [18]. A major difference is that PSS mobile networks incorporate dispatcher stations for managing the communication of the user groups. Fig. 4 visualises the use of the SuI.

3.2 Focus of Investigation: Android Platform

The majority of the Android platformâŁ™s security solutions originate from Linux. Memory, process, user, and access control permission management are provided by the Linux kernel. However, they have been modified from traditional desktop usage. A major challenge with AndroidâŁ™s modified Linux kernel is the slow upgrade process: In many older versions, the version of the kernel deployed is clearly out of date, and many users have devices whose Android version is no longer upgraded by the device manufacturer. Furthermore, a remarkable challenge affecting to the overall security level is that understanding system resource permissions can be difficult for users. During installation of an application, the user is able to see the required permissions but have only two options to choose from: to let the application access all desired resources, or not to install the application at all. This lack of better control is a challenging situation from a security perspective, because there have been [19] and still are [20] several malicious applications in Google Play and other marketplaces.

To investigate advanced SCs and security measurement in the target system, a demonstrator has been developed using an Elektrobit RaptorPad Android device. PandaBoard SE development board [21] was also used in tests. The demonstrator includes advanced SCs for access control and integrity measurements.

4 Risk-driven Security Objectives and Controls

An effective and efficient risk-driven approach assumes development of SOs based on the prioritized risks rather than best practices. As our goal is high SE, we use the prioritized RA results as the basis. Often in practice, though, the SOs are based on risk management (RM) decisions. The RM can choose the a risk to be mitigated, cancelled, or accepted, whereas plain RA results include the whole collection of risks without feedback from decision-making.

In general, SO definition should be carried out in priority order. However, this process is not straightforward as there is no one-to-one mapping between the risks and core abstractions, SOs and SCs. For example, the most critical risk, R1 (unauthorised input of falsified data) in Table 1 has interdependency with R7 (loss of life). Even though both are critical risks, the criticality of R1 should be even more stressed than before investigation of the interdependencies. A risk may enable several other risks. For example, R1 is able to cause various unexpected risks.

The interdependencies between the risks can easily cause the process of SO definition to be tough. Even though there are many interdependencies among risks, they should be listed in the manner similar to Table 1. Otherwise, information about the prioritization is lost, potentially impeding SE in the SCs to be implemented.

Table 1 lists the top 10 risks out of the 26 identified ones for the PSS case (Case 1 in [8]). The rank of each risk is shown by the number in the first column. âŁ˜SâŁ™ refers to the severity of the consequences if the risk is actualized, and âŁ˜PâŁ™ denotes the probability of the risk being realized. The scale for each is 0âŁ“3 from âŁ˜no riskâŁ™ (0) to âŁ˜extremely high riskâŁ™ (3) S or P, with increments of 0.25. âŁ˜R:âŁ™ denotes âŁ˜risk arising fromâŁ™, and is used in connection with *attack types*, *vulnerabilities*, and *faults* that cause a risk. These challenges often dominate the discussion during RA sessions rather than the risks itself. The prioritization was carried out using expert opinions, because many

Table 1 Top 10 prioritised risks for the PSS Case 1 [7]

R#	Description	S	P
1	R: unauthorised input of falsified data	3.00	2.00
2	R: unavailability of the PSS Network at a critical moment (DoS, denial of service)	3.00	2.00
3	R: unauthorised root access	2.75	1.25
4	R: malicious loading of remote code	2.75	1.00
5	R: critical security functionality deployed in software (SW) but designed for hardware (HW)	2.25	3.00
6	R: network shut down due to device problems	3.00	0.50
7	Loss of life due to lack of resuscitation	3.00	0.50
8	R: activation of dormant malware at a critical moment	2.50	1.00
9	R: investigation of the target device in a laboratory environment	2.50	1.00
10	R: utilisation of open interfaces for attacks	2.25	1.00

quantized results were quite close to each other, and trying to use an automated approach according to a suitable formula depending on S and P would result to undesired outcome.

The results from RA, SOs and SC descriptions are discussed only as examples, because the actual RA/SO/SC/metrics process is the focus of this study.

Table 2 elaborates the SOs and SCs of the main risks listed in Table 1. It is obvious that most of the SOs address the main security dimensions according to the âŁ˜CIA modelâŁ™, confidentiality, integrity and availability. The text in them is shortened, emphasizing the main objectives. In the SC column, âŁ˜TâŁ™ indicates technical controls, whereas âŁ˜MâŁ™ refers to management (non-technical) controls.

It is obvious from the content of Table 2, that there are a lot of commonalities between different SOs and different SCs. As expected, authentication and authorization is a core security control. Configuration correctness and integrity enforcement also play an important role in the mitigation of these top prioritized risks.

Interdependencies of different SOs can be very different from the interdependencies of different risks. This is expected, since moving from the analysis of risks to the analysis of how one should protect from them is a significant abstraction change, yet the risk knowledge is needed when thinking about the SOs. For example, consider the case of R2 and R7: Although R2 and R7 are quite different, the resulting SOs are reminiscent of each other. However, there are two big differences: SO2 addresses networks and aims at preserving confidentiality, whereas SO7 also addresses devices, and aims to provide a quick response with the help of the PSS system, and if needed, does not pay much attention to *information* confidentiality. SO5 and SO8, and the respective SCs [22], are very similar.

SO7 is an example of a particular trade-off situation, which may be challenging to implement in practice. In SO7, the trade-off is between security and availability. In life-threatening situations, possible in PSS systems, common sense tells one not to pay so much attention to security procedures such as authentication and authorization. On the other hand, shortcuts designed to the system can be abused by attackers.

Table 2 Security objectives and controls for Table 1, based on [7]

