Development of Measurable Security for a Distributed Messaging System

Reijo M. Savola
VTT Technical Research Centre of Finland
Oulu, Finland
E-mail: Reijo.Savola@vtt.fi

Habtamu Abie
Norwegian Computing Center
Oslo, Norway
E-mail: Habtamu.Abie@nr.no

Abstract—**Systematically developed security metrics make it possible to gather sufficient and credible security evidence for runtime adaptive security management and off-line security engineering and management. This study introduces and analyzes security metrics and parameter dependencies for one particular distributed messaging system. The focus is on the effectiveness and correctness of security-enforcing mechanisms. The security metrics development approach that the study utilizes is risk-driven, requirement-centric, and integrated with the development of Quality-of-Service metrics. In this approach, the security requirements are expressed in terms of lower-level measurable components by applying a decomposition approach. Security metrics are then developed based on the leaf components of the decomposition. The paper also analyzes the benefits and shortcomings of the metrics development approach and introduces a trust, confidence and trustworthiness calculation model for basic measurable components of the decomposition.**

Keywords-security metrics; security indicators; security strength; security requirements; messaging systems

I. Introduction

In order to obtain sufficient and credible evidence of the security performance of a system, service or product, a systematic approach to measuring security is required. Systematic definition of security metrics is a young field that still lacks widely accepted approaches, mainly because the current practice of security is still a highly diverse field.

This study's primary contribution is that it analyzes and defines an initial collection of security metrics and parameter dependencies for the security-enforcing mechanisms of one particular system that was used as an example, using the development method introduced in earlier work [1]. The paper also analyzes the benefits and shortcomings of the development method used in the study, and introduces a framework for calculating trust, confidence and trustworthiness of security metrics. This study advances the state of the art in security metrics in practical and concrete measurement methods, in measurable components, and in semi-formal models of security measurement and metrics. The scope of the study did not include formal modeling and validation of the defined metrics.

At a high level, the objectives measured by security metrics can be classified into three groups: security correctness, effectiveness and efficiency [2]. The discussions on metrics in this paper concentrate on the effectiveness and correctness of security-enforcing mechanisms, although it also discusses efficiency. The Security Metrics Objectives Segments (SMOS) model for the taxonomization of security metrics [2] classifies the main viewpoints of the metrics of the System under Investigation (SuI) into three categories: (*i*) security-enforcing mechanisms, (*ii*) the security quality of the system, and (*iii*) secure system lifecycle, project and business management. This study focuses on the first category: security-enforcing mechanisms. It should be noted that, from the point of view of the security metrics completeness for the target system, metrics are also required for the other two categories. The goal of this study was to provide extensive identification and high-level definition of metrics for security-enforcing mechanisms, while also being selective in the details thereof.

The study investigated security metrics, and how they were developed in an example system called GEMOM (Genetic Message Oriented Secure Middleware) [3]. GEMOM has been developed in the GEMOM EU FP7 ICT project, which focuses on security measurability, adaptive security, and the resilience of complex, distributed information systems. Security solutions with varying strength levels are required in resilient and distributed business-critical systems such as GEMOM so that they can manage security in an adaptive way according to the needs of varying situations. In adaptive security management, security metrics provide the means with which score different solutions during the system's operation. For instance, different authentication and authorization mechanisms can be utilized based on metrics. In addition, metrics are used off-line during Research and Development (R&D) and when the system configuration is changed.

This paper is organized as follows. Section II provides an introduction to security metrics. Section III presents the security metrics development process that was originally introduced in the earlier work of the authors of this paper, and analyzes its benefits and challenges. Section IV briefly introduces the GEMOM system and its monitoring approach; then Section V discusses GEMOM security threats and security requirements. Section VI identifies Basic Measurable Components (BMCs) for the effectiveness and correctness of security-enforcing mechanisms in GEMOM and proposes an initial collection of metrics and parameter dependencies. Section VII presents some observations from the study and discusses the feasibility and potential research directions of security metrics. Section VIII discusses related work and Section IX offers concluding remarks and poses some questions for future research.

II. Security Metrics, Security Indicators and Security Strength

It is often claimed that an activity cannot be managed well if it cannot be measured. System developers, managers, and security assurance personnel, as well as automated security monitoring approaches, require sufficient and credible evidence that the SuI implements the intended security level or performance. An improvement in the value of a measurement result makes it more likely that the related objective or sub-objective will be met. The term *metrics*, as used in the context of Information Technology (IT), is misleading because it implies that traditional concepts in metrology, as used in physics and other areas of science and technology, apply equally to IT [4]. There are unknown multi-disciplinary dependencies in IT, as well as doubts and often subjective judgments about technical maturity due to the novelty of applications and other technical solutions. Terms such as *security indicators* or *security strength* might be more appropriate in the case of security related objectives. The term *security strength* has traditionally been used among cryptographers and has only recently been used in reference to more general security concepts. This study uses the term *security metrics*, while recognizing the imperfections of the term.

A. *Metrics and Measurements*

Measurement is the process by which numbers or symbols are assigned to attributes of real world entities in such a way that describes them according to clearly defined rules [5]. In general, measurements provide single-point-in-time views of specific, discrete factors, while *metrics* are derived by comparing two or more measurements taken over time with a predetermined baseline [6].

B. *Use of Metrics*

Security metrics can be used for decision support, particularly in assessment, monitoring, and prediction. Security measurement targets can include a technical system, service, or product, or an organization, its processes, and resources [7]. Some of the ways in which security metrics can be used include [8]:

- Risk management activities for mitigating security risks,
- Comparison of security-enforcing mechanisms or solutions,
- Obtaining information about the security posture of an organization, process, or product,
- Security assurance (analysis, testing, monitoring) of a product, organization, or process,
- Security testing (functional, red team and penetration testing) of a system,
- Certification and evaluation of a product or organization,
- Adaptive security monitoring and management during system operation, and
- Intrusion detection and prevention in a system.

The intended use and target audience influences the security metrics requirements. Complex metrics structures with various metrics and sub-metrics can be used in automatic calculations and decision-making. However, if the goal is to develop security metrics for a human audience, such as a company's senior management in a company, it is important to visualize the result and the final metrics should be clearly understandable.

C. *Dimensions to be Measured*

Information security, as a target in itself, cannot be satisfactorily measured because it is such an abstract concept and has many multi-disciplinary dependencies. Therefore, security objectives should be investigated in greater detail. The security dimensions that the metrics should address depend largely on the application domain and environment.

The most commonly recognized dimensions of information security are Confidentiality, Integrity and Availability (CIA) [9], often referred to as the *CIA model*. Confidentiality objectives ensure that information is accessible only to those authorized to have access. Integrity encompasses safeguards of several aspects of accuracy and completeness of information. The Availability dimension can be defined by the objectives to ensure that authorized users have access to the information and associated assets they require within a reasonable timeframe [2]. The CIA model has some limitations, such as the fact that authenticity and non-repudiation of critical business transactions do not fit naturally in the model. Moreover, authorization depends on authentication. A more concise collection of security objectives includes various 'lower-level' dimensions like confidentiality, integrity, availability, authentication, authorization, and non-repudiation. This more accurately emphasizes the objectives of security-enforcing mechanisms [2]. The International Telecommunication Union (ITU) [10] has defined a larger set of security dimension: access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy. Several other factors affect the security of information systems, such as accountability, auditing, controllability, correctness, identification, recovery, reliability, robustness, safety, dependability, supervision, and trustworthiness, as well as functionality [11][12][13]. Security, Trust, Dependability, and Privacy (STDP) are often grouped together when defining security-relevant objectives for technical systems and services. It is important to note, however, that these terms are overlapping and, in some cases, even contradict each other [14]. Avižienis *et al.* [15] presented a detailed taxonomy of security and dependability quality attributes that can be used in the selection of adequate dimensions to be investigated [2].

III. Security Metrics Development Approach

The authors' earlier work [16] proposed an iterative process for security metrics development, which this paper enhances and clarifies. The process aims to develop a balanced and detailed security metrics collection for the SuI and the related measurement architecture. Measurement architecture is the operational structure for measurement and

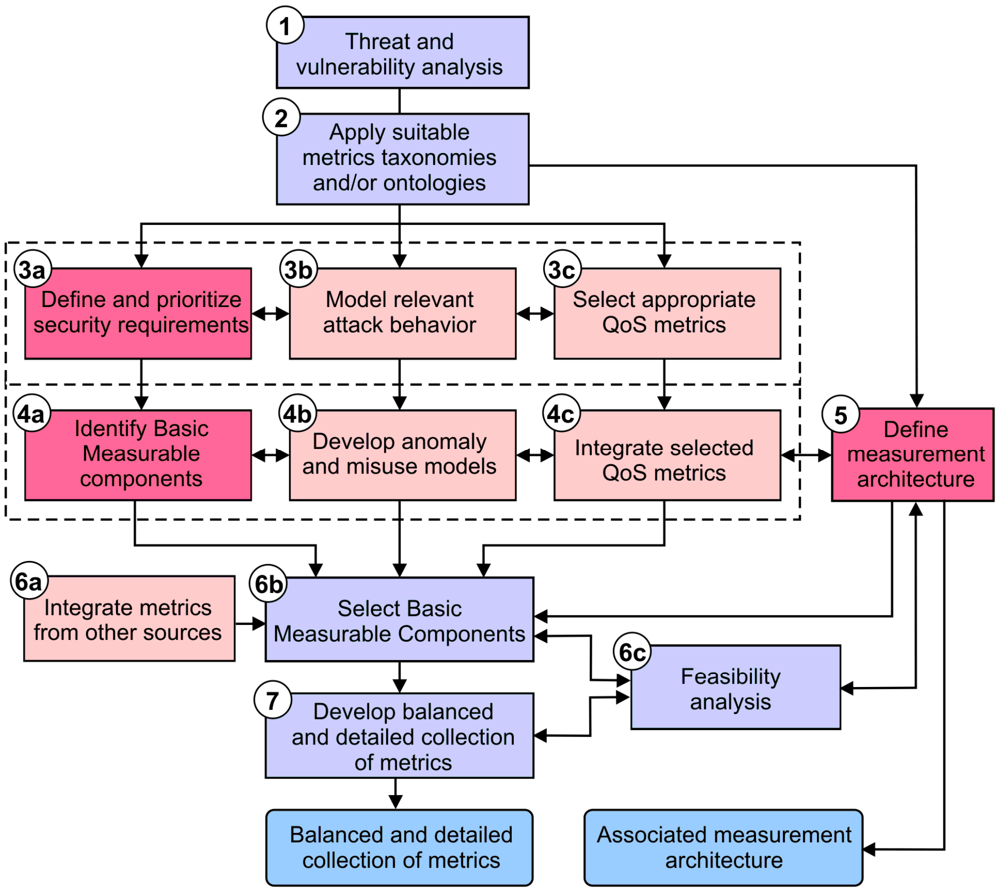
1
Threat and vulnerability analysis
2
Apply suitable metrics taxonomies and/or ontologies
3a
Define and prioritize security requirements
3b
Model relevant attack behavior
3c
Select appropriate QoS metrics
4a
Identify Basic Measurable components
4b
Develop anomaly and misuse models
4c
Integrate selected QoS metrics
5
Define measurement architecture
6a
Integrate metrics from other sources
6b
Select Basic Measurable Components
6c
Feasibility analysis
7
Develop balanced and detailed collection of metrics
Balanced and detailed collection of metrics
Associated measurement architecture

Figure 1. Security metrics development method for GEMOM. The left-most branch concentrates on security requirement decomposition.

evidence collection. The steps are as follows (points (a), (b), and (c) in Steps 3, 4 and 6 represent parallel activities):

1. Conduct a **threat and vulnerability analysis** of the SuI and its use environments, with appropriate impact and risk exposure analyses**.** This phase can be bypassed if there are valid pre-existing analysis results;
2. If applicable, utilize suitable security metrics **taxonomies and/or ontologies** (see, for example [2]) to further plan the measurement objectives and metrics types;
3. Develop **security requirements and start modeling**: (a) Define and prioritize the security requirements holistically, based on the results of Steps 1 and 2, giving the most attention to the most critical security requirements; (b) Model relevant attack strategies in prioritized order and carry out an attacker cost-benefit analysis; and (c) Select appropriate QoS metrics for the security-oriented availability metrics;
4. **Decomposition, modeling, and integration:** (a) Identify Basic Measurable Components (BMCs) from the requirements using a decomposition approach. BMCs are leaf components of the decomposition that clearly manifest a measurable property of the system; (b) Develop possible anomaly and/or misuse models; and (c) Integrate the selected QoS metrics into the collection of BMCs;
5. Define **measurement architecture** with sufficient intrinsic security-measurability (i.e., self-contained readiness for security measurement). Pay attention to readily available counters, measurement points, etc;
6. **Integrate metrics and select BMCs:** (a) Integrate metrics from other sources; and (b) Carry out BMC selection based on (c) Feasibility analysis. The feasibility analysis is an iterative stage that takes into account the measurement architecture, individual metrics, and the entire collection of metrics;
7. Develop an appropriate **balanced and detailed collection of metrics** with on-line/off-line division and the functionalities and processes in which they are used.

All steps in the process are highly iterative and the sequence of the steps can be varied if relevant information becomes available in a different sequence. Steps 1 to 3 should be started as early as possible in the system development and elaborated iteratively as the design becomes more mature. Steps 4 and 5 can be carried out in parallel. Step 5 can also already be partially started during the architectural design phase.

