A SECURE RUNNING ENVIRONMENT FOR MULTIPLE PLATFORMS

Reijo M. Savola
VTT Technical Research Centre of Finland, Oulu, Finland

Keywords: Secure operating environment, Security assurance, Security metrics.

Abstract: At present, the security critical operations of terminal devices are often being executed in the operating system, which may include security vulnerabilities due to implementation faults, for example. These vulnerabilities leave the system open to data leaks and attacks from viruses or other harmful programs. The European €-Confidential ITEA research project is developing device-independent, next-generation security solutions for software platforms. Critical operations are executed on a simple platform where the security operations are isolated in a separate module, which can be physically located in a terminal device or in a separate device such as a memory stick. This paper introduces a Secure Running Environment (SRE), in which the core security management of the platform is located. This contains sensible parts for the security of the operating system, middleware and applications. The security platform alone does not guarantee an adequate level of security. Security is a challenging and interdisciplinary field that demands holistic understanding, and validation of the realization of the security objectives and the solutions advancing them. The most common methods for security assurance are security analysis, security testing and security monitoring.

1 INTRODUCTION

Information security systems today encounter various kinds of information security threats. The increasing complexity of software-intensive and telecommunication products, together with pressure from information security and privacy legislation, is increasing the need for adequately tested and managed information security solutions in telecommunications and software-intensive systems and networks. The systems are constantly under the threat as data theft, terrorism and vandalism increase. Security assurance is increasingly important in order to ensure the correct operation of increasingly complex equipment, software and services. Equipment, services and networks are more vulnerable to security attacks, as the software of terminal devices has become more complex and their interfaces and connectivity have increased. Even simple vulnerabilities may remain undetected because of complexity. Most of the information security threats in the traditional world of PCs also concern terminal devices.

The €-Confidential (2009) ITEA-Eureka project proposes a new, promising data security platform and the use of adaptable information security assurance methods throughout the entire product and service life cycles. The most significant aims of information security – confidentiality, integrity, availability and non-repudiation – are by nature very challenging and interdisciplinary fields. Security is a challenging and interdisciplinary field that demands holistic understanding and validation of the realisation of the security objectives and the technical solutions advancing them.

The main contribution of this study is introduction of a multi-platform secure running environment supported with adaptive security assurance methods. Section 2 presents the operating environment, Section 3 overviews case studies and Section 4 introduces relevant security assurance methods. Finally, Section 5 discusses future work and Section 6 gives conclusions.

2 A TRUSTED AND SECURE OPERATING ENVIRONMENT

In the €-Confidential project, a novel secure and reliable platform has been developed for the needs

M. Savola R. (2009).
A SECURE RUNNING ENVIRONMENT FOR MULTIPLE PLATFORMS.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 129-134
DOI: 10.5220/0002244201290134

of end users. The platform enables the execution of critical operations in different types of terminal devices (mobile phones, laptops, etc.). Examples of critical operations are authentication, encryption, the processing of personal data, secure connections to peripheral devices and secure banking connections. The trustworthiness and security requirements should be taken into account as early as the initial stages of software design and continue throughout the life of the software and product. The architecture used should enable the development of reliable solutions, and be sustainable throughout the life cycle of the system.

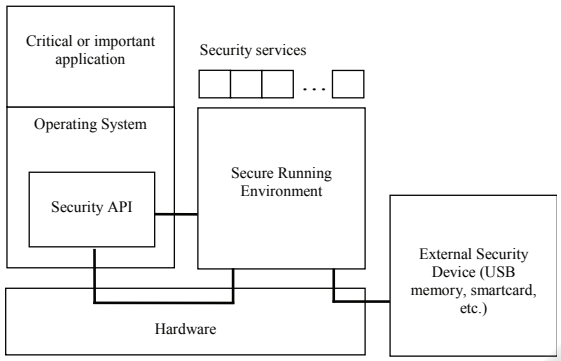

Figure 1: The Secure Running Environment (SRE) includes sensitive parts of an operating system, middleware and applications (Savola *et al*., 2008).

The project has developed a reliable and secure operating environment, Secure Running Environment (SRE), in which the core security management of the platform is located, and which is isolated from other components in the device. SRE contains sensitive parts for the security of the operating system, middleware and applications. SRE and the surrounding processing environment can exchange data through a secured communication channel. The role of SRE is to be a proxy between a user/consumer and a vendor and/or a financial institution. Fig. 1 shows the role of SRE on a general level.

