TOWARDS SECURITY EVALUATION

BASED ON EVIDENCE COLLECTION

Reijo Savola

VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90570 Oulu, Finland
phone: +358 40 569 6380, fax: +358 20 722 2320
Reijo.Savola@vtt.fi

ABSTRACT

Information security evaluation of software-intensive systems typically relies heavily on the experience of the security professionals. Obviously, automated approaches are needed in this field. Unfortunately, there is no practical approach to carrying out security evaluation in a systematic way. Here we introduce a general-level holistic framework for security evaluation based on security behaviour modelling and security evidence collection, and discuss its applicability to the design of security evaluation experimentation setups in real-world systems.

KEY WORDS

Information security evaluation, security metrics, security modelling, security testing

TOWARDS SECURITY EVALUATION

BASED ON EVIDENCE COLLECTION

1 INTRODUCTION

Products and services, and the technical infrastructures that enable them are showing a strong trend towards convergence and networking. At the same time, industrial companies and other organizations are creating very complex value nets to design and manufacture products and to maintain them. These trends, together with pressure from information security and privacy legislation, are increasing the need for adequately tested and managed information security solutions in software-intensive systems and networks. The lack of appropriate information security solutions might have serious consequences for business and the stakeholders.

Security evaluation, testing and assessment techniques are needed to be able find adequate solutions. Seeking *evidence* of the actual information security level or performance of systems still remains an ambiguous and undeveloped field. In order to make progress in the field there is a need to focus on the development of better experimental techniques, better security metrics and models with practical predictive power [4].

Security evidence can be used for both quantitative and qualitative analysis methods. The evidence is more useful when it is meaningful for most of the system's lifecycle:

- *During research and development***,** security evidence helps researchers to develop more secure solutions and find design vulnerabilities. Research-oriented security evidence can be constructed using analytical models that take account of factors contributing to security and the cross-relationships of components. Research-oriented metrics can concentrate on the critical parts, especially the technical challenges.
- *During system implementation***,** security evidence can be used to find design and implementation vulnerabilities as a part of security engineering. These are also based on analytical models. If security metrics are part of a security engineering process, they are more valuable.
- *During the system maintenance phase***,** security evidence can be used for preservation of the achieved security level during possible updates, integration or modifications, and to find implementation vulnerabilities. From the point of view of the security engineering process, a technical system can be constantly in the system maintenance phase. In addition to preservation of the security level, this level can be improved using feedback obtained from the application of security evidence information.

The main contribution of this study is to introduce a holistic approach to security evaluation based on evidence collection and to discuss the evidence collection process in practice. The rest of this paper is organized as follows. Section 2 discusses security metrics and their relationships in general, Section 3 presents our theoretical security modelling and evaluation framework, Section 4 analyses how evidence collection can be done in practice with the help of the framework, and, finally, Section 5 discusses future work and Section 6 gives conclusions.

2 BACKGROUND

The wide majority of the available security metrics approaches offering evidence information have been developed for evaluating security policies and the maturity of security engineering processes. The most widely used of these maturity models is the Systems Security Engineering Capability Maturity Model SSE-CMM (ISO/IEC 21827) [8]. Other well-known models are Trusted Computer Security Evaluation Criteria (TCSEC, The Orange Book) (TCSEC 1985) [16] and Common Criteria (CC) [7]. In connection with policy and process metrics, it is extremely important to evaluate the security functionality of products at the technical level, without forgetting their life cycle management. The goal of the whole process of seeking of security evidence should be targeted at understanding the information security threats and vulnerabilities of the product and its usage environment holistically.

Jonsson [9] sorts the methods of security measurement into the following techniques: risk analysis, certification and measures of the intrusion process:

- *Risk analysis* is an estimation of the probability of specific intrusions and their consequences and costs: it can be thought of as a trade-off to the corresponding costs for protection,
- *Certification* is the classification of the system in classes based on design characteristics and security mechanisms. “The ‘better’ the design is, the more secure the system.”
- *Measures of the intrusion process* means statistical measurement of a system based on the effort it takes to make an intrusion. “The harder it is to make an intrusion, the more secure the system.”