R#	SO	SC
1	(i) Allow only *authorized* persons to use network infrastructure and devices, (ii) ensure *integrity* in input data	(i) Sufficient authentication and authorization network infrastructure equipment and devices (T), (ii) integrity enforcing mechanisms in data communication (T), (iii) enforcement of authorization policies (M)
2	Ensure the high *availability* of the network at critical use situations, with enough *confidentiality*	(i) Network resource management for authorized prioritization (T), (ii) intrusion detection and network traffic monitoring (T), (iii) sufficient authorization in devices and network infrastructure equipment (T), (iv) enforcement of authorization policies during critical moments (M)
3	(i) Allow only *authorized* persons to use network infrastructure and devices, (ii) ensure that no procedures make unauthorized root access possible, (iii) ensure *confidentiality* of authorized root access procedures	(i) Sufficient authentication and authorization in devices and network infrastructure equipment (T), (ii) systematic management of correct configuration (T), (iii) proper testing of apps to be used in the PSS scenarios, (T,M) (iv) enforcement of authorization policies (M)
4	(i) Allow only *authorized* persons to use network infrastructure and devices, (ii) ensure that no procedures make unauthorized root access possible, (iii) ensure *confidentiality* of authorized remote code procedures	(i) Sufficient authentication and authorization in devices and network infrastructure equipment (T), (ii) systematic management of correct configuration (T), (iii) proper testing of apps to be used in the PSS scenarios, (T,M) (iv) enforcement of authorization policies (M)
5	Ensure high *integrity* of critical security functionality	(i) Systematic management of correct configuration (T), (ii) high-quality testing of critical security solutions (T,M)
6	Ensure sufficient *integrity* of communication, networks and devices, and do not allow unauthorized actions	(i) Integrity and authenticity enforcing mechanisms in devices, network infrastructure and data communication (T), (ii) enforcement of authorization policies (M)
7	Ensure the high *availability* of the network and devices at life-threatening critical use situations, even with some *loss of confidentiality* of information	(i) Network resource management for authorized prioritization (T), (ii) intrusion detection and network traffic monitoring (T), (iii) use policies during critical moments (M)
8	Ensure sufficient *integrity* of devices	(i) Systematic management of correct configuration (T), (ii) proper testing of critical security solutions (T,M)
9	Ensure sufficient *confidentiality*, *integrity* and *authorization* of critical functionality, and enforce revokation procedures from the network when a device is investigated by intruders	(i) Sufficient authentication and authorization in devices and network infrastructure equipment (T), (ii) side-channel attack protection, (iii) mechanisms to revoke device's rights
10	Ensure sufficient *confidentiality*, *integrity* and *authorization* in critical points near open interfaces	(i) Sufficient authentication and authorization in devices and network infrastructure equipment (T), (ii) integrity enforcing mechanisms in data communication (T), (iii) enforcement of authorization policies (M), (iv) correct firewall configuration (T).

Table 3 Security control categories for the PSS Case 1

R#	Security Control Category	SOs	N
1	Authentication and authorization	SO1, SO2, SO3, SO4, SO9, SO10	6
2	Integrity mechanisms and correct configuration	SO1, SO4, SO5, SO6, SO8, SO10	6
3	Security testing	SO3, SO4, SO5, SO8	4
<4	Intrusion detection and traffic monitoring	SO2, SO7	2
5	Side-channel attack protection	SO9	1
6	Firewall	SO10	1

5 Security Metrics

In the following, we propose guidelines for security metrics development based on the risk-driven SOs. They consist of (i) security control categories, (ii) security effectiveness abstract models, and (iii) BMCs. The first two are discussed in the first two subsections, and the third one in subsequent subsections concentrating on selected core BMCs. The proposed BMCs are modified from the ones proposed in [15].

5.1 Security control categories

From the SOs and SCs, one can establish the core security control categories to enable security metrics development.

The following SC categories can be identified from Table 2: (i) authentication and authorization, (ii) integrity mechanisms and correct configuration, (iii) security testing, (iv) intrusion detection and traffic monitoring, (v) side-channel attack protection and (vi) firewall. The mapping between these categories and the SOs of Table 2 is shown in Table 3.

The categorization of SOs and SCs helps in developing Security Effectiveness Abstract Models (SEAMs), the next step in metrics development according to the process described in [13]. Note that there can be several different SOs and SCs in one SC category. For example, access control can be designed for network equipment management, for the end-user device or for some other purpose.

5.2 Security effectiveness abstract model

SEAM [13] is an abstract decomposition model that encompasses the core knowledge of factors contributing to the SE of the SuI. For example, an authentication SEAM can be developed using the information in Fig. 3. In [13], six strategies for security measurement objective decomposition were proposed, consisting of basic and integrated strategies. The *basic strategies* addressed direct partial security effectiveness, software and system quality and security configuration correctness. *Integrated strategies* were proposed for trade-offs, for pure security effectiveness, and for compliance measurements.

There is a need for SEAMs for all the security control categories mentioned above. A particular SEAM for SW and system quality should be developed in addition because in Android there are many software-related quality concerns. Applicable vulnerability database information should be integrated to the metrics hierarchy resulting from this SEAM.

Compliance with regulations is crucial for the use of a PSS mobile network system. In addition to the SEAMs for all SC categories and SW and system quality, SEAM compliance is clearly needed, although compliance concerns are not listed in the RA. The compliance measurements use best practice documents and regulations as their reference models.

5.3 Adaptive Security and BMCs

Integrating adaptive risk management into the SuI allows continuous monitoring of SE and gives insight on impeding risk. Providing anticipatory self-adaptive risk analysis models in near real-time with the ability to identify, predict, and react to potential threats proactively will allow us to adapt to the dynamic nature of the threats and their ability to spread in very short time intervals. The envisioned SuI can be used in varying contexts (e.g. fire departments, police stations, hospitals, airports), and the associated information confidentiality and privacy regulations are highly varying depending on the context. The RA based results presented in Table 2 and 3 offer a starting point for adaptive techniques. Different prioritized results, depending on the context, will be selected to the basis for different adaptive scenarios. Moreover, the changing risk landscape will be taken into account. Adaptive security management solutions are crucial for high-quality security management of the system. In practice, adaptive techniques adjust internal working parameters, and make dynamic changes in the structure of the security countermeasures. Examples of these internal parameters are security algorithms, authentication and authorization mechanisms, encryption schemes, security protocols, and security policies. The core input needed by adaptive security solutions include SE level information, communicated by appropriate SC, security metrics, and contextual information. In [22], we proposed an adaptive security management for learning and adapting to changing environment dynamically and anticipating unknown threats. Moreover we developed a context-aware Markov game theory model for security metrics risk impact assessment to estimate and predict risk damages and future benefits and adapt security decisions upon those estimates and predictions. The security metrics are utilized for measurably evaluating and validating the run-time adaptivity of IoT security solutions [23]. Adaptive security management here refers to a security management solution that is able to learn and adapt to changing environment dynamically and anticipates unknown threats [6], [22]. The potential risk impacts of the threats to the BMCs can be calculated using the metrics that measure the effectiveness of the security services they provide. Using such an adaptive risk impact assessment method the security metrics can be quantified according to how efficient the security services of the SuI are.

6 Enhancements of Security Controls in Demonstrator

Table 1 describes prioritized risks for the PSS device. The risks are analysed and SOs with related SCs are identified and described in Table 2. SCs are then categorized and their relation to SOs is described in Table 3. The analysis clearly shows that SCs related to authorization and integrity mechanisms are dominating ones. Enhancements to access control and integrity protection mechanisms should be developed to enhance security controls related to these categories. In the following, we discuss some implementation-level enhancements to mandatory access control (MAC) and integrity protection, implemented in our target system demonstrator.