SRE includes several external APIs (Application Programming Interface), see Fig. 2. In an environment open to attack, SRE-API includes a TrustZone® API used for communicating and service management, a management API for SRE software management, a secure programming interface for the storage area and a standardised encryption algorythm programming interface Cryptoki-API. In a secured environment, the SRE-API includes a Secure Service Developer Interface (SSDI) and a Small Terminal Interoperability Platform (STIP).

2.1 Uncoupling of Applications and Execution Environment

A key feature of the €-Confidential platform is to use a virtual machine to implement security services. The use of a virtual machine has several advantages from portability, interoperability, certification and security point of view. First and foremost, applications and execution environment are uncoupled. Furthermore, the presence of a virtual machine allows for a close control of the sensitive resources of the platform by the security services. It is easy to filter access to files, communication points and peripherals when applications cannot access these resources directly in memory and are obliged to go through well controlled instructions and APIs. It must be noted too that the byte code implemented by the virtual machine can have specific properties that may help security property checks using security assurance techniques such as static code analysis.

2.2 Native and Interpreted Service Frameworks

Native and interpreted service frameworks allow security services to access SRE functions or platform security resources in a platform independent way. The frameworks include functionalities for cryptography (authentication, integrity, confidentiality and non-repudiation), key storing, persistent storage, trusted time base, external world access, user interaction and customization.

2.3 Overall €-Confidential Architecture

There are three main layers in the €-Confidential architecture: at the top Service Layer, Separation Layer under it and Hardware Layer at the bottom. In addition, at the very top are several special entities, *environments*. The Service Layer provides secure services and and trust to the environments. The Separation Layer ensures the separation and strong isolation between the environments. It provides memory isolation, safe computing time sharing, device assignment and an inter-process communication mechanism.
Finally, the Hardware Layer comprises of peripheral devices and is dictated by the microprocessor architecture.

There can be four kinds of environments above the Service Layer: Legacy Environment, Attestable Environment, Secure Environment and Trusted Environment. A legacy environment is a virtual

execution environment, isolated from other environments. An attestable environment is used to run an attested system, be it a specific application (e.g. banking or DRM player) or a hosting place for security services. A secure environment is an environment which has been proved as correct, i.e. which has received a certification.

The Secure Running Environment (SRE) is part of the Service Layer and acts as the host for all sensitive operations required by the Service Layer.

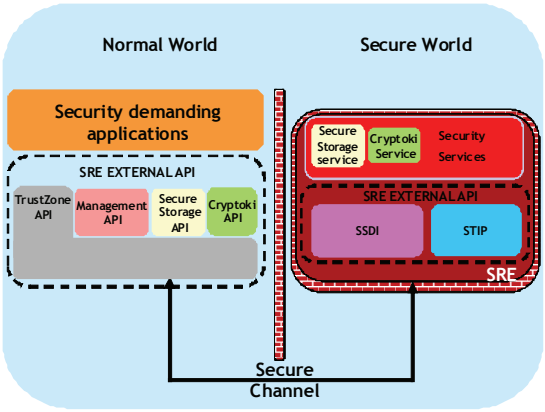

Figure 2: External Application Programming Interfaces of the SRE.

3 CASE STUDIES

The €-Confidential architecture was implemented in a server-customer environment. Implementation made use of case studies, through which the correct operation of the architecture could be tested and assured. A *Voice over IP* (VoIP) with a Help Desk function was created for the case studies.

The case studies were integrated into the same environment using SAML modules. SAML (Security Assertion Markup Language) is XML (eXtensible Markup Language) standard, which is intended for authentication and authorisation data exchange between information security management areas. In this case, these are the fields of an identity service and application service provider. SAML modules were used for the building of an SSO service.

3.1 Voice over Internet Protocol

In the VoIP scenario, the end user clicks on the 'Call Help Desk' link in the authenticated session. The user's session is assured by using Liberty Alliance-*type identity federation and Single Sign-On (SSO). In the session, use of Help Desk functions* and an SIP (*Session Initiation Protocol*) server connection or SIP application are permitted. The SIP application is connected through a VPN link to the SIP server, and the customer is authenticated into the VoIP service using certificates that are in a smart card connected by USB. Once the SIP control signal has been formed, the user can call the Help Desk.