In addition to these methods, it is justifiable to consider *auditing* and *security evaluation* as measurement techniques for information security. Most technical security analysis is currently performed using penetrate-and-patch or “tiger team” tactics. The security level is evaluated by attempting to break into a system under evaluation by exploiting previously known vulnerabilities. If a break-in attempt is successful, the vulnerability is patched. Penetrate-and-patch tactics have been used by special security testing professionals whose methods and tools have not been made public knowledge. There are several problems with penetrate-and-patch: it requires experienced professionals, the actual testing is carried out too late, and the patches are often ignored and even sometimes introduce new vulnerabilities. Most of the technical testing metrics are meant for the unit or source code level. Various methods for system security evaluation and assessment have been proposed in the literature, see, e.g., [2, 12, 13, 18]. These frameworks are conceptual and help in understanding the problem area. However, these frameworks do not offer aggregated means for practical security evaluation or the testing process.

3 FRAMEWORK FOR SEEKING EVIDENCE OF SECURITY

In the following we introduce our holistic framework for model-based information security evaluation or testing of software-intensive systems. This collection of constructs and abstractions forms the basis of our approach to seeking evidence of security in a system. Please note that the framework could be expressed in a formal way using various types of representations, such as Labelled Transition Systems (LTSs). However, in this paper we discuss the implications of the framework for practical security testing and evaluation rather than intending to formalize the framework.

3.1 Role of Threat Analysis

The most important task in the whole process of security evaluation is to identify the security risks and threats, taking enough assumptions of the attackers' capabilities into account. A subtask in threat analysis is to identify valuable *assets* that may be subject to security risks. An asset is something in the context of the system that is to be protected. A threat description can be represented by, e.g., *threat* / *asset* combinations. A holistic and cross-disciplinary threat picture of the system controls the development of security solutions. Threats that are possible during the whole life cycle of the system under evaluation must be considered.

It must be noted that the collection of security threats to a system is not static. Security algorithms and other solutions are cracked and new vulnerabilities are found every now and then. Even complete platforms or communication protocol structures can be compromised. As a consequence, a system's threat landscape is constantly changing, possibly reflecting different kinds of trends. A *weak* signal is a factor for change that is hardly perceptible in the present but which will constitute a strong trend in the future. Some weak signals can represent on-going or anticipated changes in the threat landscape. The actual change in time can happen in small steps or in one leap. In the former case the trend could be exposed, if the weak signals presenting the steps could be detected [11].

Example 1 We denote the original set of identified threats in a system by T, consisting of the threat factors T_0, T_1, T_2,...T_n. Later, the effect of discipline, D_x, is introduced into the system. This effect manifests itself as a weak signal type of threat, τ_x, which can or cannot be identified. In the former case T is updated to $T := T \cup \tau_x$. In the latter case the effect of D_x remains a hidden threat represented by the undetected weak signal τ_x. □

3.2 Role of Security Requirements

The goal of defining security requirements for a system is to map the results of risk and threat analysis to practical security requirement statements that manage (cancel, mitigate or maintain) the security risks of the system under investigation. Security requirements are constraints on functional requirements intended to reduce vulnerabilities. Security mechanisms are then developed to fulfil the requirements. Haley *et al.* [6] present an interesting method for deriving security requirements from threat descriptions. They derive the security requirements using an iterative process where each iteration recomposes the threat descriptions with the functional requirements. Iterations are required because identifying and eliminating vulnerabilities will often create new vulnerabilities.

The security requirements play a crucial role in the security evaluation. 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 security behaviour 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 [1] 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. In fact, an unusable security construct can even turn out to be a security threat.

The safety community has developed a standard approach to solving the problem of requirements relevance, and the similarity between safety and security implies that it would be well worth considering if something similar could be done for security [3]. For example, Security Importance Levels (SILs) could be used for categorizing non-security requirements in terms of their security relevance and Security Evidence Assurance Levels (SEALs) could be used to enforce the additional measures needed to develop the more security-critical parts of systems.

It must be noted that one cannot easily define a general-level security requirement list that could be used for different kinds of systems. The actual requirements and role of the security dimensions heavily depends on the system itself, and its context and use scenarios.

