Security Objectives, Controls and Metrics Development for an Android Smartphone Application

Reijo M. Savola and Markku Kylänpää
VTT Technical Research Centre of Finland
Oulu, Finland
reijo.savola (at) vtt.fi, markku.kylanpaa (at) vtt.fi

***Abstract*—Security in Android smartphone platforms deployed in public safety and security mobile networks is a remarkable challenge. We analyse the security objectives and controls for these systems based on an industrial risk analysis. The target system of the investigation is an Android platform utilized for public safety and security mobile network. We analyse how a security decision making regarding this target system can be supported by effective and efficient security metrics. In addition, we describe implementation details of security controls for authorization and integrity objectives of a demonstration of the target system.**

Keywords-Android; security objectives; security metrics; security effectiveness; risk analysis

I. INTRODUCTION

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

Security metrics, designed to give efficient input to the main questions addressed by the security decision-making during the full lifecycle, increase our understanding of the security effectiveness level of the target Android system. The metrics should be designed based on prioritized risk analysis results. Prioritized security objectives 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.

The main contribution of this study is in proposing heuristics for security objective and control description, and security metrics development, based on results of our recent risk analysis from an Android platform [1]. In addition, we discuss some implementation considerations of the resulting security controls of authorization and integrity objectives which have been demonstrated in the form of enhancements to Android.

II. BACKGROUND

A. Security Effectiveness, Objectives, Controls and Metrics

Sufficient *security effectiveness* (SE) level of the target system is the primary concern of security decision-making and consequently, security measurement. SE is the assurance that the stated *security objectives* (SOs) are met in the target system 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 [2–4]. SOs are high level statements of intent to counter identified threats and/or satisfy the organizational security policies and/or assumptions identified [5]. *Security controls* (SCs) should be developed based on SOs.

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

SE can be measured with a relatively high degree of accuracy only during long periods of actual operation of the target system, when it is exposed to real *security risk occurrence*. However, in this case, the accuracy declines due to the dynamic nature of the risks. Moreover, penetration testing is often used to obtain evidence of SE, but this kind of testing has many limitations compared to a life case.

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 high-level evidence contributing to the overall SE evidence: see Fig. 1 [6].

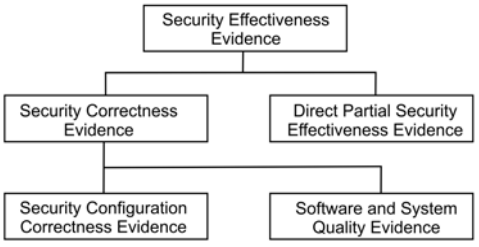

Figure 1. Example factors contributing to SE [6]

Security correctness is a key factor that contributes to the SE of the SuI (System under Investigation) due to its concreteness. However, it must be noted that it does not automatically

978-1-4799-3383-9/14/$31.00 ©2014 IEEE

imply SE, and proper RA (Risk Analysis) 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 security objectives.

B. Iterative Risk Analysis

The reference requirements used in security decision-making are characterized to be based on (i) security risk, or (ii) best practices or regulations. Security risk-driven 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. When the SE level is in focus, sufficient risk knowledge is needed.

Sufficiently detailed target-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. For example, the initial phase can be conducted when product requirements are defined, the second phase when the product is being specified, and the third phase when then 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. For the purposes of this study, a RA was performed in co-operation between security researchers from VTT Technical Research Centre of Finland, and Android experts with Elektrobit Wireless Communications Ltd. The results were reported earlier in [1]. Business aspects were focused upon in the RA.

