Information Security Evaluation based on Requirements, Metrics and Evidence Information

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

Abstract

Information security assurance and 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. We introduce an iterative process for security evaluation based on security requirements, metrics and evidence collection, and discuss its applicability to the design of security evaluation experimentation set-ups in real-world systems. In this approach, security requirements can be used to define the basis for security measurements. Furthermore, other kinds of security metrics and other security evidence can be used to security decision-making.

Keywords: information security evaluation, security metrics, security modeling, security testing, security evidence

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. 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.

Security evidence can be used both for quantitative and qualitative analysis methods. Evidence is more useful when they are meaningful for most of the systems lifecycle:

- During research and development, security evidence helps researchers to develop more secure solutions and to 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 proposes an iterative information security evaluation process, Section 3 discusses the role of information security requirements in this process, Section 4 analyses security metrics and Section 5 discusses the role of evidence and analyses how evidence collection can be done, and, finally, Section 6 discusses practical considerations and Section 7 gives conclusions.

### 2 Iterative Information Security Evaluation Process

We propose the following process to carry out practical-level security evaluation. In this process it is essential that the gathering of evidence information is used for probability and impact estimation.

- **Risk and threat analysis.** Carry out risk and threat analyses of the system and its use environment if not carried out before. These are lacking in many practical systems.

- **Define and prioritize 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. The most critical security requirements should be paid the most attention. Remember that the weakest links of the system are critical too. The definition of security requirements is a highly iterative process.

- **Model the security behavior.** 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. Use suitable evidence collection tools like vulnerability identification and assessment tools.
- **Estimate the probabilities and impacts** of security actions based on the evidence. Aggregate the results to form a clear picture of whether or not the system fulfils the security requirements.

An important task in the process of security evaluation is to identify the security risks and threats, taking enough assumptions of the attackers’ capabilities into account. In information security a risk can be defined as a function of three variables: the probability that there is a threat, the probability that there are vulnerabilities and the potential impact. Note that threat is a similar concept to hazard in safety.

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 in a system is not static. Security algorithms and other solutions are cracked and new vulnerabilities are found every now and then.

### 3 Role of Information Security Requirements

A security requirement is a manifestation of a high-level organizational security policy in the detailed requirements of a specific system (Devanbu & 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.

One of the most widely known standards on security requirements is the Common Criteria (CC) (ISO/IEC, 2004), which is used as the basis for evaluation of the security properties of information technology systems. Another relevant reference is the ISO/IEC 17799 standard (ISO/IEC, 2005), which includes a section on systems development and maintenance.

Information 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), see Figure 1. The adequate level of security typically lies in the center region of the “S-U-C pyramid” of the figure. Various stakeholders are needed in making the tradeoff decisions, such as managers, developers, security experts and end users.
In the following we discuss the practice of defining security requirements in the related areas of requirements engineering, software engineering and safety engineering.

A lot of work done in the requirements engineering community is based on interdisciplinary activities, where an important practice is the treatment of non-functional and functional requirements. Unfortunately, satisfactory approaches to capturing and analyzing non-functional requirements have yet to mature (Nuseibeh & Easterbrook, 2000).

A functional requirement defines something the system must do, capturing the nature of the interaction between the component and its environment. A functional requirement must be testable, which means it must be possible to demonstrate that the requirement has been met by a test case resulting in pass or fail (IEEE, 1990).

A non-functional requirement is a software requirement that, rather than describing what the software will do, describes how the software will do it. Non-functional requirements restrict the manner in which the system should accomplish its function. Non-functional requirements tend to be general and concern the whole system, not just some parts of it. Security requirements are often conceived solely as non-functional requirements along with such aspects as performance and reliability within the requirements engineering community (Chung et al., 1994). From the security engineering viewpoint this is a too simplified way of thinking. Information security cannot be represented only by non-functional requirements since security goals often motivate new functionality, such as monitoring, intrusion detection and access control, which, in turn, need functional requirements. In addition, a distinctive feature of security requirements is that they are asset-driven – their goal is to protect the set of identified assets.