The traditional desktop approach to malware protection has been antivirus software. However, controls such as antivirus software and malware scans for apps from application stores, are reactive solutions to the malware problem. The proactive solution is to harden the platform itself, so that attacking it is more difficult. Hardening should be done without breaking legacy applications, which means that all userspace modifications should be minimized. However, PSS devices could be considered to be special devices that are only meant to run specific applications, relaxing this compatibility requirement slightly. The Android kernel, which is based on the Linux kernel, contains many security frameworks which can be enabled and configured in a way that is still compatible with legacy applications.

6.1 Mandatory access control

SEAndroid: Mandatory Access Control provides access control restrictions that can be overridden only by administrator, typically by configuring security policy. This contradicts with more traditional Discretionary Access Control (DAC) where e.g. the owner of a file can control access to the file by setting file permissions. SELinux is a MAC implementation for Linux originally developed by National Security Agency (NSA). Since Android is based on Linux kernel it is also possible to utilize SELinux in Android. However, this is not straightforward as Android userspace is totally different from traditional Linux userspace. SEAndroid is SELinux architecture ported to Android.

Google has now integrated SELinux-based SEAndroid [24] to Android. SEAndroid is used to control a Dalvik virtual machine (VM) running Java bytecode and Interprocess Communication (IPC) mechanisms. Since Android release 4.4 (KitKat), SEAndroid is run in enforcing mode, making it the default choice to enhance access control. SEAndroid is mainly used to protect Android system software. Java-based system applications are classified to pre-defined SEAndroid security domains (*media_app*, *platform_app*, *shared_app*, and *release_app*) based on their origin and installation package signing. Isolated applications are using *isolated_app* domain. Other Java applications are mapped to *untrusted_app* domain. Application compatibility requirements with older Android releases have limited the ways in which MAC can be applied to third-party applications as modifications should not break Android Compatibility Test Suite [25].

Smack: SELinux was the first MAC framework in Linux kernel, but it is not the only one. Another Linux upstream kernel, MAC framework Smack [26], which is now also used in the Tizen operating system [27], could be a potential alternative to SEAndroid, but provides only limited functionality, and extensions to control middleware are less clear than in SEAndroid. Although it would have been straightforward to use SEAndroid as a MAC framework, we considered using Smack and also implemented Smack support in the demonstrator. Both SEAndroid and Smack are based on access rules for subjects (e.g. processes) and objects (e.g. files). File system labels are stored in extended attributes of the file system.

Modifications to AOSP code: Android Open Source Project (AOSP) source code was used as a basis for the demonstrator. The modifications required to support Smack were quite straightforward, as required changes were closely correlated to areas where the SEAndroid project has also modified the AOSP source code. Kernel-level changes were:

1. Kernel configuration modifications âŁ“ enabling Smack, auditing, and network security related settings required by Smack. The use of extended attributes was also enabled, along with separately the use of security labels in extended attributes.
2. Adding Binder IPC Linux Security Module (LSM) hooks using a patch from SEAndroid [28]. The patch adds four new LSM hooks. The hooks can be registered and used in security modules.
3. Registering and implementing Binder IPC LSM hooks in Smack.

Userspace changes included:

1. Init process was modified to mount Smack filesystem and to load Smack policy during boot. Smack userspace library was used [29]. Also the file init.rc was modified to include Smack security label settings for processes stared by init.
2. Initial ramdisk (initrd) was used to store initial Smack policy.
3. Dalvik VM was modified to support Smack labels.
4. Zygote application launcher was modified to support Smack labels.
5. Toolbox commands *ls* and *ps* were modified to display Smack labels.

There were also changes to Android build tools to support Smack and Smack labels:

1. File system labelling tools for ext4 filesystem were modified to support also Smack labels. Device file labelling is still missing in our demonstrator but could be added, utilizing ideas from [30].
2. Smack specific files and tools were included in system images.

Security policy: The main reason to use Smack instead of SEAndroid as a MAC framework is that it is simpler and has more understandable security policy definitions. Recent kernels also contain many Smack-related changes, e.g. providing support for longer labels and inheriting file security labels for new files from directory labels instead of process labels (so-called transmute functionality). However, we are using only classic Smack features, as some of our test systems were using rather old kernel versions. As a starting point we tried to emulate SEAndroid security policy. Security policy development turned out to be time-consuming and difficult, because of the lack of an equivalent to the SELinux permissive mode in Smack. Such a mode was proposed for Smack, but the idea was rejected by the author of Smack [31]. Also, SEAndroid policy emulation is not the best approach for initial security policy development. Simpler alternative would have been to just separate the system into two domains (system vs. third-party applications) and then gradually refine this isolation.

Another drawback was recognized when the Smack model was used to control Android Binder-based IPC. According to Smack documentation [32], sockets are data structures attached to processes, and sending a packet from one process to another requires that the sender must have write access to the receiver. The receiver is not required to have read access to the sender. As Smack recommends using file write permission also to control IPC access, adding Smack rules could open unnecessary write access for certain files. This could be prevented by adding a new IPC-specific access method attribute to Smack. Currently

Smack access settings can contain settings âŁ˜rwxaâŁ™ (read, write, execute, and append). A lack of transmute functionality in our Smack version also caused problems. New files that were created by labelled processes inherited security labels from processes so that security labels that were intended to be used for processes were now also used as file labels. New Smack rules had to be created to solve these access problems.

The initrd location of Smack security policy file was also inconvenient. There should be a writable and updatable policy file that could be signed and could be loaded after verification. However, the initrd policy should be kept as a fallback and should support system restore operations.

One approach could be to use both SEAndroid and Smack simultaneously. There is an unofficial patch set called LSM stacking [33] to support multiple security modules in the kernel. SEAndroid could be used to protect system software using the AOSP code and there could be additional Smack rules to sandbox third party software. However, this approach is probably too complex. Currently there are no plans to integrate LSM stacking to the official Linux kernel. Major modifications to the AOSP code are not convenient as the code is tightly controlled by Google. Although the code is open source, the development model is not truly open. Modifications become visible only after releases, and Google does not share development plans in public. A large porting effort may be needed after releases.

6.2 Integrity protection and attestation

Kernel-based integrity protection frameworks can be used to protect Android systems against unauthorized system soft-ware modifications (e.g. utilizing offline attacks). Android release 4.4 (KitKat) includes an experimental block-based integrity scheme called dm-verity [34]. There are other alternatives such as the file-based Integrity Measurement Architecture Extended Verification Module IMA/EVM [35]. IMA maintains a runtime measurement list, which can be displayed by root access. These frameworks are meant for read-only filesystems. There is also a block-based alternative called dm-integrity [36] that can be used with writable filesystems. Block-based alternatives must also have storage to store reference block hashes.