C. Security Metrics Development by Hierarchical SO Decomposition

The hierarchical SO decomposition approach was originally described in [10]. Fig. 2 depicts a simplification of the decomposition principle [11]. Basic Measurable Components (BMCs) are leaf components of decomposition that clearly manifest a measurable property of the system [10]. The actual security metrics, Derived Measures (DMs) can be developed based on the BMCs. The high-level BMCs shown in Fig. 2 are Authentication Mechanism Integrity, Authentication Mechanism Reliability, Authentication Identity Uniqueness, Authentication Identity Structure and Authentication Identity Integrity [10]. The number of nodes in the hierarchy is much larger in practice. The BMCs of Fig. 2 can be further decomposed, taking into account the specific system characteristics better.

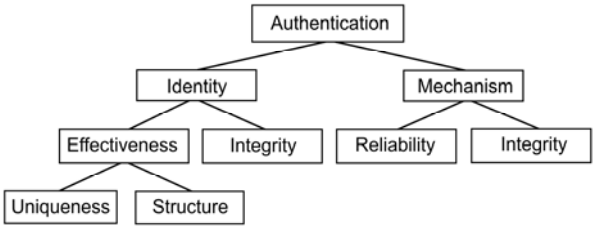

Figure 2. An example authentication decomposition based on [10]

III. SYSTEM UNDER INVESTIGATION

The target system of the investigation here is a public safety and security (PSS) mobile network system using the Android platform. The main services and characteristics of PSS mobile networks are group communication, encryption services and high availability [7]. PSS mobile networks utilize a mobile infrastructure that is very similar to cellular networks. One major difference is that they incorporate dispatcher stations for managing the communication of the user groups. To investigate advanced security controls and security measurement in the target system, a demonstrator has been developed using an Elektrobit RaptorPad Android device (see Fig. 3). Currently, the demonstrator includes advanced security controls for access control and integrity measurements.

The majority of the Android platform's security solutions originate from Linux. Memory, process, user, and access control permission management are provided by the Linux kernel, but are modified from traditional desktop usage. A major challenge with Android's modified Linux kernel is the upgrade process: In many older versions, the version of the kernel deployed is clearly out of date, and users have devices whose Android version is no longer upgraded by the device manufacturer. Moreover, 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 is a challenging situation from a security perspective, because there have been [8] and still are [9] several malicious applications in Google Play and other marketplaces.

Figure 3. Demonstrator Android device: Elektrobit RaptorPad

IV. RISK-DRIVEN SECURITY OBJECTIVES AND CONTROLS

The risk-driven approach assumes development of SOs based on the prioritized risks. 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 for a risk to be mitigated, cancelled, or accepted.

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, SOs and SCs. For example, the most critical risk, R1 (unauthorised input of falsified data) has interdependency with R7 (loss of life). Even though both are critical risks, the criticality of R1 can be seen as even more emphasised 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 challenging. Even though there are many interdependencies among risks, they should be listed in the manner shown in the tables. Otherwise, information about the prioritization is lost, potentially impeding SE in the security controls to be implemented.

Table I lists the top 10 risks out of the 26 identified ones in [1] for Case 1. The rank of each risk is shown by the number in the first column. 'S' refers to the severity of the consequences if the risk is actualized, and 'P' denotes the probability of the risk being realized. The scale for each is 0–3 from 'no risk' to 'extremely high' S or P, with increments of 0.25. 'R:' denotes 'risk arising from', and is used in connection with *attack types*, *vulnerabilities*, and *faults* that cause a risk.

Note that the RA/SO/SC/metrics process is the focus of the study. The results from RA, SOs and SC descriptions are discussed as examples.

TABLE I. TOP 10 PRIORITISED RISKS FOR CASE 1

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	1.00
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

Table II lists the heuristics for SOs and SCs of the risks mentioned in Table II. Most of the SOs address the main security dimensions according to the 'CIA model', confidentiality, integrity and availability. The text in them is shortened, emphasizing the main dimensions. In the SC column, 'T' indicates technical controls, whereas 'M' refers to management (non-technical) controls.