The established practices of software requirements engineering can be split into requirements development and requirements management (Wiegers, 2003). Requirements development can be further subdivided into elicitation, analysis, specification and validation (Abran & Moore, 2001). These sub-disciplines encompass all the activities involved with gathering, evaluating and documenting the requirements for a software or software-intensive product. Requirements management entails establishing and maintaining an agreement with the customer on the requirements for the software project (Paulk et al., 1995). Security requirements have not been profoundly addressed within the
software engineering community: they are still regarded as being in a side role in most of the software requirements engineering codes of practice. Ideally, the characteristics of excellent software requirements include (Davis, 1993 and IEEE, 1998):

- **Completeness**: each requirement must fully describe the functionality to be delivered.
- **Correctness**: each requirement must accurately describe the functionality to be built.
- **Feasibility**: it must be possible to implement each requirement within the known capabilities and limitations of the system and its operating environment.
- **Necessity**: each requirement should document a capability that is really needed.
- **Prioritization**: each functional requirement should be assigned an implementation priority.
- **Unambiguity**: all readers of a requirement statement should arrive at a single, consistent interpretation of it.
- **Verifiability**: the product should properly implement each requirement.

Safety and security are closely related concepts. The similarity between safety and security implies that it would be well worth considering whether some established safety requirements engineering practices can be carried out for security. For example, the safety community has developed a standard approach to solving the problem of requirements relevance. It is more important to properly implement those requirements with major safety ramifications than with less safety significance. Prioritization can be used in security requirements engineering too.

In safety engineering the following types of safety analyses are typically carried out (Firesmith, 2005): asset, harm, incident, hazard and risk analysis. Once the safety risks have been identified and categorized, safety engineers use the results of these analyses to assign corresponding Safety Integrity Levels (SILs) to individual requirements or collections of related requirements, and Safety Evidence Assurance Levels (SEALs) to the associated architectural, design and code components that implement these requirements. Similar “level” measures can be developed for security. The reference level should be ruled by appropriate security requirements.

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., Avizienis et al. (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.
It is obvious that a model of the security behavior 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. Structural aspects, e.g. security architecture, influence the security behavior. The models can represent both positive and negative security behavior. Most non-security requirements stipulate that the system must do something, while security requirements are frequently focused on ensuring something does not happen (negative behavior requirement). For example, the chain of attacker strategy usually has a negative effect, functionality in key generation and distribution either a positive or negative effect, or functionality in using publicly accessible memory areas either a positive or negative effect. The lack of understanding of negative requirements is at the root of many security problems.

Security requirements and the systems they address typically include different kinds of dependencies. Dependencies represent situations where a security behavior is affected by the occurrence of another.

4 Role of Security Metrics

It is a widely accepted management principle that an activity cannot be managed well if it cannot be measured. In overall, metrics provide four fundamental benefits – to characterize, to evaluate, to predict and to improve. Security metrics and measurements can be used for decision support, especially in assessment and prediction. Furthermore, we believe that if common metrics approaches between different security disciplines (e.g. information, software and network security) can be found, this will advance our holistic understanding and capabilities in security management and engineering.

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) (ISO/IEC, 2002). Other well-known models are Trusted Computer Security Evaluation Criteria (TCSEC, The Orange Book) (TCSEC, 1985) and Common Criteria (CC) (ISO/IEC, 2004). 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 information security threats and vulnerabilities of the product and its usage environment holistically.

Methods of measurement can be divided into (Katzke, 2001):

- Direct testing (like functional, red team, and penetration testing). Measure of the intrusion process is a 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
- Evaluation (for example with Common Criteria).
- Assessment (like risk/vulnerability assessment) is estimation of the probability of specific intrusions and their consequences and costs, and it can be thought of as a trade-off to the corresponding costs for protection.
- Accreditation.
- Training and education.
- Observation of system performance, such as intrusion detection.
- 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.”

*Measurements* provide a one-time view of specific measurable parameters; *metrics* are produced by taking measurements over time and comparing two or more measurements with predefined baselines, thus providing a means of interpretation for the collected data (Jelen, 2000). A security metrics model according to Katzke (2001) consists of the object being measured, the security objectives (“measuring rod”), and the method of measurement.