The demonstrator is using IMA for integrity measurements. When a native application, a shared library or a shell script is loaded for execution SHA1 hash of the content is calculated and measurement is stored by including the hash value to a kernel internal storage variable using a so-called extend operation.

IMA supports only measurements, and there is no integrity enforcement. EVM component is for integrity enforcement, but it requires storage of integrity reference values to extended attributes and also signing these extended attributes and key management for verification keys. IMA/EVM concept requires the use of recent kernels, unless EVM part is replaced by a more straightforward HMAC-based approach. There were still important EVM-related modifications even in the recent Linux 3.16 kernel [37]. Another problem with IMA is that it only measures native applications and not Java-based Dalvik applications. Nauman et al. [38] have developed a framework that allows measurement of Java code running in Dalvik VM. Google is now replacing Dalvik VM with a new virtual machine called ART. ART is using install-time bytecode to native code conversion instead of load-time conversion used by Dalvik VM. This would simplify integrity protection as code loading can now be tracked by existing IMA mechanism in kernel without VM-level modifications. However, both Dalvik and ART are currently offered as options to users to select, so it is not possible to fully avoid VM-level modifications.

Table 4 Results of AnTuTu benchmark (N=30)

	AOSP		Smack	
	Mean	SD	Mean	SD
Total	5197.60	189.64	5265.00	211.57
Memory	924.73	55.49	944.33	52.72
Integer	1439.53	67.08	1478.20	77.33
Float	1165.70	64.59	1196.90	74.47
Score2d	315.23	10.99	304.96	10.56
Score3d	942.40	15.83	933.10	24.21
Database	410.00	83.66	407.50	74.27

An encrypted file system provides confidentiality and protection only against offline attack, but does not offer control point to execution of native code executables. The choice obviously depends on a solution domain-specific threat model.

7 Performance Measurements

Additional security framework is easier to justify if it does not impose significant runtime performance overhead. Performance measurements were made to compare performance of unmodified AOSP implementation and Smack-based MAC implementation ported to the same AOSP version. Performance measurements were made using PandaBoard SE hardware [21] based on AOSP Android 4.3 version utilizing two well-known benchmark applications called AnTuTu [39] and Softweg [40]. The same benchmark applications were also earlier used to verify performance of SEAndroid [24] and we present our measurements using the similar format as was done in [24]. AnTuTu and Softweg tests for SD card read/write performance hanged both in AOSP and Smack implementation so we had to omit those tests.

7.1 AnTuTu benchmark

AnTuTu version 2.9.1 was used in performance measure-ments. The benchmark includes tests for memory, integer, and floating point calculations. There are also tests for 2D and 3D graphics and database I/O. Benchmark tests were run 30 times both for AOSP and Smack implementations. Mean and standard deviation (SD) of measurements were calculated. Results are presented in Table 4 (Larger number is better).

These measurements show only minor performance differences. Memory, integer, and float tests, which should not be much affected by the security framework modifications, gave slightly better scores with Smack implementation. Test scores for 2D/3D graphics and database I/O were slightly better with AOSP implementation. However, differences were small and within limits of standard deviations.

7.2 Softweg benchmark

Android benchmark software Softweg contains also tests for computing power, graphics, and filesystem I/O. Also here benchmark tests were run 30 times both for AOSP and Smack implementations. Mean and standard deviation (SD) of measurements were calculated. Results are presented in Table 5 (Larger number is better except in file create/delete, which is given in seconds).

Also in these tests the measured benchmark figures between target systems were very close to each other. Memory access operations were slightly better with AOSP. There were 7 computing power measurements tests. AOSP implementation gave better results in four of those. As memory and computing power tests typically do not include system calls it is expected that security framework implementation does not affect much to these measurements.

Graphics benchmark results were slightly better with Smack. Filesystem access is expected to demonstrate performance degradation of utilizing security framework. There was a small difference in file creation and delete tests in favour of AOSP implementation. Also file writing was slightly faster with AOSP. However, file reading and also total score were surprisingly slightly better with Smack.

The test results using AnTuTu and Softweg gave very similar results as earlier tests performed for SEAndroid [24]. It seems that most of these tests measure performance of the underlying hardware so that these tests can be used to compare different Android devices. Another approach could be to measure delays of LSM hook calls as is done in [41].

8 Discussion

8.1 Access control

From the results of this study, it can be seen that there are six critical focus areas, or security control categories, described in Table 3, for security management in the target system. In the following, we discuss the findings regarding them.

The core security objective for the system is related to authentication and authorization. This objective has crucial role because inadequate authorization decisions can lead to a situation where users and potentially attackers can gain too wide privileges, making it possible to cause severe damages, as emphasized by the top 10 risk list. Solutions like the experimental Smack mandatory access control discussed above offer an adequate level of security effectiveness because processes can be sandboxed to allow access only to resources they need, following a principle of least privilege [42] by assigning Smack security labels to processes and files. However, developing effective security policy still remains a challenging task. Android system architecture with Dalvik virtual machine executing bytecode and Zygote application launcher require extending classic kernel-level MAC frameworks to userspace in order to implement fine-grained access control. Authentication and authorization metrics can be developed using the given guidelines and incorporating security effectiveness information about the authentication and authorization mechanisms in the implementation.

Table 5 Results of Softweg benchmark (N=30)

	AOSP		Smack	
	Mean	SD	Mean	SD
Total memory	206.52	26.08	203.47	28.05
Copy memory	187.66	23.70	184.89	25.49
Total CPU	3791.35	165.22	3762.94	187.56
MFLOPS DP	40.37	13.37	36.15	9.77
MFLOPS SP	72.17	22.04	67.21	17.00
MWIPS DP	241.77	9.65	243.38	13.49
MWIPS SP	309.07	8.82	309.18	13.35
VAX MIPS DP	182.04	13.88	183.38	23.69
VAX MIPS SP	207.97	14.26	206.21	19.09
Graphics				
Total score	476.04	51.24	483.89	56.37
Opacity	202.07	30.05	204.15	31.47
Transparent	89.08	7.34	91.82	7.81
Filesystem				
Total score	103.92	5.31	104.86	7.43
Create files	0.481	0.032	0.580	0.041
Delete files	0.255	0.009	0.281	0.009
Read file	152.00	11.62	154.83	15.83
Write file	57.07	1.92	56.12	1.78

8.2 Integrity protection

Integrity mechanisms and correct configuration is another main security control category. Without adequate integrity protection attackers may be able to install permanent modifications (so called rootkits) to system software by replacing original software components or by modifying configuration files so that the system becomes vulnerable. This could endanger privacy of the user and the infected device can also be used to attack against other connected devices. The filesystem containing Android system software is typically mounted as read-only, which prevents direct modifications. However, authorized process is able to remount the volume as read-write and then do the modifications. Adequate integrity protection should prevent access to files that have been modified without authorization. This requires that valid reference integrity metrics is available and that measurement and integrity verification is done using trusted code. New Android versions support dm-verity [34] that provides block-level integrity protection for read-only volumes. Integrity measurements using IMA or integrity protection mechanisms for read-write volumes (e.g. IMA/EVM) can also be used. There is clearly a need for attestation solutions, as plain measurements, such as utilizing IMA/EVM does not offer enough. Attestation supports building *confidence* in the measurements.