TABLE I. BENEFITS AND CHALLENGES OF THE PROPOSED STEPS

Step	Benefits	Challenges
1	Security-enforcing mechanisms, expressed by the requirements, can be tailored as accurately as possible to mitigate or cancel the actual threats.	Actual information about threats and vulnerabilities is sometimes difficult to obtain. It might also be too time-consuming to carry out a thorough analysis.
2	Taxonomical information systematizes metrics development and helps in planning the metrics types.	Validated security metrics taxonomies and ontologies based on experimental results are difficult to find.
3 a	Security requirements steer the R&D in the right direction and act as the baseline for evidence gathering by security metrics.	Definition of sufficient security requirements is demanding. "Negative requirements" (Section III.B) require a great deal of effort.
3 b	Knowledge about attack strategies is required for anomaly and misuse models, acting as the basis for attack-oriented metrics.	There are several ways to compromise a system, only some of which can be modeled with a reasonable amount of time and effort.
3 c	QoS metrics also reflect availability of the SuI from a security perspective.	The line between performance and security-oriented QoS metrics is fuzzy.
4 a	The decomposition expresses the relationship between the components and the requirements.	In some cases, it might be difficult to decompose the essential subcomponents or impossible to carry out the related measurements.
4 b	Anomaly and misuse models can be used in QoS and security monitoring.	The development of feasible models is time-consuming and it can be difficult to obtain proper training data.
4 c	Integration of QoS and other availability metrics increases the amount of availability evidence from a security perspective.	It is difficult to choose feasible QoS metrics from a security perspective. The type of QoS might be different from the security metrics.
5	A practical measurement architecture with proper evidence collection is necessary. Intrinsic security-measurability enables smart evidence collection.	Performance constraints and other conflicting functionality goals might complicate the design and measurement architecture and measurement support in components.
6 a	Metrics from sources other than the actual metrics development process increase the completeness of the metrics collection.	The relationship between the metrics from other sources and the security requirements are not directly visible as they are in the decomposition process.
6 b	The systematic selection of individual metrics is needed in order to increase the feasibility of the final collection of metrics.	The selection process is challenging. The need for individual metrics and the entire metrics collection must be taken into acount.
6 c	Feasibility analysis of the chosen metrics is needed in order to select the final practical collection of metrics	Feasibility analysis could require substantial information from realistic situations in which the SuI is used.
7	Eventually, detailed metrics will be needed in order to use the metrics system.	The detailed development of metrics involves several challenges, such as a lack of data from realistic situations, scaling, assessment of confidence values, and fine-tuning of decision support.

This approach is based on earlier work by the same authors: the early approach presented in [17] was enhanced in [18] with decomposition and in [16] with QoS metrics and anomaly monitoring branches. The present approach adds optionality to the threat and vulnerability analysis, changes the order of taxonomical and ontological work, emphasizes the importance of intrinsic security-measurability support, simplifies the BMC selection step (Step 6), and adds feasibility analysis as a separate stage with connections to the measurement architecture, individual metrics, and the metrics collection. The present paper also analyzes the benefits and challenges of the proposed steps (see Table I). The process is visualized in Figure 1, with the pink boxes depicting the steps for optional QoS and other metrics development.

A. *Threat and Vulnerability Analysis*

The first step in a risk-driven methodology is threat analysis, with the goal of identifying security threats and their sources, and analyzing their likelihood. It is also the starting point of security metrics development, unless sufficient threat information exists beforehand. There are various ways to carry out a threat analysis, from simply listing threats to modeling them in a more rigorous way.

The extent of threat analysis depends, for example, on the criticality of the planned applications of the SuI. The Microsoft threat risk modeling process [19] suggests the following steps:

1. Identify security objectives,
2. Survey the SuI architecture,
3. Decompose the SuI architecture to identify functions and entities that impact security,
4. Identify threats, and
5. Identify vulnerabilities.

Vulnerability analysis can be carried out once appropriate technological choices have been made. Technology and implementation-dependent vulnerabilities cause different kinds of threats to the system. Well-known vulnerability listings and repositories, such as Open Web Application Security Project (OWASP) Top 10 [20], can be used in vulnerability analysis. OCTAVE tools and methods [21] offer support for threat and vulnerability analyses.

B. *Security Requirements*

Security requirements – high-level statements of countermeasures that adequately mitigate the identified risk [22] – form the reference basis for security metrics development. They can derive from *threats*, *general organizational policies*, and *environment properties*. Security requirements derived from threats represent countermeasures, or security-enforcing mechanisms. Note the distinction between general organizational policies and *security policies*. A security policy is concerned with the design, analysis, implementation, deployment, and use of efficient and secure technology that handles the SuI in accordance with the relevant set of security rules and procedures, and is based on security requirements [22]. Environment properties contribute to the security of the SuI from the outside [23]. A security requirement of the SuI r_i is derived from applicable threat(s) ξ_i, general organizational policies p_i and the environment properties e_i [23]:

$$r_i = (\xi_i, p_i, e_i), \qquad r_i \in R, \xi_i \in \Xi, p_i \in P, e_i \in E, \tag{1}$$

where i is index number, R is the collection of all security requirements of SuI, Ξ is the collection of all security threats to be canceled or mitigated, P is the collection of all general organizational policies applied to SuI, and E is the collection of all environment properties that contribute to the security of the SuI from the outside. The effectiveness of security policies, derived from security requirements, is crucial for achieving adequate security performance and the security objectives should be inline with the security requirements. According to Firesmith [24], the most current software requirement specifications (*i*) are totally silent regarding security, (*ii*) only specify vague security goals, or (*iii*) only specify commonly used security mechanisms, such as encryption and firewalls, as architectural constraints.

Non-security requirements can have a significant effect on the quality of the system's security. Business constraints can affect the impact of security risks and the SuI's exposure to these risks. The usability and performance of security-enforcing mechanisms are also important objectives of system design. Ideally, the characteristics of excellent software requirements, including security requirements, include completeness, correctness, feasibility, necessity, prioritization, unambiguity, and verifiability [25]. The main difference between security requirements and software requirements is that most non-security requirements stipulate that the SuI must take specific necessary or desired action, while security requirements often concentrate on avoiding the occurrence of something that is undesired (*negative behavior* requirement). The lack of an understanding of and attention to negative requirements is at the root of many security problems [17].

C. Security Requirement Decomposition

A substantial mechanism in this paper's requirement-centric security metrics development approach is *requirement decomposition*. The following decomposition process, based on the work by Wang and Wulf [26], is used to identify measurable components from the security requirements:

1. Identify successive components from each security requirement (goal) that *contribute to the correctness, effectiveness, and/or efficiency* of the goal. Correctness or effectiveness goals are emphasized depending on the needs for metrics in either dimension;
2. Examine the subordinate nodes to determine whether further decomposition is needed. If it is, repeat the process with the subordinate nodes as current goals, breaking them down to their essential components; and
3. Terminate the decomposition process when none of the leaf nodes can be decomposed any further, or when further analysis of these components is no longer necessary.

When the decomposition terminates, all leaf nodes should be measurable components.

D. Measurement Architecture

In practice, it is necessary to identify measurable information and the mechanisms of how to obtain and process that data. Both on-line measurement architecture and off-line evidence collection should be designed. On-line and off-line measurements often depend on one another. In the example GEMOM system, the Monitoring Tool is the central module of the measurement architecture, with connections to the GEMOM Broker, publish/subscribe clients, Authentication and Authorization functionality, and Adaptive Security Management at the overlay level. Furthermore, the Monitoring Tool has an interface for tracking resources that are outside the GEMOM node, such as storage, memory, I/O devices, and network interfaces.

E. More Detailed Metrics Development

The potential BMCs should be selected on the basis of the feasibility, complexity and availability of information needed for the metric. The detailed development of the chosen security metrics should include definition of the following issues [1]:

- **Purpose** of the metric,
- **Target** description of the metric, for example, using composition-decomposition,
- **Formalization** of the metric into a computational or understandable form,
- **Value scale or ordering**,
- Close-to-optimal or *appropriate* **value range** depicting the 'desired level of security' and
- **Thresholds** (if needed).

Metrics can be used for many different purposes, which means that the above suggestions are not valid for all situations. Security metrics can be classified in many different ways and one metric can incorporate several metric characteristics.

IT system security comprises two independent aspects: security correctness and security effectiveness. In practical research and development, security efficiency objectives are also important. Security correctness denotes an assurance that security has been correctly implemented [2]. Security effectiveness, on the other hand, denotes an assurance that the security solutions meet the stated objectives: that is, they satisfy expectations for resilience in the use environment while not doing anything else other than what they are intended to do [2][4]. Security efficiency is concerned with the productivity dimension: the resources, time and money spent on security work and solutions [2].

The final choice of metrics depends on their use and feasibility: the metrics should add value to the decision-making process. Metrics can also assist in making the best decision based on incomplete knowledge. In addition, good security metrics are aligned with business objectives and should allow comparisons to both internal and external benchmarks.

F. Integration of Metrics

Different weights can be associated with different component metrics in order to indicate the relative importance among the component metrics. A 'close to correct' weight assignment is used in practice, because there are no analytical results for determining the relative priorities of the elements, other than the careful use of one's expertise and judgment [27]. Two important dimensions strongly affect the weight: the potential *impact* of the threat and the SuI's *exposure* to that threat Impact analysis can be iterated to react to changes in the threat environment. Accordingly, the actual threat exposure estimate, the *exposure weight*, can vary dynamically depending on the system and the use of the application. The impact and exposure weights can be integrated in the component metrics weight factors using suitable heuristics that can interpret their interaction. Figure 2 shows the effect of impact and exposure. The 'high impact, high exposure' region obviously depicts the most critical zone, resulting in higher weight coefficients. Threats that fall into the 'low impact, high exposure' or 'high impact, low exposure' categories also result in increased weighting compared to the 'low impact, low exposure' zone [1].

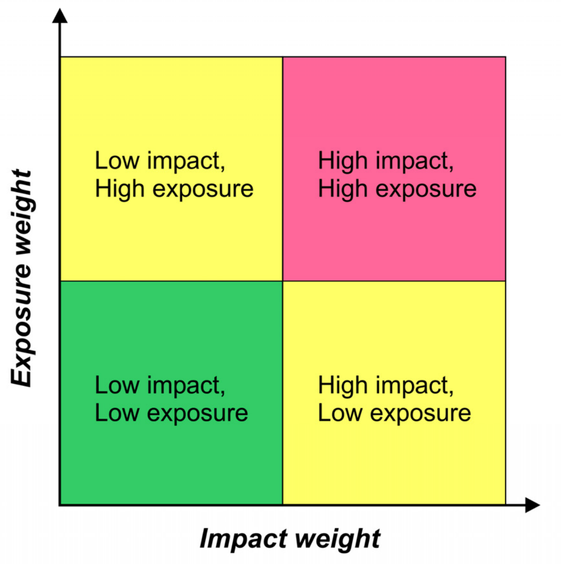

Figure 2. Threat exposure and impact dimensions [1].

The dynamic nature of threats, their impact, and the system's exposure to them can be reflected in the collection of security metrics by developing a method that relates these parameters to the actual weights used in a combination of different security metrics. In this case, the weighting acts as the 'interface' to threat dynamics from a more stable collection of security metrics.

The overall collection of chosen metrics can be managed, for example, in the form of a *balanced scorecard*, in which a score is assigned for all metrics components [1]. Different component scores are aggregated into an overall score using a suitable function. The following considerations are important when developing a balanced scorecard for security metrics [28]:

- Scorecards are a raw approximation of security risks;
- Attention should be paid to the selection of the correct component metrics;
- Scales must be normalized;
- Special care is needed if the scale types (nominal, ordinal, interval, ratio) are mixed;
- The mathematical functions used for the metrics should be carefully designed (average, sum, minimum, maximum, logical functions, and inference rules);
- Explicit rules must be defined for interpreting the aggregate score;
- The interdependencies between threats, threat agents, vulnerabilities, assets, etc., should be identified;
- The method for dealing with uncertainty, vagueness, missing information, imprecision, and contradictory information should be incorporated;
- The scorecard should support an increased security level and awareness of it;
- Essential data should be kept visible and open to peer review; and
- Standard terminology and definitions should be used.

IV. SYSTEM UNDER INVESTIGATION: GEMOM

Message Oriented Middleware (MOM) increases the interoperability, portability, and flexibility of architectures by enabling applications to exchange messages with other programs without having to know the platform on which the other application resides within the network [29][30][31]. GEMOM (Genetic Message Oriented Secure Middleware) [3] is an MOM based on the publish/subscribe messaging paradigm, the most efficient method of integrating medium to high complexity distributed systems. The GEMOM project use scenarios include a collaborative business portal, a financial market data delivery system, a road management system and a money transfer banking system [16].

A. Characteristics of the GEMOM System

The term *resilience* refers to a system's ability to return to its normal operational state after encountering an attack or other problem and continue its planned tasks. The GEMOM system is a resilient and scalable MOM that supports adaptive security management with the help of a monitoring functionality that is based on security and Quality of Service (QoS) metrics. The Adaptive Security Management system in GEMOM is able to learn and adapt to the changing threat environment without significantly sacrificing the efficiency, flexibility, reliability, and security of the system. This involves gathering relevant information both from within the system and from the environment, analyzing the collected information, and responding to changes by adjusting security functions such as encryption schemes, security protocols, security algorithms, and different authentication and authorization mechanisms. Information gathering is carried out by security and QoS monitoring services and related administration services [1]. The publish/subscribe paradigm in GEMOM is based on publishing to *topics* and subscribing to them. Topics belong to *namespaces*, a higher-level concept in a hierarchy.