According to Jelen (2000) a good metric is Specific, Measurable, Attainable, Repeatable and Time-dependent (“SMART”). Payne [Payne] reminds that truly useful security metrics indicate the degree to which security goals, such as data confidentiality are being met.

Security objectives used as a measuring rod can include for example (Katzke, 2001) (Lindqvist et al., 1998):

- Security requirements, such as specifications, standards, control objectives and Common Criteria Protection Profiles. Standards like TCSEC (Trusted Computer System Evaluation Criteria) specify criteria against which security evaluations can be made. These are useful for vendors and manufacturers, as the functionality of the system and the development process has been the target.
- Best practices.
- Due diligence, i.e. security management based on experience.
- Maturity models like SSE-CMM.
- Baseline security documents benefit producers to set a minimum set of requirements for security features that an information technology (IT) should possess.
- Security policies are useful to managers, operators and users of the IT system, who need to follow certain rules in order to minimize potential threats.

Assurance-related metrics (Ferraiolo, 1998) collected in order to track the success of security engineering efforts can include:

- System/product vulnerabilities.
- Deviations from policy.
- Cost deviation.
• Schedule deviation.
• Number of policy changes.
• Number of security requirement changes.
• Use of security features.
• Failed security tests.
• Success of system accreditation effort.
• Success of product evaluation effort.

Methods of measurement can be divided into (Katzke, 2001):
• Direct testing (like functional, red team, and penetration testing). Measure of the intrusion process is a 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”.
• Evaluation (for example with Common Criteria).
• Assessment (like risk/vulnerability assessment) is estimation of the probability of specific intrusions and their consequences and costs, and it can be thought of as a trade-off to the corresponding costs for protection.
• Accreditation.
• Training and education.
• Observation of system performance, such as intrusion detection.
• 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.”

Security metrics properties can be investigated based on the following classification:
• **Quantitative vs. qualitative metrics.** In general, quantitative metrics are more desirable than qualitative ones. It is challenging to find quantitative metrics that depict information security phenomena.
• **Objectivity vs. subjectivity of metrics.** The goal of security metrics development is to find metrics that are as objective as possible. In reality, as security contains a lot of human behavioral aspects, many metrics tend to be highly subjective.
• **Direct vs. indirect metrics.** According to ISO9126 standard (2000), a direct measure is a measure of an attribute that does not depend upon a measure of any other attribute. On the other hand, an indirect measure is derived from measures of one or more other attributes.
• **Static vs. dynamic metrics.** Dynamic metrics involve time, and static ones do not. Time perspective is important in security metrics as the information security threat pictures are constantly changing.
• **Absolute vs. relative metrics.** Dynamic metrics involve time, and static ones do not. Time perspective is important in security metrics as the information security threat pictures are constantly changing.
Technical security metrics can be used to describe the security performance of technical objects. This includes algorithms, specifications, architectures and alternative designs, products, and as-implemented systems at different stages of the system lifecycle.

Using product-oriented technical security metrics, both design vulnerabilities and implementation vulnerabilities can be sought. Design vulnerabilities can result from an insecure design, whereas implementation vulnerabilities are connected to poor implementation of a product. Thus the former term typically refers to lower technology maturity, see Figure 2.

![Figure 2. Design and implementation vulnerabilities](Image)

Traditionally, industrial-strength technical security metrics solutions mostly rely on the “penetrate and patch” or “tiger team penetration” approach. The technical systems are security tested by applying common security attacks and determining if such attacks are successful. If an attack results in an intrusion, an appropriate patch for the software is developed and applied to the system. Tiger teams often use dissimilar approaches, and their capabilities and experience vary a great deal. An essential problem is that the manual process of tiger team penetration testing easily results in statistically non-reproducible data.

Technical security metrics can be applied in many ways, including:

- **Goal establishment**: to establish goals and measure how well the object achieves those goals;
- **Prediction**: to predict security performance before implementation or in an implemented system, to predict possible intrusion using an Intrusion Prevention System (IPS);
- **Comparison**: to compare the security performance of objects;
- **Monitoring**: to monitor or scan the security performance of an object (e.g. Intrusion Detection System IDS); and
- **Fault analysis**: in the case of fault injection methods, metrics enable analysis.