Load-time integrity protection does not help if software contains vulnerabilities like buffer overflows. Android access control framework can be used to isolate processes so that they have only just those permissions that are needed and access to those files that are necessary. This will limit the damage caused by infected process. Android DAC mechanism provides basic isolation and can be augmented by using MAC frameworks like SEAndroid or Smack to enhance isolation. Utilization of Address Space Layout Randomization (ASLR) would make this kind of attacks more difficult.

8.3 Vulnerabilities and advanced threats

Security testing can be used in development phase to increase robustness of applications. Developers should utilize tools like protocol fuzzing tools to create test material so that potential vulnerabilities and constructs enabling risks can be detected early. Many testing tools offer detailed-level metrics than can be used to offer partial evidence. In order to reason about the software and system quality, however, a wider perspective is needed. Combination of results from different testing tools, different testing practices and testing processes is needed in the software and system quality measurements.

Intrusion detection and traffic monitoring software can be used to detect anomalies and to create alerts. Integrity measurements provided by IMA can also be used to verify status of the system as remote system can send an attestation request. Attestation replies can be analysed and possible anomalies could trigger alerts.

Side-channel attacks were also listed as potential threats to devices in PSS mobile networks. Typical side-channel attacks are execution time [43] or power consumption [44] related attacks against cryptographic algorithms. Protection requires removal of correlation between an observable event and the secret (e.g. between encryption key and encryption time). However, a lack of awareness of these risks is common as these risks are not self-evident. Also new side-channels can be detected.

8.4 Network access

Many threats originate from network and can also be mitigated by strictly controlling network access. Android third-party applications require a special permission that should be granted at installation time in order to be able to access Internet. Firewalls can be used to isolate networks and services. Android Linux kernel contains a packet filter firewall mechanism called iptables that can be used to filter Internet Protocol (IP) packets. There is also a rarely used option called Common IP Security Option (CIPSO) to allow labelling sensitivity of IP packets [45]. This has typically only been used in governmental and military networks that have also utilized MAC frameworks [46]. Smack is able to utilize and process CIPSO labelled packets [47] so utilizing CIPSO labelling in PSS mobile networks could be an option.

8.5 Adaptation

Adaptive security solution can also be used to adapt to changing environment and context dynamically and anticipate unknown threat based on SE, correctness and efficiency evidence to respond to these needs. For instance adaptive authentication mechanisms can cope with changing context of use, security threats and the user behavior. Adaptive solutions can also be used to setting requirements and for enforcing the sufficient authentication mechanisms.

As we argued the successful deployment of mobile applications depends on ensuring security and privacy that need to adapt to the mobile devicesâŁ™ processing capabilities and resource use can be met through the development of adaptive and context-aware security for the next generation of digital ecosystems. Such adaptation can be conceptualized and described using the biological and ecosystem metaphors that provide interesting parallels to a complex adaptive system that utilizes autonomic systems mimicking biological auto-immune systems at the microscopic level (adapting and making decisions at individual component Android system in this case) and utilizing the behaviours of an ecosystem of disparate entities at the macroscopic level (adapting and making decisions about the run time operation of the system that require a wider perspective than the individual component in Android system in this case) [48]. Biological and ecological systems maintain system integrity by reacting to known changes, adapting to unknown changes, or dying .The adaptations and responses can be at a macroscopic ecosystem level (e.g., system or species) or a microscopic biological level (e.g., molecular, cellular), or at hybrid levels. The self-adaptive component achieves its goal through the following properties [49], [50]: (i) autonomy, which allows it to operate without the direct intervention of humans or others and to have some kind of control over its actions and internal state, (ii) social ability, which allows it to interact with other agents (possibly humans), (iii) reactivity, which allows it to perceive its environment and respond in a timely fashion to changes that occur in it (the environment), and (iv) pro-activeness, learning, and adaptiveness, which allow it to exhibit goal directed behaviour by taking the initiative, to learn when reacting and/or interacting with its external environment, and to modify its behaviour based on its experience.

9 Related Work

9.1 Security metrics

Haddad et al. [51] introduced an abstract model called Assurance Profile (AP) for security metrics definition. Its focus is on security assurance objectives, and risk-driven security management and engineering is not well supported by the approach. The security metrics decomposition approach discussed in this study is similar to the Goal Question Metrics (GQM) of Basili et al. [52] refining specification of software measurements. Unfortunately, GQM definitions lack guidelines to define security metrics. The generic challenges of requirement decomposition have been discussed by Koopman [53] and Kirkman [54]. As problems, they mention excessive hierarchy, excessive subsystem decomposition, insufficient de-composition, âŁ˜gamingâŁ™ promoted by too great a focus on goals, unattributed requirements and issues of change management. These problems assume that not much human interaction is used in the decomposition process, and that there are no tools available for decomposition management. There are already a variety of specific security metrics proposed in the literature, as summarized e.g. in metrics collections [55], [56], [57], [58]. These metrics can be used at the detailed level, when developing DMs from BMCs. In general, standards have achieved only limited success in advancing security metrics and measurement, because they are rigid and created for certification, and carrying out these processes requires significant amounts of time and money [59]. The most widely used of these efforts is the CC (ISO/IEC 15408) Standard [12] which focuses primarily on documentation rather than the actual security effectiveness of the operational system. The ISO/IEC 27004 standard [60] addresses measurement, reporting and improving the effectiveness of Information Security Management Systems (ISMS). However, this standard does not support technical systems well.

9.2 Android Security Frameworks

Android OS security risks were studied in general by Fedler et al. [61]. They summarize that most successful attacks affecting Android can be attributed to negligent user behaviour. However, they admit that attacks on Android devices are be-coming more sophisticated. Their conclusion calls for enough emphasis on security policies (management perspective). The security-critical case investigated in our original RA calls for a variety of security controls (and a variety of security metrics).