3.2 Electronic Banking

In the project, a highly secure electronic banking application was created, in which the user is able perform some of the more common banking operations (like transferring money and requesting a balance) and use the Help Desk VoIP (Voice-over-Internet-Protocol) service. The security solutions include central bank server and user identification, data encryption and maintenance of connection integrity. The application can be used on a normal PC or wireless terminal device.

3.3 Electronic University

In addition to the banking application, an electronic application for universities was created, where the student can use his/her user account and the Help Desk service through the VoIP service. The user account includes calendar information and performance data and it is possible to use it to carry out functions such as registration and choice of materials. The security solutions included centralised authentication of the university server and student software, data encryption and maintenance of connection integrity. Authentication was done using electronic certificates, a smart card and a Spanish electronic ID card.

4 SECURITY REQUIREMENTS AND ASSURANCE

A secure and trusted environment solution is not a sufficient security solution by itself. The security and dependability requirements of a system and applications should be well analysed and adequate security assurance methods should be used.

4.1 Security Requirements

Security is clearly a system-level problem. Consequently, one cannot accurately determine the information security requirements outside the context and environment of the system. Building

security requirements is often a process of making trade-off decisions between high security (S), high usability (U) and low cost (C). The adequate level of security typically lies in the center region of the "S-U-C pyramid" (Savola, 2008). Various stakeholders are needed in making the tradeoff decisions, such as managers, developers, security experts and end users.

A security requirement is a manifestation of a high-level organizational security policy in the detailed requirements of a specific system (Devanbu and Stubblebine, 2000). According to Firesmith (2004), the most current software requirement specifications are either (i) totally silent regarding security, (ii) merely specify vague security goals, or (iii) specify commonly used security mechanisms (e.g., encryption and firewalls) as architectural constraints. In the first case security is not taken into account in an adequately early phase of design. In the second case vague security goals (like "the application shall be secure") are not testable requirements. The third case may unnecessarily tie architectural decisions too early, resulting in an inappropriate security mechanism.

The goal of defining security requirements for a system is to map the results of risk and threat analyses to practical security requirement statements that manage (cancel, mitigate or maintain) the security risks of the system under investigation. The requirements guide the whole process of security evidence collection. For example, security metrics can be developed based on requirements: if we want to measure the security behavior of an entity in the system, we can compare it with the explicit security requirements, which act as a "measuring rod".

All applicable dimensions (or quality attributes) of security should be addressed in the security requirements definition. See, e.g., (Avižienis, 2004) for a presentation of quality attribute taxonomy. Well-known general dimensions include confidentiality, integrity, availability, non-repudiation and authenticity. Quality attributes like usability, robustness, interoperability, etc, are important requirements too.

One cannot easily define a security requirement list that could be used for different kinds of systems. However, the functional part of Common Criteria (ISO/IEC 15408, 2004) includes general-level requirement lists and can be used as guidance. The actual requirements and role of the security dimensions heavily depend on the system itself and its context and use scenarios. The requirements should embody adequate system design and security countermeasure design information too.

The definition of security requirements is an iterative process. The sequence of security requirement iterations depends on use cases, the available architectural and other technical information and the maturity of the system model. As mentioned before, security requirements cannot be considered only as non-functional requirements; they may potentially create new functional requirements during the iteration of the architectural and more detailed technical design.

4.2 Security Assurance

If new vulnerabilities already noticed in the systems or found in testing can be eliminated sufficiently early in the product development phase, considerable savings can be made in comparison with security repairs carried out at a later stage in the product life cycle. Security assurance methods such as security analysis, security testing and security monitoring help in the development of products and services and in the later stages of their life cycles. In recent years, the understanding and tools of security assurance have developed in leaps and bounds, enabling the functional testing and monitoring of data security to be part of the normal product process. Comprehensive security analysis guides testing and monitoring activity. Security analysis may include the investigation of threats, the specification of security requirements, security modelling, the investigation of vulnerability and the assessment of security levels using security metrics.