3.3 Modelling Entities and their Cross-Relationships

It is obvious that a model of the security behaviour of a system is needed in order to be able to evaluate security systematically. To make a decision about whether a system is secure, we need evidence that (i) each software or hardware component and subcomponent and (ii) the composition formed from them, taking account of cross-relationships, are secure. Essentially, the process of security evaluation takes use scenarios and the context of the system into account. In addition to this structural classification of entities, it is important to find the behavioural entities in the system. In order to help investigate the security behaviour of a system, we define *security action, atomic security action and security behaviour*:

Definition 1 (*security action*) A security action, a_r, is a behavioural entity of a system that has some effect, either incremental or decremental, on the security defined by a certain security requirement, r, of a system. □

Definition 2 (*atomic security action*) An atomic security action, α, is a security action that cannot be split into other security actions. It is the lowest level of observable security behaviour. □

Definition 3 (*security behaviour of a system*) The security behaviour, A, of a system consists of a composition of atomic security actions of all the security requirements of the system that take their cross-relationships into account. □

The security behaviour of the system, expressed using suitable modelling language, is the basis model of the system under security evaluation. For example, a pattern language could be used to describe the security actions and security behaviour.

Security actions can represent one or several dimensions or quality attributes of security. We define the dimensions of a security action in the following:

Definition 4 (*dimensions of a security action*) A security action, a_r,, having an effect on security requirement r, has an impact, $i(a_r,u)$, and a probability, $p(a_r,v)$. □

Definition 5 (*impact of a security action*) The impact $i(a_r,u)$ of security action a_r is the estimate of its impact in scale [-1,1] with security requirement r. If the impact increases security, it is positive; u is the uncertainty of the impact estimate, between [0,1], where 1 presents complete certainty. □

Definition 6 (*probability of a security action*) The probability $p(a_r,v)$ of security action a_r is the estimated probability of the security action to be realized with uncertainty, v. □

It is important to notice that impact analysis of security actions is within the focus of our approach. After all, we are interested in the impact of a system's security behaviour on the whole – i.e. the overall impact.

Definition of the actual security actions in the system under investigation is a challenging task. In practice, this task may turn out to be impossible due to the amount of functionality and use scenarios in practical systems. Real-world implementations are far too complex for this kind of analysis. There is a need for automated and easily applicable and standardized technical methods for software implementation to ensure and measure security, e.g. standard secure memory management support and component-level life cycle management support.

Modelling the security behaviour is an iterative process. Voas [17] states that we do not know *a priori* whether the security of a system composed of two components, A and B, can be determined merely from knowledge of the security of A and B. Rather, the security of the composite is based on more than just the security of the individual components – it hinges on the cross-relationships. Both the atomic behaviour and the cross-relationships have to be known and analysed in an iterative way.

The security behaviour of a system could be modelled using tree representations, such as Attack Trees [14], evaluation criteria, such as Common Criteria [7] and several formal approaches and semi-formal approaches, such as UML and its security extension UMLSec [10]. Perhaps the most interesting method is to develop security patterns [15]. A security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution. In practice, a chosen set of security patterns could guide the process of defining security requirements. Security behaviour with an adequate set of security actions could be associated with these patterns. The key elements of security patterns include the following:

- *Name:* a label representing the structure,
- *Context:* general conditions,
- *Problem:* a statement that defines the problem that will be solved by the security pattern, and
- *Solution:* the solution to the problem.

3.4 Evidence Information

Security evidence is gathered from various sources as input to the decision process of security evaluation. The evidence collection should be arranged in a way that supports evaluation of the security behaviour and security actions. We classify the types of security evidence information into three categories:

- *Measured evidence:* The process of gathering measured or assessed information uses security metrics as its basis. Table 1 lists some examples of measured security evidence. Measured evidence can be collected during security testing or in a security audit based on pre-defined metrics.
- *Reputation evidence:* Reputation of software or hardware constructs, or their origin, is an important class of evidence. A software company in charge of implementing a product might have some confidential knowledge of the security of different software components. Table 2 lists some examples of reputation evidence. Reputation evidence can be collected from experience of R&D departments and be based on general-level knowledge.
- *Tacit evidence:* In addition to the measured and reputation evidence, there might be some "silent" or "weak" signals of security behaviour. The subjectivity level of tacit evidence might be higher than in the case of measured and reputation evidence. Collection of tacit evidence is typically an ad hoc process. Senior security experts and "tiger teams" play an important role in this kind of evidence.

Table 1: Examples of measured evidence

Dimension	**Metric types**
Confidentiality	Use of compartmentalization in memory use
Confidentiality	Encryption strength
Integrity	Result of one-way hash function
Integrity	Robustness of data synchronization algorithm
Availability	Validation result of access control rules
Usability	Amount of user interaction needed

The objectivity level of the evidence varies a lot. In many cases even the measurements are arranged in a highly subjective manner. Typically, no single measured value is able to capture the security value of a system. Thus several pieces of security evidence have to be combined.