Before Google adopted the SEAndroid approach there were many examples of applying various MAC implementations in Linux kernel to Android. For example, TrustDroid [62] uses Tomoyo and FlaskDroid [63] is using SELinux. There are also many research prototypes that have tried to tackle Android related security issues in various ways. A brief overview of these is available in [64]. Also SEAndroid project developed modules that Google has not yet included to official Android. Nowadays there are also many Android firmware ROM variations that are utilizing AOSP source code, mixing it with vendor specific binaries. Some of these are also tackling security issues such as CyanogenMod [65] providing tools to revoke permissions from installed applications and providing fake services to less trusted applications [66]. Testing tools create statistical information about the tests. The metrics affecting security effectiveness should be selected to be part of the risk-driven security metrics hierarchy.

Samsung has extended Android SEAndroid concept in their Knox product [67], providing more isolated containers targeting bring-your-own-device (BYOD) enterprise customers. Other manufacturers have so far kept GoogleâŁ™s security approach, although some of them have replaced GoogleâŁ™s services with their own equivalents. Examples of these are Amazon and Nokia/Microsoft who provide Android devices without Google services. This is also common approach for Chinese manufacturers who have many local service alternatives. Android application ecosystem can be also utilized in different operating system by providing system emulation as in Jolla Sailfish case. Metrics related to side-channel attacks depend on the overall security strength of the algorithms used in the security solution, and metrics like mean-time-attack.

There are also attempts to provide an interface layer between Android system software and security module implementations. Android Security Framework (ASF) [68] and Android Security Modules (ASM) [41] provide such interfaces. Both systems are inspired by LSM mechanism and allow easy experimentation with new security frameworks without need to maintain large patch sets. Smack modifications could also be implemented using either ASF or AMS. Although this makes sense from research point of view there is no guarantee that Google will adapt any such framework as it could also create more fragmentation to Android.

9.3 Adaptive Security

Bauer et al. [3] described a dynamic security mechanism for Android-powered devices based on runtime verification which allows users to monitor the behaviour of installed applications. The authors outlined general idea and a prototype implementation, demonstrated an application to real-world security threats, and sketched the underlying logical foundations, relating to the employed specification formalism. They concluded that runtime verification is feasible on Android devices and can improve system security by identifying known and yet unknown malware, and pointed out ways to optimize the runtime performance.

Liang et al. [4] developed EAdroid, adaptive security mechanisms for Android platform by exploiting the framework layer of Android system and synthetically applying Smack security module of Linux. They argued that both the security rules of framework layer and kernel layer in EAdroid can adapt to the current environment context and concluded that their test results show that EAdroid can efficiently protect the security of usersâŁ™ devices and privacy with negligible overhead of performance.

Mowafi et al. [5] developed a context-aware adaptive security framework for eliciting context information such as location, time, network, etc. and dynamically adapting the security settings of mobile applications for different situations and user actions. The proposed framework consists of a mobile application incubator or sandbox which is built inside the mobile OS and conceals all mobile application data, code execution, and network access, and a context shadow application which is implemented as a shadow application so as to avoid any changes to the mobile OS. The authors have prototyped the framework for Andriod OS, evaluated using a facebook use case, and concluded for the efficacy of their framework in providing adaptive security measures based on real-time user context.

Enck et al. [1] developed Taint-Droid by extensively modifying the entire Android stack to track the flow of sensitive data on smartphones through third-party applications at runtime. The authors argued that TaintDroid can detect leakage of sensitive data by sending an email or SMS containing the sensitive data, or by uploading a file directly by keep tracking the use of âŁœtaintsâŁž sensitive information throughout the system.

Ongtang at al. [2] developed the Saint framework - a semantically rich application centric security in Android by modifying the Android application installer and AppPolicy Provider. The former ensures that only applications which do not violate policies stored in the latter can be installed at install-time. The Saint framework also checks permissions of existing applications for suspicious permission requests and derives practically useful policies to enforce.

Abie and Balasingham [6] proposed a novel risk-based adaptive security framework for Internet of Things (IoT) in eHealth. The framework estimates and predicts risk damages and future benefits using game theory and context-awareness techniques, and the security methods and mechanisms adapt their security decisions upon those estimates and predictions. The framework is based on a continuous cycle of adaptive risk management, adaptive security monitoring, predictive analytics, automated adaptive decision-making, and evaluation and validation security metrics. Savola et al. [22] further argued that adaptive security management is needed especially for setting the sufficient security requirements and for enforcing the adequate security controls in the face of changing security risks and use context and informed adaptive security decision-making is based on adequate security effectiveness, correctness and efficiency evidence offered by security metrics.

All of the above demonstrate the need and advantages of adaptive security to improve and increase the strength of security and the degree of trust in the system. Providing anticipatory self-adaptive risk analysis models and risk based metrics in near real-time with the ability to identify, predict, and react to potential threats proactively will allow us to adapt to the dynamic nature of the threats, solve the problem of limitations in the robustness and resilience of an Android system and its performance, and improve its reliability and robustness. However, adaptivity has also some disadvantages: its effectiveness depends on the correct definition of security goals; it requires additional resources to carry out the adaptation processes, and it is not always able to ensure only minimal deviations in the systemâŁ™s normal mode of operations while it is adapting.

10 Conclusions and Future Work

Risk-driven security engineering, management and metrics development bases the development of security objectives and controls on the sufficient risk knowledge. We analysed security objectives and security control categories for an Android platform utilized for a public safety and security mobile network based on iterative industrial risk analysis results. During the risk analysis, it was discovered that there were many interdependencies between the original risks. Furthermore, security objectives and controls show a different pattern of interdependencies. However, the original risk analysis results should be preserved to enable decision-making about the relative importance of the risks, and weighting to be used in the metrics formulas.

The core security controls for the target system are authentication and authorization, confidentiality and integrity controls. In particular, access control plays an important role in the target system, where there are health and societal-critical usage scenarios. There are a lot of vulnerabilities in Android platforms, and many of them can give root access. Therefore, software and system quality assurance are crucial for the system.

We also argued that the successful deployment of mobile applications depends on ensuring security and privacy that need to adapt to the mobile devicesâŁ™ processing capabilities and resource use. This can be achieved through the development of adaptive and context-aware security for the next generation of digital ecosystems. We used the biological and ecosystem metaphors that provide interesting parallels to the conceptualizations and descriptions of the adaptions and responses which can be at a macroscopic ecosystem level (e.g., system or species) or a microscopic biological level (e.g., molecular, cellular), or at hybrid levels.

We proposed guidelines for security metrics development, based on the risk-driven security objectives. The guidelines are categorized by security controls, security effectiveness abstract models, and basic measurable components.

We also described some implementation-level enhancements to mandatory access control and integrity protection, implemented in our target system demonstrator. Experimental Smack access control framework and IMA-based integrity measurement framework were discussed. It should be noted that although open source Android can be used to experiment with new features, the lack of an open development model in Android can make custom modifications hard to maintain in the long run.