TABLE II. SECURITY OBJECTIVES AND CONTROLS FOR TABLE I

R#	SO	SC
1	(i) Allow only *authorized* persons to use devices and network infrastructure, (ii) ensure *integrity* in input data	(i) Sufficient authorization in devices and network infrastructure equipment (T), (ii) integrity enforcing mechanisms in data communication (T), (iii) authorization policies (M)
2	Ensure the high *availability* of the network at critical moment, 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), (iii) use and authorization policies during critical moments (M)
3	(i) Allow only *authorized* persons to use devices and network infrastructure, (ii) ensure that no procedures make unauthorized root access possible, (iii) ensure *confidentiality* of authorized root access procedures	(i) Sufficient authorization in devices and network infrastructure equipment (T), (ii) Management of correct configuration (T), (iii) proper testing of apps to be used in the PSS scenarios, (T,M) (iv) authorization policies (M)
4	(i) Allow only *authorized* persons to use devices and network infrastructure, (ii) ensure that no procedures make unauthorized root access possible, (iii) ensure *confidentiality* of authorized remote code procedures	(i) Sufficient authorization in devices and network infrastructure equipment (T), (ii) management of correct configuration (T), (iii) proper testing of apps to be used in the PSS scenarios, (T,M) (iv) authorization policies (M)
5	Ensure high *integrity* of critical security functionality	(i) Management of correct configuration (T), (ii) proper testing of critical security solutions (T,M)
6	Ensure sufficient *integrity* of devices, communication and networks, and do not allow unauthorized actions	(i) Integrity and authenticity enforcing mechanisms in devices, network infrastructure and data communication (T), (ii) authorization policies (M)
7	Ensure the high *availability* of the network and devices at life-threatening critical moment, 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) 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 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 authorization in devices and network infrastructure equipment (T), (ii) integrity enforcing mechanisms in data communication (T), (iii) authorization policies (M), (iv) correct firewall configuration (T).

As can be seen from the SOs and SCs, there are a lot of commonalities. As expected, 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 against them is a significant step, 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. SC5 and SC8, and the respective SCs, are very similar.

V. Security Metrics Heuristics

In the following, we propose heuristics for security metrics development based on the risk-driven SOs. The heuristics are categorized as (i) security controls, (ii) security effectiveness abstract models, and (iii) BMC development. The first two are discussed in the first two subsections, and the third one in subsequent subsections concentrating on core BMCs. The proposed BMCs are modified from the one proposed in [10].

A. Security Controls Categories

From the results of the SO and SC development one can find the core security control categories which should be addressed by metrics.

In Table II, the following SC categories can be identified: (i) authorization (SC1, SC2, SC3, SC4, SC9, SC10), (ii) integrity mechanisms and correct configuration (SC1, SC4, SC5, SC6, SC8, C10), (iii) security testing (SC3, SC4, SC5, SC8), (iv) intrusion detection and traffic monitoring (SC2, SC7), (v) side-channel attack protection (SC9) and (vi) firewall (SC10).

The categorization of SCs helps in developing Security Effectiveness Abstract Models (SEAMs), the next step in metrics development. Note that the actual SCs in a category contain a lot of differences. For example, access control can be designed for the end-user device, for network equipment management or for some other purpose.

B. Security Effectiveness Abstract Models

SEAM [1] is a simplified decomposition model that encompasses the core knowledge of factors contributing to the SE of the target system. For example, an SEAM can be developed using the information in Fig. 2 for the purposes of authentication metrics development. Six strategies for security measurement objective decomposition were proposed in [1], consisting of basic and integrated strategies. The *basic strategies* addressed direct partial security effectiveness, security configuration correctness, and software and system quality. *Integrated strategies* were proposed for pure security effectiveness, the security effectiveness vs efficiency trade-off and for compliance measurements, using as their reference models best practice documents and regulations.

In general, there is a need for SEAMS for all the security control categories mentioned above. As in Android there are many software-related quality concerns: the SEAM for SW and system quality should be developed in addition. Applicable vulnerability database information should be integrated to it.