Table 2: Examples of reputation evidence

Metric types
Reputation of practices of subcontractor
Reputation of implementation results of subcontractor
Reputation of a software version
Reputation of a software component provider
Reputation of a standard used in the implementation
Reputation of an integrator

3.5 Trust Assumptions

A *trust assumption* is a decision to trust the given properties of some domain and go no further in the analysis [5]. Trust assumptions set the boundaries for the need for evidence. Trust assumptions can be made based on reputation evidence: if we trust a software version fully, there is no need to investigate it at a more detailed level.

Trust assumptions can make the security evaluation process feasible by taking a certain risk in assuming that the object left out of more detailed investigation is trusted.

3.6 Decision Process

The most final phase of security evaluation is the decision process. The overall goal of the decision process is to make an assessment and form conclusions on the information security level or performance of the system under investigation. The decision process can be split into sub-decisions based on the security action model.

The decision process can be carried out in the following way:

1. For each security requirement and security action composition, seek evidence and estimate the probability and impact of that action, taking cross-relationships and trust assumptions into account.
2. Estimate the overall impact of the gathered evidence on each security requirement
3. Make a decision whether the security of the system with regard to the requirements is at a sufficient level.

In a high abstraction level, the overall impact of all security actions on a security requirement can be defined as follows:

Definition 7 (*overall impact*) The overall impact of all security actions on a security requirement is

$$I = \sum_{t=0}^{T} w_t \cdot p_t \cdot i_t$$

where I is the overall impact, T the number of all security actions of a security requirement, and w_t is a weighting factor, p_t a probability and i_t the impact of security action t on the requirement under investigation. The weighting factor depends on time and context.

4 PRACTICAL CONSIDERATIONS

In the previous section we presented an approach to security modelling and evaluation. Unfortunately, in practice, a thorough modelling of security behaviour is only possible in a few ideal cases. Typically, today's software-intensive products are very complex, and their functionality is not well documented and often has unknown dependencies. Development of an ambiguous security behaviour model at an atomic security action level is a very challenging and time-consuming task.

The practical needs for security evaluation are often limited too. This results in a situation in which we should be able to try to find the security actions that are most critical and most typical. To reach the desired security level it is not important to try and measure every part and component that affects security. Instead, we need enough evidence to make trade-off decisions.

We propose the following process to carry out practical-level security evaluation:

- *Risk and threat analysis.* Carry out risk and threat analysis of the system and its use environment, if not carried out before. In real-world engineering risk and threat analyses are not carried out adequately. Consequently, the set of security requirements might not be sufficient.
- *Define security requirements* in such a way that they can be compared with the security actions of the system. Based on the threat analysis, define the security requirements for the system, if not yet defined. These are lacking in many practical systems.
- *Prioritize security and other requirements.* The most critical and most often needed security requirements should be paid the most attention.
- *Model the security behaviour.* Based on the prioritized security requirements, identify the functionality of the system that forms the security actions and their dependencies in a priority order.
- *Gather evidence* from measured, reputation and tacit security information.
- *Estimate the probabilities and impacts of security actions* based on the evidence.
- *Aggregate the results from the probability and impact estimation* to form a clear picture of whether or not the system fulfils the security requirements.

5 DISCUSSION AND FUTURE WORK

A practical security evaluation framework based on the ideas discussed in this paper requires a lot of future development. In the following we list some goals for future work.

A suitable language needs to be developed to formalize and express security actions and their cross-dependencies, as well as security requirements. Both the system security behaviour and requirements need to be expressed in such a way that it is possible to compare them. A language able to express behavioural patterns is a good candidate for this purpose. Security patterns are currently under development in the pattern community. Security patterns augmented with semantics representing security properties could offer a feasible means to model the security requirements and security behaviour of systems. Possibly, the use of fuzzy logic might be connected to that kind of language. A mechanism to describe the interactions and cross-dependencies of security actions is needed.

A knowledge base of typical security constructs should be established to offer pattern information on their security behaviour. The security actions of a system can be expressed using patterns. Typical constructs include encryption elements, firewalls, proxies, compartmentalization, inter-process communication, access control mechanisms and authentication mechanisms. The information needs to be collected experimentally to enable development of the knowledge base.