B. *GEMOM System Architecture*

The GEMOM system architecture is composed of a set of communicating entities known as GEMOM Nodes or G-Nodes. Some of these G-Nodes are operational (micro nodes, depicted in blue in Figure 3 [16]) and some are managerial (macro nodes, shown in pink in the figure). The operational G-Nodes, including Message Brokers, Clients (for publishing and subscribing messages), and Authentication and Authorization Modules, interact with relevant managerial nodes according to the situation. The managerial G-Nodes include Adaptive Security Managers, Audit and Logging Modules, Anomaly Monitors, and Monitoring Tools with associated Security Measurement Managers and QoS Managers. These G-Nodes make runtime operation decisions and require a wider perspective of the system than the individual operational G-Nodes. A Message Broker is a core GEMOM functionality package that consists of an application server, numerous plug-and-play objects, configuration files, and database schemas [16]. Several components have been built in GEMOM in such a way that they exhibit properties that support security measurements. In other words, they can be considered *intrinsically security-measurable* components [4], which are entities that are inherently attuned to security measurement.

C. *Use of Security Metrics in GEMOM*

Security evidence in the form of metrics is central to GEMOM. The resulting metrics are used for different purposes [16], including:

- Security and QoS monitoring (on-line activity),
- Adaptive Security Management (a combination of on-line and off-line activities), and
- Security engineering, management and software security assurance of the system, and service (off-line activities).

D. *GEMOM Monitoring Concept*

The GEMOM Monitoring Tool is responsible for collecting security and QoS evidence, and for maintaining an appropriate metrics database in GEMOM. There is one Monitoring Tool for each Message Broker. The Monitoring Tool consists of the Monitor Core (MC) software process functionality and the Monitor Modules. The MC runs in the background and the Monitor Modules can be preconfigured or added runtime. The MC offers database and messaging services to the Monitor Modules: it connects to a GEMOM Broker and the modules use it to publish and subscribe to

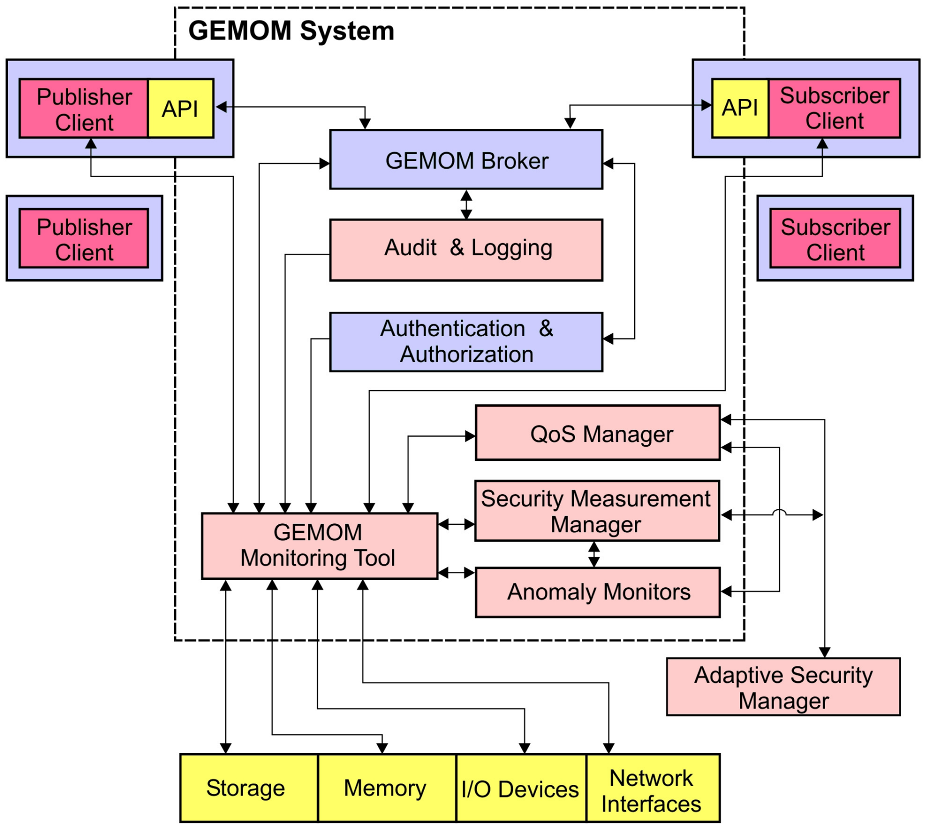

Figure 3. Example of information flows to and from the GEMOM system [16].

relevant topics in a measurement namespace [16]. A Monitoring Tool is connected to a Message Broker, an Audit and Logging Module, an Authentication and Authorization Module, a QoS Manager, an Anomaly Detector Module, a Security Measurement Manager and relevant memory (used and free), storage (hard disks, memory sticks), and network interfaces and Input/Output devices, such as a keyboard. In addition to logs, the monitoring system is able to monitor messages and metadata. Figure 3 depicts the information exchange connections of the GEMOM modules. GEMOM supports multi-point monitoring in the following way: Monitor modules can be added that are able to communicate with other Monitor modules that are monitoring other brokers, clients or modules, via publish/subscribe topics runtime. Consequently, all distributed Monitoring Tools have up-to-date information at their disposal.

E. *Adaptive Security Manager*

At the managerial G-Nodes level, the Monitoring Tools co-operate with the Adaptive Security Manager (ASM). The ASM monitors and analyzes security details based on metrics and other evidence, plans adjustments, and executes the planned adjustments through a global control loop, using both manual and automated information. In this way, the ASM manages the behavior of the overall system with the help of Monitor modules.

V. SECURITY THREATS AND SECURITY REQUIREMENTS OF GEMOM

A. *Threats and Vulnerabilities of GEMOM*

The main security threats to GEMOM are the Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. In addition to activities, a malicious authorized node can execute a message flooding attack by sending a stream of false publish or subscription messages on behalf of unauthorized nodes. Availability is the most fundamental security dimension, particularly in telecommunication systems, as shown in an example study in [32], due to the effects of DoS attacks. Availability threats have a high impact because the system, or a part of it, is most vulnerable during this type of attack. By exploiting the high vulnerability time-window, attackers can not only achieve their own specific goals, but potentially causing threats to other security dimensions [32]. If the resilience and self-protection mechanisms of GEMOM fail, an intruder could even seize the system using this strategy. Spoofing attacks can be made by sending false registration or de-registration requests concerning an authorized node. An unauthorized agent could also send registration or de-registration requests. A malicious node could also replay entire message(s) that had previously been sent by an authorized agent. General security threats of distributed messaging systems also include unauthorized nodes accessing messages, functionalities or services, masquerading attacks, eavesdropping and modifying, deleting, or tampering messages. The corruption of topics, namespaces, messages, requests, metadata, and functionalities processing data can jeopardize a system's integrity and confidentiality. An eavesdropper may be able to obtain information that was not meant to be divulged.

One potential source of authentication and authorization threats is the fact that client applications pertaining to different organizations are part of the GEMOM system. These organizations could use different authentication and identification technologies and standards. Furthermore, user credential management will not usually be under a single organization control [1].

B. *Security Requirements of GEMOM*

The results of the GEMOM threat and vulnerability analysis were mapped into security requirements that concentrate on *security-enforcing mechanisms*. The main security requirements set the basis for security metrics development in GEMOM. Note that the collection of security requirements is simplified and the following lists only the main requirements [1], each of which was allocated a prioritization description of *high*, *medium*, or *low*. Due to space limitations, only high and some medium requirements are discussed. The security requirements were originally developed during the architectural design phase of the GEMOM system [1]. They were later prioritized and iterated to remove gratuitous overlap and to increase coherence. The diagram in Figure 4 shows how the GEMOM security requirements can be classified. Adaptive security requirements are high-level and common to all other categories. In the following, the term *node* refers to a GEMOM node.

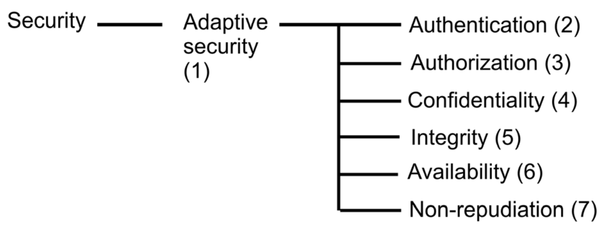

Figure 4. Categories of GEMOM security requirements [1].

TABLE II. ADAPTIVE SECURITY REQUIREMENTS [1]

Requirement	Description
Adaptive authentication (Req. 1.1)	The system should be able to choose among different authentication methods based on an adaptive trust metric.
Adaptive authorization (Req. 1.2)	The system should be able to choose among different authorization methods based on an adaptive trust metric and authentication strength.
Adaptive conf-identiality (Req. 1.3)	The encryption strength should be able to be adapted according to an adaptive trust metric.
Self-protection (Req. 1.4)	The system should be able to predict, prevent, detect, and identify attacks, and to protect itself against them.

The main *adaptive security requirements* for GEMOM, which impact all other requirements, are listed in Table II

[1]. The core functionality of the adaptive security is to be able to carry out self-protection and resilience activities. In addition, authentication, authorization, and confidentiality mechanisms are varied in GEMOM. A self-protecting system can anticipate, detect, identify, and protect itself by taking corrective actions against threats.

TABLE III. AUTHENTICATION REQUIREMENTS [1]

Requirement	Description
Authentication mechanism (Req. 2.1)	The system should be able to choose from different authentication methods.
User authentication (Req. 2.2)	Any user accessing any node should be authenticated and the node should be aware of the authentication level.
Node to node authentication (Req. 2.3)	Any node accessing any other node should be authenticated and the node should be aware of the authentication level.
Message or metadata authentication (Req. 2.4)	The node should verify the authenticity of a message or metadata.
Identity federation (Req. 2.5)	GEMOM must be able to operate in an identity federated environment.
Identity management (Req. 2.6)	User identities should be securely managed.

Authentication refers to actions in which a user's credentials are used to verify the user's identity. Table III lists the requirements for the authentication of users and other system entities. The basic generic requirements for authentication are not explicitly discussed here.

TABLE IV. AUTHORIZATION REQUIREMENTS [1]

Requirement	Description
Authorization policy (Req. 3.1)	The policy identifies specific users, user groups, and types of users, specifies the operations permitted and authorization levels on authorization objects, and specifies the delegation privilege and depth.
Access control mechanism (Req. 3.2)	Access control can use different identity-based and role-based mechanisms depending on authentication strength.
User authorization (Req. 3.3)	The Authorization Manager should verify user identity and grant access to the resource allowed.
Revoking authorization (Req. 3.4)	The authorization functionality should support the revocation of authorization, for example, to users identified as harmful.
Authorization objects (Req. 3.5)	Authorization objects are namespaces, topics, metadata, and messages. Authorization rights can be granted for single authorization objects.
Delegation of privileges (Req. 3.6)	It should be possible to delegate security credentials to an entity so that it can act on the delegators behalf. The chain of delegation depth is three.

Authorization refers to the parties that are authorized to access specific resources of the system. Table IV lists GEMOM's main authorization requirements. Authorization functionality is responsible for granting rights, including access control based on access rights. The requirements start with a definition of the authorization policy. The authorization objects must be identified.

TABLE V. CONFIDENTIALITY REQUIREMENTS [1]

Requirement	Description
Message, log and metadata conf-identiality (Req. 4.1)	Messages, logs and metadata should only be delivered to authorized receivers.
Traceability info confidentiality (Req. 4.2)	Traceability information can only be accessed by authorized users.
Confidentiality classification (Req. 4.3)	The system should be able to assign confidentiality levels to both users and message contents.
Cryptography strength (Req. 4.4)	Cryptographic algorithms should be assigned a strength value.
Storage confidentiality (Req. 4.5)	Messages stored in non-volatile media at the Message Broker should be protected.

Confidentiality countermeasures ensure that information is protected from unauthorized disclosure [1]. Table V presents GEMOM's main confidentiality requirements. Different confidentiality levels are required depending on the sensitivity requirements of information and the level of trust in specific users. The scalability of confidentiality is important; confidentiality should be ensured despite the addition or removal of system users.

TABLE VI. INTEGRITY REQUIREMENTS [1]

Requirement	Description
Message, log and metadata integrity(Req. 5.1)	The system should ensure message, log, and metadata integrity.
Environment integrity (Req. 5.2)	The integrity of node system software, add-on components, and underlying operating systems shall be verified.
Persistent data integrity (Req. 5.3)	The integrity of data passing through, stored in, or persistent to the node shall be protected.

Integrity means that data or the system processing it is not altered or destroyed in an unauthorized manner [1]. Integrity requirements are presented in Table VI. As will be seen later, integrity is a horizontal requirement area, which means that it is part of other security dimensions.

TABLE VII. AVAILABILITY REQUIREMENTS [1]

Requirement	Description
Robustness to faults (Req. 6.1)	GEMOM functionality should be available to authorized users even in the case of several node faults.
Self-healing (Req. 6.2)	The system should be able to automatically create new redundancy in case of node faults.
Sudden reconfiguration (Req. 6.3)	The system should allow for the sudden reconfiguration of the available resources.
Metadata availability (Req. 6.4)	It should be possible to distribute the metadata of Brokers to an authorized user (in order to spawn new redundancy).