Compliance with regulations is crucial for the use of a PSS mobile network system. In addition to the SEAMS for all security control categories and SW and system quality, SEAM compliance is clearly needed, although compliance concerns are not listed in the RA.

C. Authorization and Authentication

A proper authorization policy is the basis for the entire authorization mechanism. In addition, seamless co-operation with the authentication mechanism is also crucial. In the target system, there are demanding requirements for both device-level authorization and network infrastructure management authorization. Consequently, the average Authentication Effectiveness (*AUE*) – is a core metric.

At the high level, *AUE* is dependent, according to Fig. 2, on the following parameters:

$$AUE = f(AIUE, AISE, AIIE, AMRE, AMIE), \quad (1)$$

where *AIUE* is Authentication Identity Uniqueness Effectiveness, *AISE* Identity Structure Effectiveness, *AIIE* Identity Mechanism Reliability Effectiveness, *AMRE* Mechanism Reliability Effectiveness, and *AMIE* Mechanisms Integrity Effectiveness.

Average Authorization Effectiveness (AZE) is largely dependent of AUE:

$$AZE = f(AUE, ACE, ACPE), \quad (2)$$

where *AUE* is the average Authentication Effectiveness, *ACE* Access Control Effectiveness, and *ACPE* Access Control Policy Effectiveness.

D. Availability

Availability is a high-level measurement dimension, as other dimensions, such as authorization and secure communication (in the form of confidentiality and integrity) are crucial preconditions for availability. QoS performance metrics are also part of the availability decomposition, especially for the purposes of DoS or DDoS attacks detection. Availability is typically measured as the percentage of the time during which the target system is ‘up’ or, meaning essentially the time when information is available from the system. Availability metrics *AV* can be based on the following parameters:

$$AV = f(AUE, AZE, IE, CE, QE), \quad (3)$$

where *AUE* is average authentication effectiveness, *AZE* is authorization effectiveness, *IE* is integrity effectiveness, *CE* is confidentiality effectiveness, *RE* is the resilience effectiveness, and *QE* is the QoS effectiveness indicator.

VI. Enhancements of Security Controls in Demonstrator

Table I describes prioritized risks for the PSS device. Risk 1 (unauthorized input) is typically handled in the application level. All input data should be verified. Also, the second priority risk 2 (unavailability of PSS Network) is infrastructure-level denial-of-service threat. The next two risks, 3 and 4 (unauthorized root access and malicious loading of remote code), could be mitigated by enhancing access control and integrity protection. In the following, we discuss some implementation-level enhancements to mandatory access control 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.

A. Mandatory Access Control

Google has now integrated SELinux-based SEAndroid [12] 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 applications are classified to pre-defined SEAndroid security domains based on their origin and installation package signing. 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 [13].

SELinux was the first MAC framework in Linux kernel, but it is not the only one. Another Linux upstream kernel, MAC framework Smack [14], which is now also used in the Tizen operating system [15], 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.

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. File system labelling tools were modified to create Smack labels. Device file labelling is still missing in our demonstrator but could be added, utilizing ideas from [16]. Userspace modifications included modifications to the 'init' process, Dalvik VM, and Zygote application launcher to manipulate Smack labels. Toolbox commands 'ls' and 'ps' were also modified to display Smack labels. The init process is used to load Smack policy and to also set Smack labels to privileged tasks. Smack userspace library was used [17]. Smack was configured into the use in kernel configuration. Kernel changes included addition of Binder IPC Linux Security Module (LSM) hooks using a patch from SEAndroid [18] and implementing these hooks in Smack kernel code. Initrd was used to store initial Smack 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. However, we are using only classic Smack features, as some of our test systems were using a rather old kernel versions. As a starting point we tried to emulate SEAndroid security policy. However, 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 [19]. Also, SEAndroid policy emulation is not the best approach for security policy development.