A practical and secure product development life-cycle model includes the identification of security objectives in the initial stage, the preparation of responsibilities and plans and security analysis. At the stage of specifying requirements, security standards are gathered together and security assurance methods, such as security testing, are planned. At the architecture and system planning stage, threat and vulnerability models and drawn up, and security architecture that meets the requirements is designed. At the implementation and testing stage, coding and testing recommendations advancing reliability and safety are observed, and adequately certified components are introduced. In the testing stage, the security testing plan is implemented along with other selected security assurance methods. In the final stage of product development, a final security review is performed along with the secure configuration of the system, and the secure and trusted distribution of the system is taken care of. In the maintenance stage, suitable security assurance measures are carried out, such as monitoring and

reviewing. Further actions include security training and promoting security awareness, risk management, security measurement, vulnerability management and clear division of roles and responsibilities.

4.3 Security Assurance Approaches

The most common approaches to security assurance are:

- Risk, threat and vulnerability analysis. Risk analysis is usually part of risk management. Threat identification is also closely related activity to risk analysis. Vulnerability recognition is a crucial part of practical technical security analysis. The traditional risk management puts less emphasis on vulnerabilities and more on threat analysis.
- Design-level information security analysis. Design-level analysis pays particular attention to software architecture, platforms and security requirements. Analysis techniques include modelling (e.g. patterns), logical analysis and interface and constraint analysis.
- Interdependency analysis. Interdependency analysis identifies the key interdependencies of software components, physical infrastructures, networks, standards and technologies that are possibly security-threatening.
- Source code analysis. Source code analysis can be done in many different ways: using special purpose-built tools, code review procedures, manual checking of source code and coding procedures supporting checking.
- Security testing can be divided into white box and black box testing. In the former, the source code is available whereas it is not in the latter. White box methods include static and direct code analyses, property-based testing, source code fault injection, fault progress analysis, fault tree analysis and source code dynamic analysis. Black box methods include fuzzers, software penetration testing, binary code security analysis, binary files fault injection, code scanning and vulnerability scanning.
- Security auditing. Security inspections (auditing) are regular "health check-ups", in which it is checked that the level of system security corresponds to requirements, and suggestions are made to correct possible deficiencies.
- Anomaly detection. Anomalies are abnormal, possibly hostile behaviour of the system. Detection can be based on the modelling of the expected behaviour of the system and on comparison of the actual behaviour and the model.

5 FUTURE WORK

The Secure Running Environment developed in the €-Confidential project can be further developed into a standardized concept, to be integrated to various potential platforms. New case studies can include e.g. mobile communication, ubiquitous computing and critical infrastructure applications.

A practical security assurance framework requires a lot of future development. In the following we list some goals for future work. The core activity in security assurance and evaluation is the definition of the security requirements, based on risk, threat and vulnerability analyses, as well as technical and architectural information. The current state-of-the-art practice is limited to a too high abstraction level.

6 CONCLUSIONS

The €-Confidential platform and its Secure Running Environment implement a novel minimal Trusted Computing Base (TCB), ensuring strong isolation between different functional environments and providing trusted services to them. Critical operations are executed on a simple platform where the security operations are isolated in a separate module, which can be physically located in a terminal device or in a separate device such as a memory stick.

A secure and trusted environment solution is not a sufficient security solution by itself. The security and dependability requirements of a system and applications should be well analysed and adequate security assurance methods should be used.

REFERENCES

€-Confidential Eureka ITEA Project Website www.itea-econfidential.org, 2009

Avižienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. In *IEEE Tr. on Dependable and Secure Computing*, Vol. 1, No. 1 Jan/Mar 2004, pp. 11-33.

Devanbu, P. T. and Stubblebine, S. Security and Software Engineering: A Roadmap. In *22nd Int. Conf. of Software Engineering (ICSE)*, Limerick, Ireland, 2000.

Firesmith, D. Specifying Reusable Security Requirements. In *Journal of Object Technology*, Vol. 3, No. 1, Jan/Feb 2004, pp. 61-75.

ISO/IEC 15408. *Common Criteria for Information Technology Security Evaluation*, Version 2.2, 2004.

Savola, R. A Framework for Security Modeling and Measurement. In *IFIP TC 11.1 Annual Working Conference on Information Security Management*, Richmond, Virginia, 2008.

Savola, R., Röning, J., Sederholm, C., Heinonen, J., Uusitalo, I., Wieser, C., Mantere, M., Karppinen, K., Karinsalo, A., Karjalainen, K. Tietoturvaa kaikille raudoille (In Finnish, Information Security for all Platforms), *Prosessori Magazine*, No. 12/2008, pp. 30-31.