In our future work, we plan to focus on defining detailed security metrics for the target system based on the guidelines, managing the metrics by a visualization tool, and gathering validation information from deploying a system similar to the demonstration system in real or realistic use scenarios. We also plan to enhance the integrity measurement part to support remote attestation that would be an important use case for PSS devices.

Acknowledgements

The work presented here has been carried out in three research projects: the IoT Program (2012-2015), launched by the Finnish Strategic Centre for Science, Technology and

Innovation Digile Ltd., ASSET âŁ“ Adaptive Security for Smart Internet of Things in eHealth (2012âŁ“2015) funded by The Research Council of Norway, and the SAICS (Situation Awareness In Information and Cyber Security) project (2014-2016) funded by Academy of Finland and VTT.

References

[1] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[2] Machigar Ongtang, Stephen McLaughlin, William Enck, and Patrick McDaniel. Semantically rich application-centric security in android. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, pages 340–349, Washington, DC, USA, 2009. IEEE Computer Society.

[3] Andreas Bauer, Jan-Christoph Kuester, and Gil Vegliach. Runtime verification meets android security. In *4th NASA Formal Methods Symposium (NFM)*, pages 174–180, Norfolk, Virginia, USA, apr 2012. Springer-Verlag.

[4] Hongliang Liang, Yu Dong, Bin Wang, and Shuchang Liu. Eadroid: Providing environment adaptive security for android system. In Dongdai Lin, Shouhuai Xu, and Moti Yung, editors, *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers*, volume 8567 of *Lecture Notes in Computer Science*, pages 118–131. Springer, 2013.

[5] Yaser Mowafi, Dhiah Abou-Tair, Tareq Aqarbeh, Marat Abilov, Viktor Dmitriyev, and Jorge Marx Gomez. A context-aware adaptive security framework for mobile applications. In *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, ICCASA '14, pages 147–153, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[6] Habtamu Abie and Ilangko Balasingham. Risk-based adaptive security for smart iot in ehealth. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 269–275, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[7] Reijo M. Savola and Markku Kylänpää. Security objectives, controls and metrics development for an android smartphone application. In *2014 Information Security for South Africa, Johannesburg, South Africa, August 13-14, 2014*, pages 1–8. IEEE, 2014.

[8] Reijo M. Savola, Teemu Väisänen, Antti Evesti, Pekka Savolainen, Juha Kemppainen, and Marko Kokemaki. Toward risk-driven security measurement for android

smartphone platforms. In *2013 Information Security for South Africa, Johannesburg, South Africa, August 14-16, 2013*, pages 1–8. IEEE, 2013.

[9] Reijo Savola. A security metrics taxonomization model for software-intensive systems. *JIPS*, 5(4):197–206, 2009.

[10] Wayne A. Jansen, National Institute of Standards, and Technology (U.S.). *Directions in security metrics research [electronic resource] / Wayne Jansen*. U.S. Dept. of Commerce, National Institute of Standards and Technology Gaithersburg, MD, 2009.

[11] Christian Jahl. The information technology security evaluation criteria. In Les Belady, David R. Barstow, and Koji Torii, editors, *ICSE*, pages 306–312. IEEE Computer Society / ACM Press, 1991.

[12] ISO/IEC. Internal Organization for Standardization and the International Electrotechnical Commission, 2005.

[13] Reijo M. Savola. Strategies for security measurement objective decomposition. In Hein S. Venter, Marianne Loock, and Marijke Coetzee, editors, *2012 Information Security for South Africa, Balalaika Hotel, Sandton, Johannesburg, South Africa, August 15-17, 2012*, pages 1–8. IEEE, 2012.

[14] Reijo M. Savola, Christian Frühwirth, and Ari Pietikäinen. Risk-driven security metrics in agile software development - an industrial pilot study. *J. UCS*, 18(12):1679–1702, 2012.

[15] Reijo Savola and Habtamu Abie. Development of measurable security for a distributed messaging system. *International Journal on Advances in Security*, 2(4):358–380, 2009.

[16] C. Wang and W. A. Wulf. Towards a framework for security measurement. In *Proceedings of the Twentieth National Information Systems Security Conference*, pages 522–533.

[17] Matti Peltola. *Evolution of public safety and security mobile networks*. Aalto University School of Electrical Engineering,, New York:, 2011.

[18] D. Gray. *TETRA Advocate's Handbook, From Paper Promise to Reality*. Looe, Cornwall, 2003.

[19] Joany Boutet. Malicious android applications: Risks and exploitation. http://pen-testing.sans.org/resources/papers/gpen/malicious-android-applications-risks-exploitation-120821, March 2010. Accessed: 2015-10-05.

[20] Riskiq reports malicious mobile apps in google play have spiked nearly 400 percent. https://www.riskiq.com/resources/press-releases/riskiq-reports-malicious-mobile-apps-google-play-have-spiked-nearly-400, February 2014. Accessed: 2015-10-05.

[21] Pandaboard faq. http://omappedia.org/wiki/PandaBoard_FAQ. Accessed: 2015-10-05.

[22] Reijo M. Savola, Habtamu Abie, and Markus Sihvonen. Towards metrics-driven adaptive security management in e-health iot applications. In Ilangko Balasingham, editor, *7th International Conference on Body Area Networks, BODYNETS 2012, Oslo, Norway, September 24-26, 2012*, pages 276–281. ICST / ACM, 2012.

[23] Reijo M. Savola and Habtamu Abie. Metrics-driven security objective decomposition for an e-health application with adaptive security management. In *Proceedings of the International Workshop on Adaptive Security*, ASPI '13, pages 6:1–6:8, New York, NY, USA, 2013. ACM.

[24] Stephen Smalley and Robert Craig. Security enhanced (SE) android: Bringing flexible MAC to android. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society, 2013.

[25] Android compatibility test suite. https://source.android.com/compatibility/cts/-index.html. Accessed: 2015-10-05.

[26] Casey Schaufler. The simplified mandatory access control kernel. http://schaufler-ca.com/yahoo_site_admin/assets/docs/SmackWhitePaper.257153003.pdf. Accessed: 2015-10-05.

[27] Tizen: Security/overview. https://wiki.tizen.org/wiki/Security/Overview. Accessed: 2015-10-05.

[28] Stephen Smalley. Add security hooks to binder and implement the hooks for selinux. https://android.googlesource.com/kernel/common.git/+/-a3c9991b560cf0a8dec1622fcc0edca5d0ced936. Accessed: 2015-10-05.

[29] Smack team. Smack userspace library. https://github.com/smack-team/smack. Accessed: 2015-10-05.

[30] Elena Reshetova. Patch for smack labelling support in udev. http://lists.freedesktop.org/archives/systemd-devel/2013-May/010934.html, May 2013. Accessed: 2015-10-05.