The maturity of different kinds of security metrics available varies a lot. The most well-known metrics include the metrics used for describing the performance of cryptographic
algorithms. There are various ways of describing cryptographic algorithm metrics, e.g. (Jorstad & Landgrave, 1997):

- **Key length metric**: the security of a symmetric cryptosystem is a function of the length of the key. However, adding an extra bit does not always exactly double the effort required to break public key algorithms;
- **Attack steps metric**: attack steps is defined as the number of steps required to perform “the best known attack”;
- **Attack time metric**: attack time is defined as the time required to perform the fastest known attack;
- **Rounds metric**: rounds are important to the strength of some ciphers;
- **Algorithm strength metric**: Jorstad and Landgrave use algorithm strength as the name of a scale developed to express the overall measurement of a cryptographic algorithm’s strength.

Generally, the most unexplored and most critical field in security metrics is human user behavior. An important consideration from the human user point of view is user acceptance, or, from a reverse perspective, user resistance to the systems with which they must interact. User resistance manifests itself in various ways, including improper use of the security mechanisms (Schultz et al., 2001). In general, systems with a poor usability design tend to evoke a greater degree of user resistance (Al-Ghatani & King, 1999). Sophisticated usability metrics are non-existent for ad hoc network application scenarios. We refer the reader to general standards like ISO/IEC 9126-4 (ISO/IEC, 2000). Performance issues have a strong influence on the usability of mobile ad hoc networks. Other human factors include the level of security awareness, and resistance to social engineering. Social engineering means taking advantage of human fallibility. Due to the lack of comprehensive research results regarding human factors, we cannot develop sophisticated metrics for them. For the time being, metrics such as usability metrics and performance metrics form the baseline for metrics representing human factors.

### 5 Role of Evidence Information

The role of evidence information in information security evaluation is crucial. 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 security behavior 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.

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 1. Examples of measured evidence.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Metric types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>Use of compartmentalization in memory use</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Encryption strength</td>
</tr>
<tr>
<td>Integrity</td>
<td>Result of one-way hash function</td>
</tr>
<tr>
<td>Integrity</td>
<td>Robustness of data synchronization algorithm</td>
</tr>
<tr>
<td>Availability</td>
<td>Validation result of access control rules</td>
</tr>
<tr>
<td>Usability</td>
<td>Amount of user interaction needed</td>
</tr>
</tbody>
</table>

### Table 2. Examples of reputation evidence.

<table>
<thead>
<tr>
<th>Metric types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reputation of practices of subcontractor</td>
</tr>
<tr>
<td>Reputation of implementation results of subcontractor</td>
</tr>
<tr>
<td>Reputation of a software version</td>
</tr>
<tr>
<td>Reputation of a software component provider</td>
</tr>
<tr>
<td>Reputation of a standard used in the implementation</td>
</tr>
<tr>
<td>Reputation of an integrator</td>
</tr>
</tbody>
</table>

A trust assumption is a decision to trust the given properties of some domain and to go no further in the analysis (Haley *et al.*, 2003). Trust assumptions set the boundaries for the need for evidence. Trust assumptions can be made e.g. based on reputation evidence: if we trust a software version fully, there is no need to investigate it at more detailed level. Trust assumptions can make the security evaluation process feasible by taking a certain risk to assume that the object left out of more detailed investigation is trusted.

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.

6 Practical Considerations

Unfortunately, in practice, a thorough modeling of security behavior is only possible in a few ideal cases. Today’s telecommunications and software-intensive products are very complex, their functionality is not well documented and they often have unknown dependencies. Development of an unambiguous security behavior model at a detailed level is a very challenging (if not impossible) and time-consuming task.

To reach the desired security strength, 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. One should remember that the security of a system is as good as its weakest link. This approach does not give us the freedom of dropping investigation of the weakest links as they belong to the most critical actions.

7 Conclusions

Information security demands are growing due to the higher interconnection of networks and systems between individuals and organizations. 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 process for information security evaluation emphasizing the role of security requirements, security metrics and security evidence information.

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

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