Availability is the property of being accessible and useable upon demand by an authorized entity [1]. As the resilience of GEMOM is a critical need, loss of availability can represent a major security complication. Table VII lists GEMOM's availability requirements. Based on the GEMOM threat analysis, DoS types of attacks are considered a very significant threat to the availability of the GEMOM system.

TABLE VIII. NON-REPUDIATION REQUIREMENTS [1]

Requirement	Description
Non-repudiation of origin or reception (Req. 7.1)	The system should protect against an originator's false denial of having published a message or a recipient's false denial of having received a message.
Non-repudiation of published and received messages (Req. 7.2)	The Broker should be able to prove retrospectively that a specific message was published by a specific publisher or that a specific message was received by a specific subscriber at a specific time.
Non-repudiation of triggered push (Req. 7.3)	The Publisher agent should, in retrospect, be able to prove the identity of a user that triggered a message.

Non-repudiation is the property of preventing users from later denying that they performed an action (sending or receiving a message, and publishing or subscribing) [1]. Table VIII includes non-repudiation requirements.

VI. EFFECTIVENESS AND CORRECTNESS METRICS OF SECURITY-ENFORCING MECHANISMS IN GEMOM

This section decomposes the requirements [1] presented above and introduces an initial collection of security metrics and parameter dependencies based on the BMCs identified in the decomposition. Adaptive security is dealt with last because it contains requirements that depend on lower-level constructs. Most of the introduced metrics require data collection before the results can be used in automated adaptive security management or off-line security management activities.

Reliability and integrity can be considered *horizontal metrics perspectives*, because they are both part of other security requirement decompositions at the leaf level. On the other hand, the security requirement categories identified in Figure 6 can be considered *vertical metrics perspectives*, which are decomposed below. Note that horizontal and vertical perspectives in this context are only abstractions, and decompositions are presented only from the vertical perspective. However, the horizontal metrics could be also expressed as decompositions, in which the child entities of the root (reliability or integrity) are composed of all decompositions mentioned in this section.

Integrity is addressed from both horizontal and vertical perspectives. The vertical integrity metrics below address integrity-enforcing algorithms that focus on data integrity, whereas horizontal integrity metrics address the integrity objectives for different system components as a precondition to data integrity. Note that it would also be possible to arrange all integrity metrics into a decomposition representation.

A. *Reliability Metrics – A Horizontal Metrics Perspective*

In the context of a security-enforcing mechanism reliability typically refers to software reliability, but depending on the type of mechanism, can potentially be the composite of hardware and software reliability. Software reliability can be seen as a part of software quality in general. According to the widely acknowledged reliability model, time-dependent reliability $R(t)$ has the following exponential function [33]:

$$R(t) = e^{-\lambda \cdot t}, \tag{2}$$

where λ is the failure rate, t is time, and e is Napier's constant (2.71828…). This equation is valid when the failure rate is constant over time and implies a Poisson probability distribution of failures. In addition, the Mean Time Between Failures (MTBF) is calculated as follows: MTBF = $1/\lambda$. Reliability is often quantified by MTBF for repairable systems. On the other hand, Mean Time To Failure (MTTF) is used for non-repairable systems. According to Bernstein [33], the general equation can be extended as:

$$R(t) = e^{-k \cdot C \cdot t / E \cdot \varepsilon}, \tag{3}$$

where k is a scaling constant, C is software complexity, t is the continuous execution time of the software, E is the development effort, and ε denotes the investment in software engineering tools, processes and code expansion that makes the development work more effective. The goal of software testing and other assurance activities is to make the reliability as close as possible to $R(0) = 1$, which represents perfect reliability. Note that the deterioration of reliability is normally an unintentional process. More detailed analyses on software reliability are provided in [33] and [34].

The following observations affect the development of reliability metrics in GEMOM:

- Reliability can be increased by adding redundancy and diversity to the functionality in question. Both constructs support high resiliency of the system; and
- Maintainability is a key consideration in reliability. *Adaptive maintainability* functionality increases reliability and is relevant in GEMOM, which uses adaptive security management.

B. *Integrity Metrics – A Horizontal Metrics Perspective*

The component integrity of a system is a precondition to data integrity, which, in turn, indicates that data has not been modified or destroyed in an unauthorized manner. The system and its components must generate, process, maintain or transmit the data so that data integrity is preserved. *Integrity errors* are central to integrity metrics and typically include the unauthorized alteration, deletion, addition, publication and subscription of data. In general, integrity I can be measured during a data gathering time period as:

$$I = \frac{n(AC)}{n(AC) + m \cdot n(IE)}, \quad (4)$$

where $n(AC)$ is the number of data processing actions by the system component in question, $n(IE)$ is the number of integrity errors, and m is a weighting factor.

Integrity errors can be reported by built-in self-tests implemented in the system, run at specified time instants or intervals. The self-tests can investigate, among other things, the correct operation of the module, interface, memory and/or system function in question, such as the control area of a security-enforcing mechanism, consistency of the data between the original and processed data, correct sequencing of the data, and the data value range. Furthermore, integrity errors can be reported off-line by users and operational personnel. Note that system and data integrity can be disrupted accidentally or intentionally. As a result, integrity is closely linked to reliability [34].

C. Authentication

The general authentication decomposition model depicted in Figure 5 [26] is used during the process of identifying potential metrics for GEMOM authentication strength.

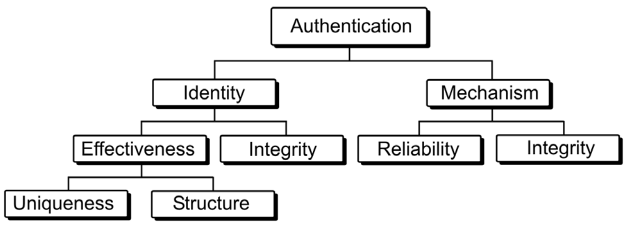

Figure 5. General authentication decomposition [26] used in GEMOM.

TABLE IX. BMCs OF AUTHENTICATION

Symbol	Basic Measurable Component
AIU	Authentication Identity Uniqueness
AIS	Authentication Identity Structure
AII	Authentication Identity Integrity
AMR	Authentication Mechanism Reliability
AMI	Authenticaiton Mechanism Integrity

Req. 2.1 states that the GEMOM system should be able to choose between different authentication mechanisms, such as a smart card, user name/password pair, and digital certificate, which represent different authentication mechanism security levels. Authentication mechanism requirements are varied by the GEMOM Adaptive Security Management functionality. If multi-modal authentication is used, that is, mechanisms combined from different authentication categories, the strength values are often higher. The model in Figure 7 suggests that the identity solution and the authentication mechanism make a significant contribution to the correctness and effectiveness of authentication. The identity tree of the decomposition emphasizes that the user identity federation (Req. 2.5) requirement is met. Identity management (Req. 2.6) is part of the authentication mechanism. There are differences in the case of user authentication (Req. 2.2), node-to-node authentication (Req 2.3), and message/metadata (Req. 2.4) authentication, and authentication metrics should be developed separately in each of these cases. The identity of the user is harder to validate than the identity of a node. See Table IX for the list of identified BMCs.

AIU, *AIS* and *AII* are mainly dependent on the identity solution, whereas *AMR* and *AMI* are mainly dependent on the authentication mechanism. Authentication Identity Uniqueness (*AIU*) is a function of the number of all identity information values divided by the values for which a *uniqueness condition* does not hold:

$$\begin{aligned} &AIU = \frac{n(ID)}{n(ID) + w_{NSID} \cdot n(NSID)}, \\ &NSID = \{x \in ID \wedge x \notin SID\}, \quad (5) \\ &SID = \{y : \exists y(ID(y) \wedge \forall z(ID(z) \rightarrow y = z))\}, \end{aligned}$$

where w_{NSID} is a weighting factor, $n(ID)$ is the total number of identity information values in use, ID is the collection of all identity information values, $ID(x)$ denotes the correspondence of identity information value x between a related actual real-world identity, $n(NSID)$ is the total number of non-unique identity information values, and $NSID$ is the collection of identity information values for which the uniqueness condition $\exists y(ID(y) \wedge \forall z(ID(z) \rightarrow y = z)$ does not hold. In other words, the maximum value of uniqueness is the case in which for every real-world identity used, there is one and only one identity information value that corresponds to a real-world identity. In the calculations, the unambiguity or ambiguity of identity information should be observed from an adversary's point of view.

Authentication Identity Structure (*AIS*) represents the security quality of the identity solution structure. The identity solution can be physical (e.g., a smart card), digital (e.g., user name/password pair), or a combination thereof. Identity structure that results from identity federation should also be taken into account. In general, the structure of a physical identity solution is stronger than one that is purely digital, provided that physical security measures have been well handled. In GEMOM, this metric has 'high, ' 'medium,' or 'low' ordinal values for different identity solutions. Attack modeling and long-term comparative data collection are needed in order to investigate the details of *AIS* quantification further.

Authentication Identity Integrity (*AII*) is dependent on the *AIS* and authentication integrity errors due to the identity solution IE_{ID}, and has ordinal values for different identity solutions. Therefore, *AII* can be denoted as a function of *AIS* and IE_{ID}:

$$AII = f(AIS, IE_{ID}) . \quad (6)$$

Failures of the authentication mechanism include the approval of unauthorized access to the part of the SuI controlled by authentication mechanism, and denial of access for authorized users. Authentication Mechanism Reliability (*AMR*) is the ability of the authentication mechanism to correctly and effectively perform its required functions under potentially hostile conditions in the operational environment. *AMR* metrics design can be based on the general software reliability metrics, via parameter *E* in Eq. 3. In particular, the following issues affect *AMR*:

$$AMR = f(T_{ANRreq}, T_{AMRtest}, R_{AMRasm}), \quad (7)$$

where T_{AMRreg} is the time spent on authentication requirements engineering, $T_{ACMRtest}$ is the time spent testing the authentication mechanism, and $R_{ACMRasm}$ is the reliability of adaptive authentication functionality and authentication maintenance activities carried out by the GEMOM Adaptive Security Management.

Authentication Mechanism Integrity (*AMI*) is a precondition for data integrity in the control area of the authentication mechanism. It means that the authentication mechanism must function correctly in order to enable data integrity. Integrity errors include the unauthorized alteration, deletion, addition, publishing, and subscribing of data. Following the example of Eq. 4, *AMI* can be measured as follows during the data gathering period:

$$AMI = \frac{n(AU)}{n(AU) + w_{IEam} \cdot n(IE_{AM})}, \quad (8)$$

where $n(IE_{AM})$ is the number of integrity errors, $n(AU)$ is the total number of authentication actions, and w_{IEam} is a weighting factor. Note that identity management solutions are also part of the authentication mechanism.

Authentication Strength (*AS*) is an aggregated metric that depicts the overall security level of the authentication solution. User-dependent Authentication Strength (AS_{usr}) can be composed from the normalized and scaled component metrics for that user:

$$\begin{aligned} AS_{usr} &= w_{AIU} \cdot \overline{AIU_{usr}} + w_{AIS} \cdot \overline{AIS_{usr}} + \\ & w_{AII} \cdot \overline{AII_{usr}} + w_{AMR} \cdot \overline{AMR_{usr}} + w_{AMI} \cdot \overline{AMI_{usr}}, \end{aligned} \quad (9)$$

where w_x is the weighting factor of component *x*, and '‾' denotes normalization and uniform scaling of the component metrics. Note that the weighting should be carefully designed to avoid instability of the overall equation. Overall Authentication Strength is the average of the Authentication Strengths of all users:

$$AS = \frac{1}{N} \cdot \sum_{i=1}^{N} AS_i, \quad (10)$$

where *N* is the number of users controlled by the authentication mechanism.

D. *Authorization*

The main measurement interest in authorization is the security strength of authentication and access control. Authorization decomposition for GEMOM is shown in Figure 8, with the corresponding BMCs in Table X.

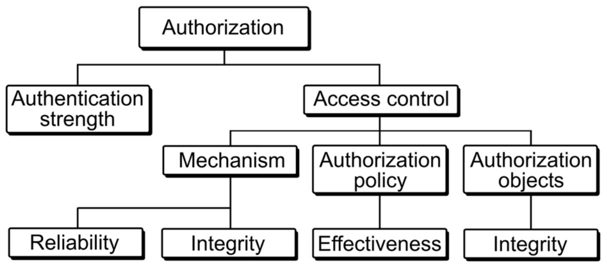

Figure 6. GEMOM authorization decomposition.

TABLE X. BMCS OF AUTHORIZATION

Symbol	Basic Measurable Component
AS	Authentication Strength (see Eq. 10)
ACMR	Access Control Mechanism Reliability
ACMI	Access Control Mechanism Integrity
APE	Authorization Policy Effectiveness
AOI	Authorization Object Integrity

Authorization policy (Req. 3.1) forms the basis for the entire authorization mechanism, while seamless co-operation with the authentication mechanism is also crucial (Req. 3.3). Consequently, the correctness and effectiveness of authentication – Authentication Strength (*AS*) – is a core metric. In GEMOM, the access control mechanism can be varied adaptively based on the authentication strength (Req. 3.2), using either *identity* or *role-based* mechanisms. Access control mechanisms can be assigned strengths that correspond to the authentication strength. As shown in Figure 6, the access control mechanism should have an adequate level of integrity and reliability. Authorization Policy Effectiveness is the main effectiveness dimension. Functionalities that revoke authorization (Req. 3.4) and delegate privileges (Req. 3.6) are part of the access control policy. The integrity of the authorization objects is also important (Req. 3.5).