[31] Onur Aciicmez. Smack: Permissive mode support. http://marc.info/?l=linux-security-module&m=130092517905137&w=4, March 2011. Accessed: 2015-10-05.

[32] Casey Schaufler. Smack description from the linux source tree. http://schaufler-ca.com/description_from_the_linux_source_tree. Accessed: 2015-10-05.

[33] Jake Edge. Another lsm stacking approach. https://lwn.net/Articles/518345/, October 2012. Accessed: 2015-10-05.

[34] Mandeep Baines and Will Drewry. Integrity-checked block devices with device mapper. Linux Security Symposium 2011, http://selinuxproject.org/~jmorris/-lss2011_slides/LSS_11_Integrity_checked_block_devices.pdf. Accessed: 2015-10-05.

[35] Integrity measurement architecture (ima). http://sourceforge.net/p/linux-ima/wiki/Home/. Accessed: 2015-10-05.

[36] Dmitry Kasatkin. dm-integrity: integrity protection device-mapper target. http://lwn.net/Articles/533558/, January 2013. Accessed: 2015-10-05.

[37] Jake Edge. The 3.16 merge window concludes. https://lwn.net/Articles/602212/, June 2014. Accessed: 2015-10-05.

[38] Mohammad Nauman, Sohail Khan, Xinwen Zhang, and Jean-Pierre Seifert. Beyond kernel-level integrity measurement: Enabling remote attestation for the android platform. In *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, TRUST'10, pages 1–15, Berlin, Heidelberg, 2010. Springer-Verlag.

[39] Antutu benchmark. https://play.google.com/store/apps/details?id=com.antutu.-ABenchMark. Accessed: 2015-10-05.

[40] Softweg benchmark. https://play.google.com/store/apps/details?id=softweg.hw.-performance. Accessed: 2015-10-05.

[41] Stephan Heuser, Adwait Nadkarni, William Enck, and Ahmad-Reza Sadeghi. Asm: A programmable interface for extending android security. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1005–1019, San Diego, CA, 2014. USENIX Association.

[42] Jerome H. Saltzer. Protection and the control of information sharing in MULTICS. In Herbert Schorr, Alan J. Perlis, Peter Weiner, and W. Donald Frazer, editors, *Proceedings of the Fourth Symposium on Operating System Principles, SOSP 1973, Thomas J. Watson, Research Center, Yorktown Heights, New York, USA, October 15-17, 1973*, page 119. ACM, 1973.

[43] David Brumley and Dan Boneh. Remote timing attacks are practical. *Comput. Netw.*, 48(5):701–716, August 2005.

[44] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.

[45] Jake Edge. Netlabel: Cipso labeling for linux. https://lwn.net/Articles/204905/, October 2006. Accessed: 2015-10-05.

[46] M. StJohns, R. Atkinson, and G. Thomas. Common Architecture Label IPv6 Security Option (CALIPSO). RFC 5570 (Informational), July 2009.

[47] Tizen: Security:smacknetworking. https://wiki.tizen.org/wiki/Security:Smack-Networking. Accessed: 2015-10-05.

[48] Habtamu Abie. Adaptive security and trust management for autonomic message-oriented middleware. In *MASS*, pages 810–817. IEEE Computer Society, 2009.

[49] Habtamu Abie, Pål Spilling, and Bent Foyn. A distributed digital rights management model for secure information-distribution systems. *Int. J. Inf. Sec.*, 3(2):113–128, 2004.

[50] Habtamu Abie, Reijo M Savola, and Ilesh Dattani. Robust, secure, self-adaptive and resilient messaging middleware for business critical systems. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*, pages 153–160. IEEE, 2009.

[51] Sammy Haddad, Samuel Dubus, Artur Hecker, Teemu Kanstrén, Bertrand Marquet, and Reijo Savola. Operational security assurance evaluation in open infrastructures. In Frédéric Cuppens, Simon N. Foley, Bogdan Groza, and Marius Minea, editors, *CRiSIS 2011, Proceedings of the Sixth International Conference on Risks and Security of Internet and Systems, Timişoara, Romania, September 26-28, 2011*, pages 100–105. IEEE Computer Society, 2011.

[52] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[53] Philip J. Koopman. Design Theory & Methodology Conference, September 1995.

[54] David Kirkman. Requirement decomposition and traceability. *Requirements Engineering*, 3(2):107–114, June 1998.

[55] Debra S. Herrmann. *Complete Guide to Security and Privacy Metrics*. Auerbach Publications, Boca Raton, New York, 2007.

[56] Andrew Jaquith. *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley Professional, 2007.

[57] B. Bates, K.M. Goertzel, T. Winograd, and Information Assurance Technology Analysis Center (IATAC). *Measuring Cyber Security and Information Assurance: A State-of-the Art Report*. Information Assurance Technology Analysis Center, 2009.

[58] Vilhelm Verendel. Quantified security is a weak hypothesis: A critical survey of results and assumptions. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, NSPW '09, pages 37–50, New York, NY, USA, 2009. ACM.

[59] Wayne Jansen, Wayne Jansen, Patrick D. Gallagher, and Deputy Director. Directions in security metrics research, 2009.

[60] ISO/IEC. Internal Organization for Standardization and the International Electrotechnical Commission, 2009.

[61] Rafael Fedler, Christian Banse, Christoph KrauÃŸ, and Volker Fusenig. Android OS security: Risks and limitations - a practical evaluation. Technical report, Fraunhofer AISEC, May 2012.

[62] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Ahmad-Reza Sadeghi, and Bhargava Shastry. Practical and lightweight domain isolation on android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 51–62, New York, NY, USA, 2011. ACM.

[63] Sven Bugiel, Stephan Heuser, and Ahmad-Reza Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 131–146, Berkeley, CA, USA, 2013. USENIX Association.

[64] N. Asokan, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Kari Kostiainen, Elena Reshetova, and Ahmad-Reza Sadeghi. *Mobile Platform Security*, volume 4 of *Synthesis Lectures on Information Security, Privacy, and Trust*. Morgan & Claypool, December 2013.

[65] Cyanogenmod - android community operating system. http://www.cyanogenmod.org. Accessed: 2015-10-05.

[66] Nathan Willis. Cyanogenmod's incognito mode. https://lwn.net/Articles/560527/, July 2013. Accessed: 2015-10-05.

[67] Knox workspace - technical details. https://www.samsungknox.com/en/products/-knox-workspace/technical. Accessed: 2015-10-05.

[68] Michael Backes, Sven Bugiel, Sebastian Gerling, and Philipp von Styp-Rekowsky. Android security framework: Enabling generic and extensible access control on android. *CoRR*, abs/1404.1395, 2014.