Security evaluation or testing can be done in practice if this kind of knowledge base support could be used for security behaviour modelling and suitable security requirement documentation of the system is available. Furthermore, the process of evidence collection from different sources, and aggregation of it, should be developed using experimental information from real-world systems.

6 CONCLUSIONS

We have discussed the problem of information security evaluation in the context of software-intensive systems. There are no systematic means of carrying out security evaluation. In this paper we have presented a conceptual holistic framework for security modelling and evaluation with some practical considerations. The framework is based on evidence collection and security requirement-centred impact analysis.

This is not a rigorous solution and future work needs to be done on developing a suitable language for expressing security requirements and security behaviour in an unambiguous way. A collection of security patterns would be very helpful in modelling the security behaviour when carrying out security testing or experimentation.

In practical security evaluation, the requirements should be prioritized and the system modelled only to the extent needed to conform to the trust assumptions. Full modelling of practical systems is not feasible without automated approaches, which might be very challenging to develop.

REFERENCES

1. Avizienis A., Laprie J.-C., Randell B. and Landwehr C. (2004); Basic Concepts and Taxonomy of Dependable and Secure Computing. In: IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, January/March 2004, pp. 11-33.
2. Brocklehurst S., Littlewood B., Olovsson T. and Jonsson E. (1994) On Measurement on Operational Security. In: IEEE AES Systems Magazine, Oct. 1994. pp. 7-15.
3. Firesmith D. G. (2005) Analyzing the Security Significance of System Requirements. In: Symposium on Requirements Engineering for Information Security (SREIS), August 25, 2005, Paris.
4. Greenwald M., Gunter C., Knutsson B., Seedrov A., Smith J. and Zdancewic S (2003): Computer Security is not a Science (but it should be). In: Large-Scale Network Security Workshop, Landsdowne, VA, March 13-14, 2003.
5. Haley C. B., Laney R. C., Moffett J. D. and Nuseibeh B. (2003) Using Trust Assumptions in Security Requirements Engineering. In: 2nd International iTrust Workshop on Trust Management in Dynamic Open Systems, 15-17 September, 2003, Imperial College, London, UK.
6. Haley C. B., Laney R. C. and Nuseibeh B. (2004) Deriving Security Requirements from Crosscutting Threat Descriptions. In: AOSD 04, March 2004, Lancaster, UK.
7. ISO/IEC 15408 (2004) Common Criteria for Information Technology Security Evaluation, Version 2.2
8. ISO/IEC 21827 (2002) Information Technology – Systems Security Engineering – Capability Maturity Model (SSE-CMM).
9. Jonsson, E. (2003) Dependability and Security Modelling and Metrics, Lecture Slides, Chalmers University of Technology, Sweden.

10. Jürjens J. (2002) UMLSec: Extending UML for Secure Systems Development. In: UML 2002 – The Unified Modeling Language, Vol. 2460 of LNCS, pp. 412-425, Springer, 2002.

11. Kajava J. and Savola R. (2005) Weak Signals in Information Security Management. In: Proceedings of the International Conference on Computational Intelligence and Security (CIS) 2005, Part II, Xi'an, China, December 15-19, 2005. Springer, pp. 508-517

12. McDermid J.A. and Shi, Q. (1992) A Formal Approach for Security Evaluation. In: Proceedings of the 7th Annual Conference on Computer Assurance, Systems Integrity, Software Safety, Process Security (COMPASS), pp. 47-55.

13. Nicol D., Sanders W. H., Trivedi K. S. (2004) Model-Based Evaluation: From Dependability to Security. In: IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, January/March 2004, pp. 48-65.

14. Schneier B. (1999) Attack Trees. In: Doctor Dobb's Journal, December 1999, pp. 21-29.

15. Schumacher M. and Roedig U. (2001) Security Engineering with Patterns. In: Pattern Languages of Programs 2001, September 11-15, Monticello, Illinois.

16. Trusted Computer System Evaluation Criteria, "Orange Book", (1985), U.S. Department of Defense Standard, DoD 5200.28-std

17. Voas, J. (2001) Why is it so Hard to Predict Software System Trustworthiness from Sofware Component Trustworthiness? In: Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems.

18. Voas J., Ghosh A., McGraw G., Charron F. and Miller K. (1996) Defining an Adaptive Software Security Metric from a Dynamic Software Failure Tolerance Measure. In: Proceedings of the 11th Annual Conference on Computer Assurance, Systems Integrity, Software Safety, Process Security (COMPASS).