Access control failures include actions that do not correspond to access control rules, such as granting access to unauthorized users and denying of access to authorized ones. Access Control Mechanism Reliability (*ACMR*) addresses the ability of the access control mechanism to correctly and effectively perform its required functions under the conditions in the operational environment. *ACMR* depends on the following parameters:

$$ACMR = f(T_{ACNRreq}, T_{ACMRtest}, R_{ACMRasm}), \quad (11)$$

where $T_{ACMRreg}$ is the time spent on access control policy and engineering requirements, $T_{ACMRtest}$ is the time spent on testing the access control mechanism and $R_{ACMRasm}$ is the reliability of adaptive access control maintenance carried out by the Adaptive Security Management. The first two affect the parameter *E* in Eq. 3.

Authorization has an important role for data integrity in that the data should not be changed without proper authorization. The objective of the Access Control Mechanism Integrity (*ACMI*) is for the access control mechanism to function correctly according to access control rules and support data integrity. Access control failures are actions that do not correspond to access control rules. In general, *ACMI* can be measured during the data gathering period as:

$$ACMI = \frac{n(AC)}{n(AC) + w_{IEac} \cdot n(IE_{AC})}, \quad (12)$$

where w_{IEac} is a weighting factor, $n(IE_{AC})$ is the number of integrity errors in access control, and $n(AC)$ is the total number of access control actions.

Authorization Policy Effectiveness (*APE*) denotes how effective the authorization policy is at performing its required functions under the potentially hostile conditions of the operational environment. Authorization policy identifies the specific users, user groups and types of users controlled by the authorization mechanism. It also specifies the permitted operations and authorization levels on authorization objects, along with delegation privileges and depth. Authorization Policy Effectiveness is enforced by the authentication and access control mechanisms. Table XI shows two operational security metrics used to address Authorization Policy Effectiveness. Development of predictive *APE* metrics requires attack modeling.

TABLE XI. EXAMPLES OF OPERATIONAL *APE* METRICS

Symbol	Basic Measurable Component
APE^{op1}	Number of authorization incidents for each reporting period/average number of authorization incidents
APE^{op2}	Hours used to manage authorization policy/average of hours used to manage authorization policy

In GEMOM, authorization objects are namespaces, topics, metadata, and messages. Authorization rights can be granted for a single authorization object. Authorization Object Integrity (*AOI*) depends on the Authorization Object Structure (*AOS*) and integrity errors caused by authorization objects IE_{AO}:

$$AOI = f(AOS, IE_{AO}) . \quad (13)$$

Access Control Effectiveness (*ACE*) can be based on normalized and scaled Access Control Mechanism Reliability and Authorization Policy Effectiveness:

$$ACE = w_{ACMR} \cdot \overline{ACMR} + w_{APE} \cdot \overline{APE}, \quad (14)$$

where w_x is the weighting factor of *x*. Access Control Correctness (*ACC*) can be based on the normalized and scaled Access Control Mechanism Integrity and Authorization Object Integrity:

$$ACC = w_{ACMI} \cdot \overline{ACMI} + w_{AOI} \cdot \overline{AOI}, \quad (15)$$

where w_x is the weighting factor of *x*. Authorization strength can be calculated from the normalized and scaled authorization BMCs similar to how Authentication Strength was calculated in Eq. 9.

Metrics from the Common Vulnerability Scoring System (CVSS) [35], which is part of the Security Content Automation Protocol (SCAP) [36], can be used to depict how easy or difficult it is to access and exploit a known vulnerability in the system. During the risk management process, a known vulnerability might be deliberately allowed to remain in the system. CVSS' *access vector metric* measures whether vulnerability is exploitable locally or remotely, and the *access complexity metric* measures the complexity of an attack that would be required to exploit the vulnerability once an attacker has access to the SuI.

E. *Confidentiality*

The decomposition of confidentiality requirements is shown in Figure 7 along with the identified BMCs in Table XII. The main components of confidentiality are cryptographic protection, physical security, and access control.

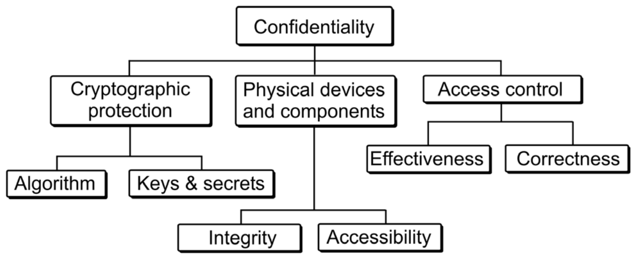

Figure 7. Confidentiality decomposition.

Special cryptographic algorithm metrics [37] can be used to measure the strength of cryptographic protection implemented by end-to-end confidentiality algorithms (Req. 4.4). It is possible to use different algorithms based on the required confidentiality level. See examples of cryptographic algorithm metrics in Table XIII. Adequate access control is a prerequisite for end-to-end-confidentiality, assured by access control correctness and effectiveness metrics. Messages, logs, metadata, and traceability information should only be delivered to authorized recipients (Req. 4.1 and Req. 4.2). The confidentiality classification requirement (Req. 4.3) also belongs to the access control function. Protection of physical devices and components which process and conserve data (storage and memory), including protection of unauthorized access to them, is vital, as stated by Req. 4.5.

TABLE XII. BMCs OF CONFIDENTIALITY

Symbol	**Basic Measurable Component**
CCA	Cryptographic protection of confidentiality algorithm(s)
CCS	Cryptographic protection of keys and secrets
ACE	Access Control Effectiveness (see Eq. 14)
ACC	Access Control Correctness (see Eq. 15)
PDCI	Physical Devices and Components Integrity
PDCA	Physical Devices and Components Accessibility

TABLE XIII. SOME CRYPTOGRAPHIC ALGORITHM METRICS [36]

Symbol	**Metrics**
KL	Key Length
ACO	Algorithm Complexity
ATS	Attack Steps
ATT	Attack Time

CCA metrics address the cryptographic strength of the overall confidentiality algorithm solution, whereas *CCS* metrics concentrate on the correctness and effectiveness of key and other secret management mechanisms. The quality of the confidentiality key and secret management architecture, *CECKSM*, also has a strong effect on *CCS*. In other words:

$$CCS = f(KL_C, CECKSM), \tag{16}$$

where KL_C is the key length of the confidentiality algorithm. The overall confidentiality algorithm solution strength is:

$$CCA = f(CCS, ACO_C, ATS_C, ATT_C), \tag{17}$$

where ACO_C is algorithm complexity, ATS_C attacks steps metric, and ATT_C attack time metric of the confidentiality algorithm.

Physical devices and components of the GEMOM system include server and client computers, smart cards, and networking equipment. The physical protection of accessibility and integrity of the servers in GEMOM is typically assumed to be high; they reside in buildings that have high physical security measures and protect the accessibility and integrity of client computers. A three-value ('high-medium-low') ordinal scale is practical in GEMOM for the assessment of the physical accessibility *PDCA*, and integrity, *PDCI*.

Confidentiality strength can be calculated as a weighted summation of the normalized and scaled confidentiality BMCs, similar to the way in which Authentication Strength is calculated in Eq. 9. CVSS includes the *confidentiality impact metric* of CVSS for measuring the impact that successful exploitation of vulnerability in the system would have on confidentiality.

F. Integrity-enforcing Mechanisms

The following sub-section discusses *integrity-enforcing mechanisms*: integrity algorithms and related keys and secrets. GEMOM requirements emphasize messages, logs, and metadata integrity (Req. 5.1), environment integrity (Req. 5.2) and persistent data integrity (Req. 5.3), with the latter two also addressed in the confidentiality decomposition. In general, environment integrity can be measured by security assurance metrics, like test coverage, and results from security and robustness testing tools.

TABLE XIV. BMCs OF INTEGRITY-ENFORCING ALGORITHMS

Symbol	**Basic Measurable Component**
CEIA	Correctness and Effectiveness of Integrity Algorithm(s)
CEIAS	Correctness and Effectiveness of Cryptographic Keys and Secrets used in Integrity Algorithm(s)

Cryptographic algorithm metrics, as in the case of confidentiality algorithms, can be used to measure the security strength of integrity-enforcing mechanisms, which consist of Correctness and Effectiveness of the Integrity Algorithm(s) (*CEIA*) and the Correctness and Effectiveness of Cryptographic Keys and Secrets used in Integrity Algorithm(s) (*CEIAS*), see Table XIV. *CEIAS* depends on the security strength of keys and secrets and the Correctness and Effectiveness of the Integrity Key and Secret Management (*CEIKSM*):

$$CEIAS = f(KL_I, CEIKSM), \tag{18}$$

where KL_I is the key length of the integrity algorithm. The overall correctness and effectiveness of integrity-enforcing the algorithm(s) is:

$$CEIA = f(CEIAS, ACO_I, ATS_I, ATT_I), \tag{19}$$

where ACO_I is the algorithm complexity, ATS_I is the attack steps metric, and ATT_I is the attack time metric of the integrity algorithm.

The *Integrity impact metric* of CVSS measures the impact of a successfully exploited vulnerability (none, partial, complete) on integrity.

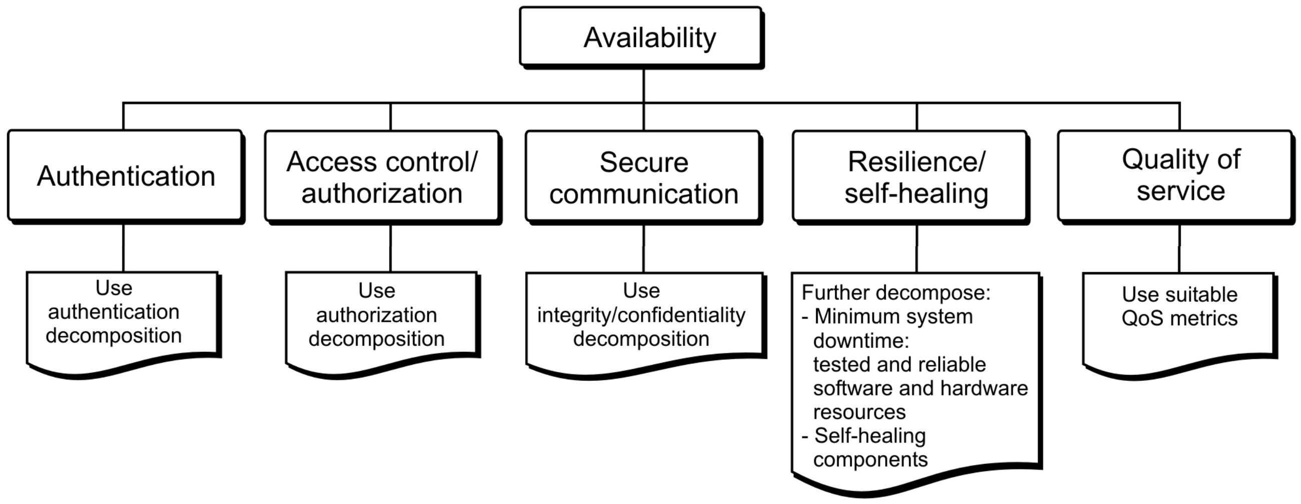
Availability
Authentication
Access control/ authorization
Secure communication
Resilience/ self-healing
Quality of service
Use authentication decomposition
Use authorization decomposition
Use integrity/confidentiality decomposition
Further decompose:
- Minimum system downtime: tested and reliable software and hardware resources
- Self-healing components
Use suitable QoS metrics

Figure 8. Availability decomposition.

G. *Availability*

Decomposition for GEMOM availability requirements is shown in Figure 8. As the resilience of the GEMOM system is a critical need, the loss of availability can be a major security complication. Furthermore, when executing resilience or self-protection actions, it is very important to preserve the overall security performance and level. The main availability requirements of GEMOM – robustness to faults (Req. 6.1), self-healing capabilities (Req. 6.2), and sudden reconfiguration capabilities (Req. 6.3) – emphasize the system's resilience.

Metadata must also be available when spawning new redundancy (Req. 6.4). Moreover, authentication, authorization and secure communication (via confidentiality and integrity) are crucial objectives that are considered to be preconditions for availability.

QoS performance metrics are also part of the availability decomposition, offering important availability information, especially for the detection of DoS attacks.

Availability has traditionally been measured as the percentage of time for which the target *system is* '*up*' or, in other words, when *information is available* from the system. However, this notion does not capture degraded states, a non-binary scale between a system being 'up' and 'down', which a resilient system such as GEMOM can demonstrate. GEMOM Availability metrics AV can be based on the following parameters:

$$AV = f(AS, AU, I, C, RI, QI), \tag{20}$$

where AS is average authentication strength, AU is authorization effectiveness, I is integrity effectiveness, C is confidentiality effectiveness, RI is the Resilience Indicator, and QI is the QoS indicator. The Resilience Indicator (RI) is based on the following parameters during the data gathering period:

$$RI = f(\min(SD), SHP, SRP), \tag{21}$$

where $\min(SD)$ is the minimum system down-time, SHP is Self-Healing Performance and SRP is Sudden Reconfiguration Performance. The self-healing requirement requires the system to be able to automatically create new redundancy of its operation.

The Self-Healing Performance (SHP) depends on the following parameters:

$$SHP = f(SHSR, TPRA), \tag{22}$$

where $TPRA$ is the Temporal Performance of Self-Healing Actions and $SHSR$ is the Self-Healing Success Rate.

The Self-Healing Success Rate ($SHSR$) is

$$SHSR = \frac{n(SSHA)}{n(SHA)}, \tag{23}$$

where $n(SSHA)$ is the number of successful self-healing actions and $n(SHA)$ is the total number of self-healing actions during the data gathering period. Similarly, the Sudden Reconfiguration Performance (SRP) of resources can be tracked by logging the success rate and temporal performance.