Another drawback was recognized when the Smack model was used to control Android Binder-based IPC. According to Smack documentation [20], 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 'rwxa' (read, write, execute, and append). 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.

One approach could be to use both SEAndroid and Smack simultaneously. There is an unofficial patch set called LSM stacking [21] 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 visi-

ble only after releases, and Google does not share development plans in public. A large porting effort may be needed after releases.

B. Integrity Protection

Kernel-based integrity protection frameworks can be used to protect Android systems against unauthorized system software modifications (e.g. utilizing offline attacks). Android release 4.4 (KitKat) includes an experimental block-based integrity scheme called dm-verity [22]. There are other alternatives such as the file-based Integrity Measurement Architecture Extended Verification Module IMA/EVM [23]. 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 [24] 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, shared library or 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 [25]. Another problem with IMA is that it only measures native applications and not Java-based Dalvik applications. Nauman et al. [26] have developed a framework that allows measurement of Java code running in Dalvik VM. However, Google is now replacing Dalvik VM with a new virtual machine called ART.

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.

VII. Related Work

Haddad et al. [27] introduced an abstract security model called Assurance Profile (AP) for metrics definition. Its focus is on security assurance objectives, and risk-driven security objectives are not addressed. The decomposition part of the security metrics development approach discussed here is similar in general to the Goal Question Metrics (GQM) of Basili et al. [28] refining specification of software measurements. It should be noted that the GQM definitions do not include heuristics to define security metrics. The challenges of requirement decomposition in general have been discussed by Koopman [29] and Kirkman [30]. As problems, they mention ‘gaming’ promoted by too great a focus on goals, excessive subsystem decomposition, insufficient decomposition, unattributed requirements, excessive hierarchy 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 [31–34]. These metrics can be used at the detailed level, when developing DMs from BMCs. Standards have achieved only limited success in advancing security measurement, because they are rigid and created for certification, and carrying out these processes requires significant amounts of time and money [35]. The most widely used of these efforts is the CC (ISO/IEC 15408) Standard [36] which focuses primarily on documentation rather than the actual security effectiveness of the operational system. The ISO/IEC 27004 standard [37] addresses measurement, reporting and improving the effectiveness of Information Security Management Systems (ISMS). However, this standard does not support technical systems well.

Android OS security risks were studied in general by Fedler et al. They summarize in [38] that most successful attacks affecting Android can be attributed to negligent user behavior. However, they admit that attacks on Android devices are becoming 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 to Android. For example, TrustDroid uses Tomoyo [39] and FlaskDroid is using SELinux [40]. 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 [41][42]. Samsung has extended Android SEAndroid concept in their Knox product [43], providing more isolated containers targeting bring-your-own-device (BYOD) enterprise customers. Other manufacturers have so far kept Google’s security approach, although some of them have replaced Google’s services with their own equivalents. Examples of these are Amazon and Nokia/Microsoft who provide Android devices without Google services.

VIII. Conclusions and Future Work

Risk-driven security engineering and metrics development assumes the development of security objectives and controls based on the prioritized risks. We analysed security objectives for an Android platform utilized for a public safety and security mobile network based on iterative industrial risk analysis results. There were many interdependencies between the original risks. Furthermore, security objectives and controls show a different pattern of interdependencies. The original risk analysis results should be preserved in further actions to offer appropriate emphasis.

The core security controls for the target system are authentication and authorization, integrity, and confidentiality controls. In particular, access control plays an important role in the target system, where there are health-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 important for the system.

Furthermore, we proposed heuristics for security metrics development, based on the risk-driven security objectives. The heuristics are categorized by security controls, security effectiveness abstract models, and base measure development.

We have also discussed 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. 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, managing the metrics by a visualization tool, and gathering validation information from realistic use cases of the demonstration system. We also plan to enhance the integrity measurement part to support remote attestation that would be an important use case for PSS devices.