Until recently, QoS and security metrics lived in separate worlds. Some security attacks have affected application performance, and the most important objective of QoS management is to ensure application performance [16]. The GEMOM monitoring approach uses both metrics for security availability and application performance measurement. In GEMOM, QoS monitoring is founded on anomaly monitoring, which consists of anomaly detection processes and a hierarchy of anomaly correlation processes, and combining the output anomalies from a set of models in anomaly detectors. The *anomaly models* describe the various aspects of the normal patterns, while *misuse models* refer to the abnormal patterns of the system. Space limitations prevent a more detailed investigation of QoS metrics in this study. The QoS indicator (*QI*) is a high-level indicator of the overall QoS level and is calculated from the metrics based on the anomaly and misuse models.

TABLE XV. GEMOM MACHINE-RELATED COUNTERS

Symbol	Counter
CIT, CPT	CPU Idle Time, CPU Processing Time
AM	Available Memory
PA	Paging Activity
BU, max(*B*)	Bandwidth Utilization, Maximum Bandwidth
L, V	Latency, Visibility between two machines

TABLE XVI. PUBLISH/SUBSCRIBE ACTIVITY COUNTERS

Symbol	Counter
PPN, PPT	Publications Per Namespace, Topic
PPB, PPC	Publications Per Broker, Client
MPB, MPC	Messages Per Broker, Client
DPB, DPC	Protocol Breaches Per Broker, Client
NN, NT	Number of Namespaces, Topics

In GEMOM, some averaged *counters* that measure machine- and functionality-related information are calculated and updated at certain time intervals. They are examples of intrinsic measurement functionalities that support the measurability of availability. Counters utilized in the GEMOM availability, resilience, and QoS measurements are listed in Tables XV and XVI [16]. Increased bandwidth utilization and latency are symptoms of DoS attacks. Publication and subscription activity counters are used to further investigate whether namespaces or topics are under attack.

The *Availability impact metric* of CVSS measures the impact of a successfully exploited vulnerability on availability, using values of *none*, *partial*, and *complete*.

H. Non-Repudiation

Intuitively, non-repudiation can be seen as a stronger variant of authentication, in which identities must be verified by proof-of-identity mechanisms. Core evidence in non-repudiation is the identity of origin, receipt, submission and/or delivery of messages. Non-repudiation can be implemented using a trusted third party or, in some cases, without using one. A decomposition of non-repudiation [26] is shown in Figure 9 and the identified BMCs are shown in Table XVII.

The evidence should be consistent and reliable and its integrity should be protected. The originators and receptors must provide proof-of-identity (Req. 7.1) as well as published and received messages (Req. 7.2).

Proof-of-identity evidence is fully consistent if all *identity conclusions* from the collection of proof-of-identity evidence *E* match. Identity conclusion is a real-world identity, as shown by the proof-of-evidence material. Let E_{ID} be the subset of *E* containing evidence that all identity conclusions are the same and presenting the majority of the results in *E*. Then Consistency of the Proof-of-Identity Evidence (*CPIE*) can then be defined as follows:

$$CPIE = \frac{n(E_{ID})}{n(E)}, \qquad (24)$$

where $n(E_{ID})$ is the number of elements in E_{ID} and $n(E)$ is the number of elements in *E*.

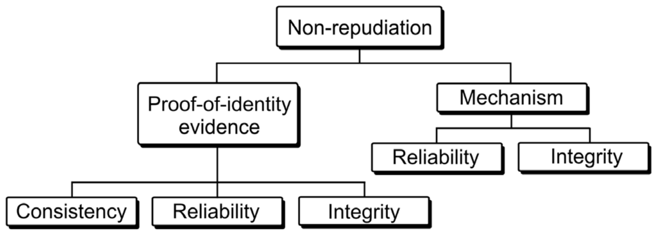

Figure 9. Non-repudiation decomposition [26].

TABLE XVII. BMCS OF NON-REPUDIATION

Symbol	Basic Measurable Component
CPIE	Consistency of Proof-of-Identity Evidence
RPIE	Reliability of Proof-of-Identity Evidence
IPIE	Integrity of Proof-of-Identity Evidence
RNRM	Reliability of Non-Repudiation Mechanism
INRM	Integrity of Non-Repudiation Mechanism

The reliability of Proof-of-Identity Evidence (*RPIE*) depends on the following factors:

$$RPIE = f(T_{3p}, R_{POIA}), \qquad (25)$$

where T_{3p} is a trust and reputation function of the third party providing proof-of-identity and R_{POIA} is the reliability of the proof-of-identity algorithm(s) in use.

The Integrity of Proof-of-Identity Evidence (*IPIE*) metrics can be designed based on the general integrity metrics of Eq. 4. Integrity errors are possible in the chain-of-proof leading to the identity conclusions: in message communication, relaying of data, storage processing, and archival and backup procedures. Furthermore, problems in application software, hardware, operating systems telecommunications equipment, user data, system data and external interfaces [34] can contribute to integrity errors in the proof-of-identity evidence.

The Reliability of Non-Repudiation Mechanism (*RNRM*) addresses the reliability of the non-repudiation mechanism and its associated processes, while the Integrity of Non-Repudiation Mechanism (*INRM*) is concerned with the integrity of them. The technical part of *RNRM* and *INRM* metrics can be based on the relevant general metrics models discussed above. The human behavioral (process) part depends on reputation and trust issues.

I. Adaptive Security

Requirements 1.1, 1.2, and 1.3 refer to the need for an adaptive trust metric that can be used when making decisions to change system parameters in authentication, authorization, and confidentiality management. In particular, authentication mechanisms, authorization mechanisms and the confidentiality algorithms will be varied based on this metric.

A user-dependent trust metric, T_{UID}, is a function of context, authentication strength, threat information, and a user-dependent trust function:

$$T_{UID} = f(\kappa, \tau, AS, \xi, t), \tag{26}$$

where *UID* is user ID, κ is context information, τ is time instant, AS_{UID} is the user's authentication strength, ξ is a variable reflecting the threat situation, and t is a trust function. Threat variable ξ can be based on off-line threat and vulnerability information and measured evidence of applicable increased threat(s) in the system.

Let the system's confidence in the trust metric T_{UID} be c_{trust}. The adaptive trust indicator AT_{UID} used for adaptive security management decision-making is:

$$AT_{UID} = c_{trust} \cdot T_{UID}. \tag{27}$$

The trust function t represents the system's trust in the user's behavior. If the user behaves well, the value of the trust function increases as a function of time asymptotically, according to a planned trust management solution. Bad behavior decreases the next value of the user's trust function. Examples of bad behavior in GEMOM include attempts to read or destroy a topic without rights, spamming, and simultaneous logging attempts.

J. Summary of BMC Sources

Table XVIII summarizes the origin of BMCs for security metrics focusing on security-enforcing mechanisms in GEMOM.

The numbers in the columns represent the security dimensions of Figure 6 (1 = adaptive security, 2 = authentication, 3 = authorization, 4 = confidentiality, 5 = integrity, 6 = availability, and 7 = non-repudiation). An '×' indicates the use of the security metrics of the respective BMC type in GEMOM [16].

TABLE XVIII. SOURCES OF BASIC MEASURABLE COMPONENTS [16]

BMC Source	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Metrics by security requirement decomposition	×	×	×	×	×	×	×
Cryptographic strength metrics		×		×	×		×
Metrics based on anomaly and misuse models (attacker-oriented weakness metrics)		×	×			×	
Availability metrics based on QoS application performance metrics						×	
System intrinsic security-relevant metrics						×	
Trust and reputation metrics	×	×					
Vulnerability metrics	×	×	×	×	×	×	×

K. Confidence, Trust and Trustworthiness Calculation

The following sub-section introduces a framework for assessing and calculating the trustworthiness of the measurements of the overall security of the system through a combination of security-based trust and trust-based security [38], see Figure 10. As sub-frameworks, the framework (*i*) contains security, trust, and confidence level calculations and (*ii*) maps trust and confidence into a trustworthiness metric. These frameworks are based on an earlier framework presented in [18], which is modified by separating trust and confidence, and then combining them to form a trustworthiness metric. The definitions of trust, confidence, and trustworthiness are similar to those presented in [39], with the following exceptions:

- **Trust** means the level of trust in the reliability of the estimation of the security level of each BMC;
- **Confidence** means the level of accuracy of and/or the assurance in the above mentioned trust relationship; and
- **Trustworthiness** means the level of trust in the reliability of the estimation of the security level of each BMC and a measurement of the degree to which the accuracy of and/or the assurance in this trust is trustworthy and can be verified.

The values of trust and confidence are both expressed as a number between zero and one, based on Bayesian statistics. A trust value equal to one indicates absolute trust and a value close to zero indicates low trust. A confidence value equal to one indicates high confidence in the accuracy of the trust value and a value close to zero indicates low confidence. Furthermore, a trustworthiness value close to zero indicates untrustworthiness and a value close to one indicates high or complete trustworthiness.

Initially, the user or expert assigns the default security levels. The normalized Security Level SL_n, $1 \le SL_n \le 10$, is calculated as follows from the measured Security Level (*SL*):

$$SL_n = \frac{\dfrac{SL}{\max(SL) - \min(SL)}}{9} + 1, \tag{28}$$

where max(*SL*) is the maximum value of *SL* and min(*SL*) is its minimum value. *SL* values from 1 to 4 indicate low security, from 5 to 7 represent good security, and from 8 to 10 indicate high security.

TABLE XIX. TRUSTWORTHINESS OF THE SECURITY MEASUREMENT

Decomposition	**BMC**	**Meas. sec. level** [1...10]	**Est. trust value** [0...1]	**Conf-idence value** [0...1]	**Trust-worth-iness** [0...1]
Authentication	AIU	*AIU*	t_{AIU}	c_{AIU}	T_{AIU}
	AIS	*AIS*	t_{AIS}	c_{AIS}	T_{AIs}
	…	…	…	…	…
Authorization	AS	*AS*	t_{AS}	c_{AS}	T_{AS}
	…	...	…	…	…
…	…	…	…	…	…

In the following, $BMC \in B$, where *BMC* denotes a BMC under investigation and *B* is the collection of all BMCs of the SuI. Similarly, $SL \in S$, where *SL* denotes a Security Level and *S* is the collection of all security levels associated with *BMC*. In order to calculate trustworthiness, like [39] and [40], some notations, associated with each $BMC \in B$ and its Security Level *SL* are defined as follows:

- $t\{BMC, SL\}$: trust value of *BMC* for security level *SL*. It has the property of $0 \le t\{BMC, SL\} \le 1$;
- $\sigma\{BMC, SL\}$: standard deviation of trust value of *BMC* for security level *SL*; and
- $c\{BMC, SL\}$: confidence value of *BMC* for security level *SL*. It also has the property of $0 \le c\{BMC, SL\} \le 1$.

Our framework uses the modified Bayesian approach [39][40][41] to evaluate trust in the security measurement of a BMC. The measurement of the security level of a BMC is assumed to be trusted with the probability of θ. In the modified Bayesian approach, several distributions can be used to represent θ, such as Beta, Gaussian, Poisson, and Binomial. The Beta distribution is the most promising of these due to its flexibility and simplicity, and because its conjugate is Beta distribution [40][41]. The trust value of the *SL* of a *BMC* can be calculated as the expectation value of the Beta distribution $Beta(\theta, \alpha, \beta)$:

$$t\{BMC, SL\} = E(Beta(\theta,\alpha,\beta)) = \frac{\alpha}{\alpha+\beta}, \tag{29}$$

where α and β denote the degree of normal behaviors and misbehaviors, respectively. In this case, while normal behavior represents a security level that can be trusted, misbehavior means that the security level is low and cannot be trusted. Trust (t) values of $0 \le t < 0.2$ indicate no trust, $0.2 \le t < 0.5$ represent low trust, $0.5 \le t < 0.8$ represent good trust, and values of $0.8 \le t \le 1.0$ indicate a high level of trust (see Table XX).

The standard deviation of the trust value $t\{BMC, SL\}$ is calculated as follows, based on [40][41]:

$$\begin{aligned}\sigma\{BMC, SL\} &= \sigma(Beta(\theta,\alpha,\beta)) \\ &= \sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}.\end{aligned} \tag{30}$$

The confidence value of *BMC* for Security Level *SL* is calculated as follows, based on [40][41]:

$$\begin{aligned}c\{BMC, SL\} &= 1-\sqrt{12\sigma(Beta(\theta,\alpha,\beta))} \\ &= 1-\sqrt{\frac{12\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}.\end{aligned} \tag{31}$$

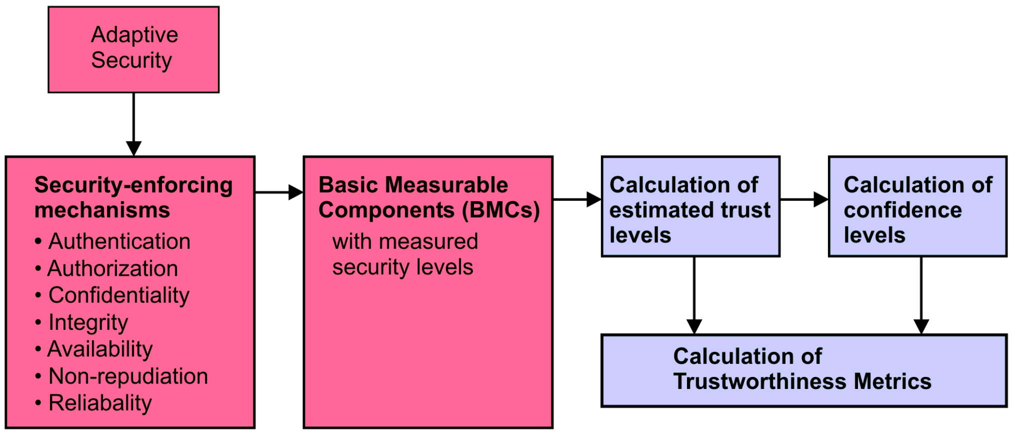
Adaptive Security
Security-enforcing mechanisms
• Authentication
• Authorization
• Confidentiality
• Integrity
• Availability
• Non-repudiation
• Reliabality
Basic Measurable Components (BMCs) with measured security levels
Calculation of estimated trust levels
Calculation of confidence levels
Calculation of Trustworthiness Metrics

Figure 10. Trustworthiness, confidence and trust calculations.