ACKNOWLEDGMENT

The work presented here has been carried out in two research projects: the IoT Program (2012-2014), launched by the Finnish Strategic Centre for Science, Technology and Innovation TIVIT Plc., and SASER-Siegfried Celtic-Plus project (2012-2015).

REFERENCES

[1] R. Savola et al., “Toward risk-driven security Measurement for Android smartphone platforms,” Proc. Information Security South Africa (ISSA) 2013, Johannesburg, August 14-16, 2013, 8 p.

[2] R. Savola, “Security metrics taxonomization model for software-intensive systems,” Journal of Information Processing Systems, Vol. 5, No. 4, Dec. 2009, pp. 197–206.

[3] W. Jansen, “Directions in security metrics research,” U.S. National Institute of Standards and Technology, NISTIR 7564, Apr. 2009, 21 p.

[4] ITSEC. Information Technology Security Evaluation Criteria (ITSEC), Version 1.2, Commission for the European Communities, 1991.

[5] ISO/IEC 15408-1:2005. Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and General Model, International Organization for Standardization and the International Electrotechnical Commission, 2005.

[6] R. Savola, “Strategies for security measurement objective decomposition,” ISSA 2013, 15–17 August 2013, Johannesburg, South Africa, 8 p.

[7] M. Peltola, “Evolution of public safety and security mobile networks,” licentiate thesis, Aalto University School of Electrical Engineering, Espoo, Finland, 2011.

[8] J. Boutet, Malicious Android Applications: Risks and Exploitation, SANS Institute, 22 March 2010.

[9] D. Gray, TETRA: Advocate’s Handbook, from Paper Promise to Reality, 2003.

[10] R. Savola and H. Abie, “Development of measurable security for a distributed messaging system,” International Journal on Advances in Security, Vol. 2, No. 4, 2009, pp. 358–380 (published in March 2010).

[11] C. Wang and W.A. Wulf, “Towards a framework for security measurement”, Proceedings of 20th National Information Systems Security Conference, 1997, pp. 522–533.

[12] S. Smally and R. Craig, “Security Enhanced (SE) Android: Bringing flexible MAC to Android,” 20th Annual Network and Distributed System Security Symposium (NDss 2013), Feb. 2013.

[13] Google, “Android Compatibility Test Suite”, http://source.android.com /compatibility/cts-intro.html [Accessed July 8, 2014]

[14] C. Schautler, “The Simplified Mandatory Access Control Kernel, Smack white paper,” schaufler-ca.com/yahoo_site_admin/assets/docs/ SmackWhitePaper.257153003.pdf [Accessed July 8, 2014].

[15] Tizen, “Security/Overview”, https://wiki.tizen.org/wiki/Security/ Overview [Accessed July 8, 2014]

[16] E. Reshetova, ”Patch for Smack labelling support in udev,” systemd-devel mailing list, http://lists.freedesktop.org/archives/systemd-devel/2013/May/010934.html [Accessed July 8, 2014]

[17] Smack team, “Smack userspace library”, GitHub source code repository, https://github.com/smack-team/smack [Accessed July 8, 2014]

[18] S. Smalley, “Add security hooks to binder and implement the hooks for SELinux”, Android kernel patch, Nov 5, 2012, https://android.googlesource.com/kernel/common.git/+/a3c9991b560cf0a8dec1622fcc0edca5d0ced936 [Accessed July 8, 2014]

[19] O. Aciicmez, “SMACK: Permissive mode support”, Linux Security Module mailing list, http://marc.info/?t=130092518400001&r=1&w=4 [Accessed July 8, 2014]

[20] C. Schaufler, “Smack description from the Linux source tree”, http://scahufler-ca.com/description_from_the_linux_source_tree [Accessed July 8, 2014]

[21] J. Edge, ”Another LSM stacking approach”, Linux Weekly News, Oct 3, 2012, http://lwn.net/Articles/518345/ [Accessed July 8, 2014]