Confidence (c) values of $0 \leq c < 0.2$ indicate no confidence, $0.2 \leq c < 0.5$ indicate low confidence, $0.5 \leq c < 0.8$ represent good confidence, and values of $0.8 \leq c \leq 1.0$ indicate a high level of confidence (see Table XX).

In order to facilitate trust-based decisions, trust and confidence have been combined into a single value, as noted above. This value, trustworthiness $T\{BMC, SL\}$, is measured in the same way by combining the estimated levels of trust and confidence with some rules for the interpretation, and has the following property: $0 \leq T\{BMC, SL\} \leq 1$. As in [40], if the confidence value of *BMC* is high, the trust value of BMC plays a more important role for the trustworthiness. Thus, the trust of *BMC* should have a larger weight than the confidence value of *BMC*. Conversely, if the confidence value of *BMC* is low, the confidence value of *BMC* is clearly more important than the trust value of *BMC*. Therefore, the trust value of *BMC* should have less weight than the confidence value of *BMC*. The trustworthiness $T\{BMC, SL\}$, associated with $t\{BMC, SL\}$ and $c\{BMC, SL\}$ is defined as:

$$T\{BMC,SL\} = 1 - \frac{\sqrt{\frac{(t-1)^2}{x^2} + \frac{(c-1)^2}{y^2}}}{\sqrt{\frac{1}{x^2} + \frac{1}{y^2}}} \qquad (32)$$

where x and y are parameters that determine the relative importance of the trust value of BMC, with t denoting $t\{BMC, SL\}$ versus the confidence value of BMC, and c denoting $c\{BMC, SL\}$. [41] showed that the appropriate values of x and y are $\sqrt{2}$ and $\sqrt{9}$, respectively, for mapping trust and confidence to trustworthiness, and can be adjusted to the needs of a particular application. Trustworthiness (T) values of $0 \leq T < 0.2$ indicate a result that is not trustworthy, $0.2 \leq T < 0.5$ represent low trustworthiness, $0.5 \leq T < 0.8$ indicate good trustworthiness, and values of $0.8 \leq T \leq 1.0$ indicate high trustworthiness (see Table XX).

The assessment and calculation of the trustworthiness of the security measurement of the system as a whole is based on the aggregation and propagation of measurements carried out in the system at different levels. This can mostly be automated since the BMCs are modeled automatically on the basis of the structural and functional relations between them.

The *trustworthiness* of the security measurement of the GEMOM system (see Tables XIX and XX) is defined by the following combination of the trustworthiness-octuple:

$$\begin{aligned} \langle\ & T_1(\mathit{Authentication}), T_2(\mathit{Authorization}), \\ & T_3(\mathit{Confidentiality}), T_4(\mathit{Integrity}), \\ & T_5(\mathit{Availability}), T_6(\mathit{Non-repudiation}), \\ & T_7(\mathit{AdaptiveSecurity}), \\ & T_w(T_1, T_2, T_3, T_4, T_5, T_6, T_7)\ \rangle \end{aligned} \qquad (33)$$

where $\langle T_1()...T_7()\rangle$ is the trustworthiness-septuple, and:

- Each term $T_1() \ldots T_7()$ has one or more BMCs, each of which has an associated trust value calculated as above;
- The values of the trustworthiness-septuple, in combination, form trustworthiness of the measured security level of the system as a whole; and
- $T_w(T_1, T_2, T_3, T_4, T_5, T_6, T_7) = r_1 \cdot T_1 + r_2 \cdot T_2 + r_3 \cdot T_3 + r_4 \cdot T_4 + r_5 \cdot T_5 + r_6 \cdot T_6 + r_7 \cdot T_7$ defines a measure in which each measure of trustworthiness, ($T_1 \ldots T_7$) is dependent on the weighting factor ($r_1 \ldots r_7$) attributed to it.

TABLE XX. LINGUISTIC EQUIVALENCES OF SECURITY, TRUST, CONFIDENCE AND TRUSTWORTHINESS LEVELS

Descr.	Security Level SL [1…10]	Trust Level t [0…1]	Conf. Level c [0…1]	Trustw. Level T [0…1]
High	$8 \leq SL \leq 10$	$0.8 \leq t \leq 1.0$	$0.8 \leq c \leq 1.0$	$0.8 \leq T \leq 1.0$
Good	$5 \leq SL < 8$	$0.5 \leq t < 0.8$	$0.5 \leq c < 0.8$	$0.5 \leq T < 0.8$
Low	$2 \leq SL < 5$	$0.2 \leq t < 0.5$	$0.2 \leq c < 0.5$	$0.2 \leq T < 0.5$
No	$1 \leq SL < 2$	$0.0 \leq t < 0.2$	$0.0 \leq c < 0.2$	$0.0 \leq T < 0.2$

VII. DISCUSSION

This study has concentrated on the metrics of effectiveness and the correctness of security-enforcing mechanisms of GEMOM. Intuitively, these security metrics form the core technical metrics of the system. However, there is a need for security metrics that address the overall security quality of the system from a technical perspective and with regard to secure lifecycle management. In addition, metrics must be aligned with business management objectives.

Every security metric has challenges. Three different categories of metrics presented in this study can be identified on the basis of their feasibility challenge:

1. **Hard-to-measure metrics**, for which the main challenges are measurability, attainability, availability, scalability, and portability. This category includes special metrics such as integrity and non-repudiation metrics;
2. **Hard-to-outline metrics**, the main challenges for which are objectivity, non-bias, representativeness, and contextual specificity. The category includes, in particular, reliability metrics and cryptographic algorithm metrics; and
3. **Integrated metrics**, the main challenges for which are controllability, scalability, portability, meaningfulness, representativeness, and contextual specificity. This category contains integrated metrics that are composed of several other metrics.

Many of the metrics and parameter dependencies introduced in this study require long-term data to be gathered in order to establish a sufficient reference value.

Consequently, these metrics are not directly applicable in a realistic system. If the reference is available, the metrics are applicable. Data gathering for non-repudiation metrics might take even longer since investigations of non-repudiation are quite rare. The feasibility criteria of security metrics for software-intensive systems in general are investigated in [42].

Measurement techniques of software-intensive systems, as a whole, are not yet particularly feasible; the same applies to security measurement. The disintegration of the security research field is a challenge to the development of security metrics. Moreover, the field is suffering from the lack of a common notation to describe security issues, its different components, and multi-disciplinary dependencies. Extremely subjective measures are often used, despite their lack of value [18]. There is also entrenched reliance on subjective, human, and qualitative input [4].

Several studies have criticized the feasibility of measuring security and developing security metrics to present actual security phenomena [43][44][45]. One important source of challenges is the major role that luck plays, especially in the weakest links of security solutions. In designing a security metric, it is important to consider the fact that the metric simplifies a complex socio-technical situation down to numbers or partial orders.

The U.S. National Institute of Standards and Technology (NIST) published a report on directions in security metrics research [4]. According to this report, security metrics are an important factor in making sound decisions about various aspects of security, ranging from the design of security architectures and controls to the effectiveness and efficiency of security operations. Strategic support, quality assurance, and tactical oversight are seen as the main uses of security metrics. The NIST report proposed several lines of research for security metrics: (*i*) formal models of security measurement and metrics, (*ii*) historical data collection and analysis, (*iii*) artificial intelligence assessment techniques, (*iv*) practical and concrete measurement methods, and (*v*) intrinsically measurable components.

The present study makes preliminary advances in items (*i*), (*iv*), and (*v*). Historical data collection and analysis is still required in order to further develop the ideas presented here, with the goal of formalizing and standardizing security measurement and metrics. As reiterated in the NIST report, security metrics development poses difficult and multifaceted problems for researchers. A quick resolution is not expected and it is likely that not all aspects of the challenges are resolvable [4].

VIII. Related Work

This section investigates related work from the point of view of (*i*) metrics for security-enforcing mechanisms and the overall system's security quality, (*ii*) security assurance metrics, and (*iii*) metrics addressing the secure system lifecycle. It also notes some relevant standards and related work in trust, confidence and trustworthiness calculation. Surveys of security metrics can be found in [34][46][47][48].

Metrics research is more mature in software engineering than it is in security engineering. There are software metrics for software specifications, designs, code coverage, cohesion, complexity, performance, software development processes and resources. Fenton and Pfleeger presented a comprehensive investigation of software metrics in [5]. Particular software metrics have the potential to be used in the measurement of the overall security quality of systems.

A. Metrics for Security-enforcing Mechanisms and Security Quality

Wang and Wulf [26] described a general-level framework for measuring security based on a decomposition approach. The present study enhances Wang and Wulf's idea by introducing a security metrics development process and by proposing actual security metrics and parameter dependencies for an example system. Heyman *et al.* [49] utilized a security objectives decomposition approach in order to define a security metrics framework and to interpret the results. They associated security metrics with security patterns and exploited the relationships between security patterns and security objectives to enable the interpretation of measurements. The security metrics development approach in the present study can integrate their approach: security patterns can be investigated during the security requirement and modeling phase.

The approaches of Wang and Wulf and Heyman *et al.*, along with the ideas in the present study, show the feasibility of decomposition approaches for identifying measurable issues and for relating the developed metrics to original security objectives.

TABLE XXI. Comparison of Metrics Approaches for Security-Enforcing Mechanisms and Security Quality

Reference	**Pros**	**Cons**	**Advances in the Present Approach**
Wang and Wulf [26]	Relationships between requirements and component metrics are visible	Lack of systematic overall methodology description	Complete methodology description from threat and vulnerability analysis to a balanced and detailed metrics collection
Heyman *et al.* [49]	Association between patterns and metrics		
SCAP [36]	Aims at a standardized approach	Lack of systematic threat-driven solutions; emphasis on vulnerabilities	Complete methodology addresses both threats and vulnerabilities
Howard [51], Manadhata and Wing [52]	Attack surface is an interesting concept for overall system security quality	No systematic connection to requirements of security functions	Emphasis on requirements of security functions

The CVSS [35] (Common Vulnerability Scoring System) is a global initiative designed to provide an open and standardized method for rating information technology vulnerabilities, an example of weakness metrics. The CVSS, along with some other security vulnerability and weakness enumerations, has been integrated by the NIST into its Security Content Automation Protocol (SCAP) [36]. While this is an important standardization effort in vulnerability management, CVSS and SCAP are not complete solutions. They lack approaches to obtain evidence of the security strength of security-enforcing mechanisms and methodologies to relate actual metrics to original security objectives. For example, CVSS can tell that the highest risk of a certain application is buffer overflow, but it cannot identify the potential operational impact of a buffer overflow [50].

Another weakness metric, the *attack surface* of a software system, is parts that can be accessed by unauthenticated users, such as attackers, including the set of entry points, exit points, the set of channels and the set of non-trusted data items. Howard [51] informally introduced the notion of attack surface, and Manadhata and Wing [52] proposed an abstract attack surface measurement method, based on Howard's notion. Table XXI compares these approaches.

B. Security Assurance Metrics

The metrics framework proposed by Bulut *et al.* [53] addresses the security assurance of telecommunication services and can be integrated to the present study's metrics development approach via reliability metrics – security assurance increases the reliability of security-enforcing mechanisms. In Bulut *et al.*'s approach, a metric defines the process towards a normalized assurance level (one out of five levels) for an object in the infrastructure. Assurance metrics include test coverage and software maturity metrics. Unlike the present study, Bulut *et al.* did not present any metrics development methodology, instead emphasizing metrics selection. Another NIST effort, the Software Assurance Metrics and Tool Evaluation (SAMATE) project [54], sought to help answer various questions on software assurance, tools, and metrics.

C. Metrics for Secure System Lifecycle

The scope of this study does not include secure system lifecycle, organizational, and business management. However, it does investigate some approaches that could be integrated into the security metrics development process. Chandra and Khan [55] introduced a three-stage security estimation life cycle. The first stage, the input stage, is analogous to the threat and vulnerability analysis stage of the present study's metrics development approach. The second stage, the 'security estimation stage,' corresponds with the subsequent stages in the present approach. Finally, the 'output stage' emphasizes the overall analysis.

D. Standardization

There have been a number of major standardization and recommendation efforts for security evaluation and the certification of technical systems. However, each of these have only achieved limited success in advancing security measurability [4]. This is largely because the standards are rigid, created for certification, and carrying out their processes is time and money-consuming. The approach presented in this study and similar ones, in which the relationships between security requirements and the resulting metrics are clearly visible in the early phases of R&D, can considerably reduce the time and money needed for certification due to early problem solving. The most widely used of these efforts is the Common Criteria (CC) ISO/IEC 15408 International Standard [56] for security evaluation. During the CC evaluation process, an Evaluation Assurance Level (EAL), from EAL1 to EAL7, is assigned to the SuI. The CC standard is based on a combination of several earlier standards, including TCSEC (Trusted Computer System Evaluation Criteria) [57], ITSEC (Information Technology Security Evaluation Criteria) [58], CTCPEC (Canadian Trusted Computer Product Evaluation Criteria) [59], and FC (Federal Criteria for Information Technology Security) [60]. Interpretations of the TCSEC have been published so that they can be applied to other contexts such as the TNI (Trusted Network Interpretation of the TCSEC) [61].

E. Trustworthiness Calculation

Zouridaki *et al.* [39][41] and Li and Li [40] introduced methodologies for calculating trustworthiness. They use modified Bayesian approaches by combining trust and confidence in their methodologies. The lack of protection against malicious attacks and/or the leakage of sensitive information are challenges in their approaches, however. The present study advanced these approaches by providing a synthesis of risk-based security, security-based trust and trust-based security into one supra-additive synergistic framework.

IX. CONCLUSIONS AND FUTURE WORK

While systematic approaches for security metrics development are desirable, they are rare because the current practice of information security is a highly diverse field; widely accepted models, methods, and tools are missing.

The present study has proposed high-level security metrics and parameter dependencies for an example distributed messaging system, GEMOM. The metrics have been developed utilizing a novel security metrics development approach based on risk-driven security requirement decomposition. The emphasis is on the effectiveness and correctness of security-enforcing mechanisms for authentication, authorization, confidentiality, integrity, availability, and non-repudiation. The developed metrics can be utilized as decision-support evidence in runtime adaptive security management and off-line system security engineering and management. Some metrics require long-term data gathering before they can be utilized for on-line monitoring and management. The main challenges relate to difficulties in attaining measurements, outlining the scope, and integrating the metrics.

Reliability and integrity are generally considered to be important objectives in security solutions, especially in authentication and authorization systems. A notable observation from the study is that reliability and integrity metrics are a solid part of all decompositions of security requirements for security-enforcing mechanisms. They can therefore be considered as horizontal Basic Measurable Components, common to all security functions. The study clearly shows the dependencies between them and other issues. Intrinsic reliability and integrity-measurability increasing mechanisms and self-check functionalities built in the system and its modules increase the coverage of security evidence gathering.

This paper has described the synthesis of the risk-based assessment of BMCs, a security-based trust model, and a trust-based security model into one framework for assessing and calculating the trustworthiness of the development of measurable security. This combination extends the capabilities of each model and leverages their best features in order to support the adaptive development of quantifiable or measurable security.

Further work is needed in the feasibility analysis and validation of the proposed metrics as well as the metrics development approach. Another interesting direction for further experimentation and analysis is building intrinsic security-measurability functionality in the system and its components. As this study has concentrated on the security-enforcing mechanisms, further work is also needed in terms of investigating metrics for security quality in the system in general and metrics addressing secure system lifecycle, organizational, and business management. Future work will also include more rigorous algorithms for analyzing functional relationships of metrics and their composite effect, and how they influence the calculation of the trustworthiness of the system as a whole.

ACKNOWLEDGMENTS

The work presented in this article has been carried out in the GEMOM FP7 research project, partly funded by the European Commission. The authors acknowledge the contributions to the GEMOM system description, threat analysis, security requirements, GEMOM Monitoring Tool and anomaly monitoring made by various GEMOM partners.

REFERENCES

[1] R. Savola and H. Abie, "Identification of basic measurable security components for a distributed messaging system", SECURWARE 2009, Athens/Glyfada, Greece, Jun. 18-23, 2009, pp. 121-128.

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

[3] H. Abie, I. Dattani, M. Novkovic, J. Bigham, S. Topham, and R. Savola, "GEMOM – Significant and measurable progress beyond the state of the art", ICSNC 2008, Sliema, Malta, Oct. 26-31, 2008, pp. 191-196.

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

[5] N. E. Fenton and S. L. Pfleeger, "Software metrics – a rigorous & practical approach", 2nd Ed., PWS Publishing Company, 1997, 638 p.

[6] G. Jelen, "SSE-CMM Security metrics", NIST and CSSPAB Workshop, Washington D.C., Jun., 2000.

[7] R. Savola, "Towards a taxonomy for information security metrics", QoP 2007, Alexandria, Virginia, USA, Oct. 29, 2007, pp. 28-30.

[8] R. Savola, "A taxonomical approach for information security metrics development", Nordsec 2007 Supplemental Booklet, Reykjavík, Iceland, Oct. 11-12, 2007, 11 p.

[9] D. B. Parker, "Computer Security Management", Reston Publishing Company, Reston, Virginia, USA, 1981.

[10] ITU-T Recommendation X.805, "Security architecture for systems providing end-to-end communications", 2003.

[11] D. Longley and M. Shain, "Data and computer security: dictionary of standards, concepts and terms", Macmillan, 1987.

[12] D. Gollmann, "Computer security", John Wiley & Sons, 1999.

[13] R. C. Summers, "Secure computing, threats and safeguards", McGraw-Hill, 1997.

[14] R. Savola, "Current and emerging security, trust, dependability and privacy challenges in mobile telecommunications", DEPEND 2009, Athens/Glyfada, Greece, Jun. 18-23, 2009, pp. 7-12.

[15] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Tr. on Dependable and Secure Computing, Vol. 1, No. 1, Jan./Mar. 2004, pp. 11-33.

[16] R. Savola and H. Abie, "Development of security metrics for a distributed messaging system", AICT 2009, Baku, Azerbaijan, Oct. 14-16, 2009, 6 p.

[17] R. Savola, "Requirement centric security evaluation of software intensive systems", 2nd Int. Conf. on Dependability of Computer Systems DepCOS-RELCOMEX '07, Szklarska Poreba, Poland, Jun. 14-16, 2007, pp. 135-142.

[18] R. Savola and H. Abie, "On-line and off-line security measurement framework for mobile ad hoc networks", Journal of Networks, Vol. 4, No. 7, Sep. 2009, pp. 565-579.

[19] M. Howard and D. LeBlanc, "Writing secure code", Microsoft Press, 2003, 768 p.

[20] OWASP: Open Web Application Security Project. http://www.owasp.org./

[21] C. Alberts and A. Dorofee, "Managing information security risks: the OCTAVE (SM) approach", Addison-Wesley, 2003. 512 p.

[22] H. Abie and Å. Skomedal, "A conceptual formal framework for developing and maintaining security-critical systems", International Journal of Computer Science and Network Security, Vol. 5, No. 12, Dec. 2005., pp. 89-98.

[23] R. Savola, "Development of security metrics based on decomposition of security requirements and ontologies", ICSOFT 2009, Vol. 2, Sofia, Bulgaria, Jul. 26-29, 2009, pp. 171-174.

[24] D. Firesmith, "Specifying reusable security requirements", Journal of Object Technology, Vol. 3, No. 1, Jan./Feb. 2004, pp. 61-75.

[25] A. M. Davis, "Software requirements: objects, functions and states", Prentice Hall, Englewood Cliffs, NJ, 1993.

[26] C. Wang and W. A. Wulf, "Towards a framework for security measurement", 20th National Information Systems Security Conference, Baltimore, MD, Oct. 1997, pp. 522-533.

[27] M. Howard and D. LeBlanc, "Writing secure code", Microsoft Press, 2003, 768 p.

[28] R. C. Thomas, "12 tips for designing an infosec risk scorecard (its harder than it looks)", newschoolsecurity.com, Sep. 14, 2009.

[29] H. Li and G. Jiang, "Semantic message oriented middleware for Publish/Subscribe networks", C31 Technologies for Homeland Security and Homeland Defense III. SPIE, Vol. 5403, 2004, pp. 124-133.

[30] D. Lewis, J. Keeney, D. O'Sullivan, and S. Guo, "Towards a managed extensible control plane for knowledge-based networking", LNCS Large Scale Management of Distributed Systems, Springer, 4269/2006 (0302-9743), 2006, pp. 98-111.

[31] S. Parkin, D. Ingham, and G. Morgan, "A message oriented middleware solution enabling non-repudiation evidence generation for reliable web services", LNCS, Springer, 4526/2007 (0302-9743), 2007, pp. 9-19.

[32] R. Savola and T. Frantti, "Core security parameters for VoIP in ad hoc networks", WPMC 2009, Sendai, Japan, Sep. 7-10, 2009, 5 p.

[33] L. Bernstein and C. M. Yuhas, “Trustworthy systems through quantitative software engineering”, IEEE Computer Society, John Wiley & Sons, 2005, 437 p.

[34] D. S. Herrmann, “Complete guide to security and privacy metrics – measuring regulatory compliance, operational resilience and ROI”, Auerbach Publications, 2007, 824 p.

[35] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky, “CVSS: A Common Vulnerability Scoring System”, National Infrastructure Advisory Council (NIAC), 2004.

[36] M. Barrett, C. Johnson, P. Mell, S. Quinn, and K. Scarfone, “Guide to adopting and using the Security Content Automation Protocol (SCAP)”, NIST Special Publ. 800-117 (Draft), U.S. National Institute of Standards and Technology, 2009.

[37] N. Jorstad and T. S. Landgrave, “Cryptographic algorithm metrics”, 20th National Information Systems Security Conference, Baltimore, MD, Oct. 1997.

[38] H. Abie, “Adaptive security and trust management for autonomic message-oriented middleware”, MASS 2009, IEEE Symp. on Trust, Security and Privacy for Pervasive Applications (TSP 2009), Macau, China, Oct. 12-14, 2009, pp. 810-817.

[39] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, “E-Hermes: a robust cooperative trust establishment scheme for mobile ad hoc networks”, Ad Hoc Networks, Vol. 7, Issue 6, Aug. 2009, pp. 1156-1168.

[40] R. Li and J. Li, “Towards neutral trust management framework in unstructured networks”, IEEE Symposium on Trust, Security and Privacy for Pervasive Applications (TSP ’09), Macau SAR, China, Oct. 12-15, 2009, pp. 771-776.

[41] C. Zouridaki, B. L. Mark, M. Hejmo and R. K. Thomas, “A quantitative trust establishment framework for reliable data packet delivery in MANETs”, ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN ’05), Alexandria, VA, Nov. 7, 2005.

[42] R. Savola, “On the feasibility of utilizing security metrics in software-intensive systems”, International Journal of Computer Science and Network Security, Vol. 10, No. 1, Jan. 2010, pp. 230-239.

[43] S. M. Bellovin, “On the brittleness of software and the infeasibility of security metrics,” IEEE Security & Privacy, Jul./Aug. 2006, p. 96.

[44] D. McCallam, “The case against numerical measures of information assurance”, Workshop on Information Security System Scoring and Ranking (WISSSR), ACSA and MITRE, Williamsburg, Virginia, May, 2001 (2002).

[45] J. McHugh, “Quantitative measures of assurance: prophecy, process or pipedream?”, Workshop on Information Security System Scoring and Ranking (WISSSR), ACSA and MITRE, Williamsburg, Virginia, May, 2001 (2002).

[46] A. Jaquith, “Security metrics: replacing fear, uncertainty and doubt”, Addison-Wesley, 2007.

[47] 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 IATAC, May 2009.

[48] R. Savola, “A novel security metrics taxonomy for R&D organisations”, 7th Annual Information Security South Africa (ISSA) Conf., Johannesburg, South Africa, Jul. 7-9, 2008, pp. 379-390.

[49] T. Heyman, R. Scandariato, C. Huygens, and W. Joosen, “Using security patterns to combine security metrics”, 3rd Int. Conf. on Availability, Reliability and Security (ARES), 2008, pp. 1156-1163.

[50] Systems Engineering Research Center, “Security metrics”, Draft, Jan. 2010.

[51] M. Howard, J. Pincus, and J. M. Wing, “Measuring relative attack surfaces”, Workshop on Advanced Developments in Software and Systems Security, 2003.

[52] P. K. Manadhata, D. K. Kaynar, and J. M. Wing, “A formal model for a system’s attack surface”, Technical Report CMU-CS-07-144, Jul. 2007.

[53] E. Bulut, D. Khadraoui, and B. Marquet, “Multi-agent based security assurance monitoring system for telecommunication infrastructures”, CNIS 2007, Berkeley, CA, USA, Sep. 24-27, 2007, 6 p.

[54] P. E. Black, “SAMATE’s contribution to information assurance”, IAnewsletter, Vol. 9, No. 2, 2006.

[55] S. Chandra and R. A. Khan, “Object oriented software security estimation life cycle – Design phase perspective”, Journal of Software Engineering, 2008, Vol. 2, Issue 1, pp. 39-46.

[56] ISO/IEC 15408-1:2005, “Common Criteria for information technology security evaluation – Part 1: Introduction and general model”, ISO/IEC, 2005.

[57] United States Department of Defense: Trusted Computer System Evaluation Criteria (TCSEC) “Orange Book”, DoD Standard, DoD 5200.28-std, 1985.

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

[59] Canadian System Security Centre, “The Canadian Trusted Computer Product Evaluation Criteria”, Version 3.0e, Jan. 1993, 233 p.

[60] U.S. National Institute for Standards and Technology and National Security Agency, “Federal Criteria for Information Technology Security – Draft Version 1.0”, 2 volumes, 1993.

[61] U.S. National Computer Security Center, “Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria – Version 1”, NCSC-TG-005, 1987.