[22] M. Baines, W. 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 July 8, 2014]

[23] D. Kasatkin and M. Zohar, “Integrity Measurement Arhitecture”. SourceForge. http://sourceforge.net/p/linux-ima/wiki/Home/ [Accessed July 8, 2014]

[24] D. Kasatkin, ”dm-integrity: integrity protection device-mapper target”, Jan 22, 2013, http://lwn.net/Articles/533558/ [Accessed July 8, 2014]

[25] J. Edge, “The 3.16 merge window concludes”, Linux Weekly News, Jun 18, 2014, http://lwn.net/Articles/602212/ [Accessed July 8, 2014]

[26] M Nauman, S Khan, X Zhang, and JP Seifert, “Beyond kernel-level integrity measurement: enabling remote attestation for the Android platform” - Trust and Trustworthy Computing, Springer Berlin Heidelberg , 2010, pp. 1–15.

[27] S. Haddad, S. Dubus, A. Hecker, T. Kanstrén, B. Marquet and R. Savola, “Operational security assurance evaluation in open infrastructures,” Proc. CRiSIS 2010, pp. 100–105.

[28] V. Basili, G. Caldiera, and H.D. Rombach, ”The Goal Question Metric approach,” J. Marciniak (Ed.), Enclyclopedia of Software Engineering, Wiley, 1994.

[29] P. Koopman, “A Taxonomy of Decomposition Strategies Based on Structures, Behaviors, and Goals,” Design Theory & Methodology ’95, 1995.

[30] D. Kirkman, “Requirement Decomposition and Traceability,” Requirements Engineering, Vol. 3, No. 2, 1998, pp. 107–114.

[31] D. S. Herrmann, Complete Guide to Security and Privacy Metrics – Measuring Regulatory Compliance, Operational Resilience and ROI, Auerbach Publications, 2007, 824 p.

[32] A. Jaquith, Security Metrics: replacing Fear, Uncertainty and Doubt, Addison-Wesley, 2007.

[33] N. Bartol, B. Bates, K.M. Goertzel, and T. Winograd, Measuring Cyber Security and Information Assurance: A State-of-the-art Report, Information Assurance Technology Analysis Center, May 2009.

[34] V. Verendel, “Quantified security is a weak hypothesis: a critical survey of results and assumptions,” New Security Paradigms Workshop, Oxford, U.K., 2009, pp. 37–50.

[35] W. Jansen, “Directions in security metrics research,” U.S. National Institute of Standards and Technology, NISTIR 7564, Apr. 2009, 21 p.

[36] ISO/IEC 15408-1:2005. Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and General Model,

International Organization for Standardization and the International Electrotechnical Commission.

[37] ISO/IEC 27004:2009. Information Technology – Security Techniques – Information Security Management – Measurement. International Organization for Standardization and the International Electrotechnical Commission.

[38] R. Fedler, C. Banse, C. Krauss and V. Fusenig, “Android OS security: risks and limitations – a practical evaluation,” Fraunhofer AISEC Technical Report, May 2012.

[39] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, ”Practical and lightweight domain isolation on Android” SPSM'11, Chicago, Illinois, USA, October 2011.

[40] S. Bugiel, S. Heuser, and A.-R. Sadeghi, ” Flexible and fine-grained Mandatory Access Control on Android for diverse security and privacy policies” 22nd USENIX Security Symposium (USENIX Security '13), USENIX. 2013.

[41] Cyanogenmod – Android Community Operating System, http://www.cyanogenmod.org [Accessed July 8, 2014]

[42] N. Willis, ”CyanogenMod's incognito mode,” Linux Weekly News, 24 July 2013, https://lwn.net/Articles/560527/ [Accessed July 8, 2014]

[43] Samsung, ”Samsung KNOX - Technical Details”, https://www.samsungknox.com/overview/technical-details [Accessed July 8, 2014]