Increasing Measurability and Meaningfulness of Adaptive Security Monitoring by System Architectural Design and Mechanisms

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

Petri Heinonen
VTT Technical Research Centre of Finland
Oulu, Finland
E-mail: Petri.Heinonen@vtt.fi

***Abstract* — Decision-making in adaptive security management relies on sufficient and credible security evidence gathered from the system under investigation, expressed and interpreted in the form of metrics. If security measurability is not paid enough attention in advance, the availability and attainability of security evidence is often a major challenge. We propose and analyze practical and systematic security-measurability-enhancing mechanisms and system architectural design choices that enable and support adaptive and distributed security monitoring of software-intensive systems. The mechanisms are discussed in detail in the context of an adaptive, distributed message-oriented system. Examples of associated security monitoring techniques implemented in this environment are given. The study also discusses the feasibility of the proposed mechanisms. Security-measurability-enhancing mechanisms are crucial to the wider acceptance of security metrics, measurements, and associated tools and methods.**

Keywords* — *security monitoring; security metrics; adaptive security management; security measurability; message-oriented systems

I. INTRODUCTION

The constantly increasing complexity and connectedness of telecommunications and software-intensive systems, together with the greater number and variety of critical business applications operating in these systems, have heightened the need to implement carefully designed security mechanisms. As security threats and vulnerabilities, context of use, and protection needs change dynamically, adaptive security management and monitoring can provide effective and flexible security solutions. Security metrics can be used in resilient, self-protective, and self-healing systems to offer sufficient and credible security evidence for adaptive decision-making.

The US National Institute of Standards and Technology (NIST) published a roadmap report on directions in security metrics research [4]. This report argued that 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. The NIST report calls for practical and concrete measurement methods and *intrinsic security-measurability-enhancing mechanisms* within systems, motivating the research discussed in this study. Many security measurement challenges have their origin in the poor measurability support from the system under investigation. Security measurability can best be supported by designing enough support for it into the systems. Some of the mechanisms discussed in this study can help the designers with this task.

Adaptive security management should be capable of adapting to different use environments, contexts, and dynamic security threats. For instance, there can be different levels of authentication requirements: in some cases, strong authentication is needed and in others, multi-factor authentication mechanisms should be used.

The primary contribution of this study is the analysis of security-measurability-enhancing mechanisms for a distributed, adaptive security monitoring system, originally introduced in earlier work in [1]. Compared with the work in [1], this study provides more details and examples of the monitoring approach, and explains the mechanisms in greater detail. The mechanisms are investigated in the context of an example system, a distributed messaging system called Genetic Message Oriented Middleware (GEMOM) [2][3], which was developed in the European Commission's Framework Programme 7 project GEMOM (2008–2010). The GEMOM project developed a full-featured message broker, monitoring tool, and adaptive security management component, and it prototyped an intelligent fuzzing tool and anomaly detectors. The prototypes were validated in the following business-critical applications: banking transaction processing, financial data delivery, dynamic road management, collaborative business portal, and a dynamic linked exchange for procurement.

The rest of this paper is organized as follows. Section II briefly introduces the GEMOM system architecture and its security monitoring approach, while Section III discusses the use of security metrics in adaptive security monitoring. Section IV proposes and discusses novel technical mechanisms to enhance security measurability, and Section V analyzes the feasibility of the proposed mechanisms. Section VI discusses related work, and Section VII offers some concluding remarks and discusses future work.

II. SECURITY MONITORING APPROACH OF GEMOM

In the following, we discuss briefly the security monitoring approach of the GEMOM system. Although this study uses GEMOM as a reference system, the discussed solutions can be applied to many types of system architectures or communication mechanisms.

A. GEMOM Architecture

Message Oriented Middleware (MOM) solutions enable applications and systems to exchange messages with their communication parties without having to know the actual platform on which the application resides within the network. GEMOM is a scalable, robust, and resilient MOM system, the basic architecture of which is formed with GEMOM Nodes (G-Nodes) [5]. G-Nodes are either operational or managerial. Different configurations of G-Nodes can be used, depending on the current and future needs of the application or service.

One example GEMOM *subnet*, an operational system formed with G-Nodes, is depicted in Figure 1 [1]. The operational G-Nodes include Brokers, Publish Clients, Subscribe Clients, and Authentication and Authorization modules. Managerial G-Nodes include Adaptive Security Managers, Audit and Logging modules, Anomaly Monitors, Monitoring Tools (MTs), Security Measurement Managers, and Quality of Service (QoS) Managers. The managerial G-Nodes carry out runtime monitoring, control, and decisions in collaboration with the operational nodes [1]. The actual GEMOM system can consists of several subnets, or different types of nodes connected together, depending on the needs of the application, and resilience and performance issues.

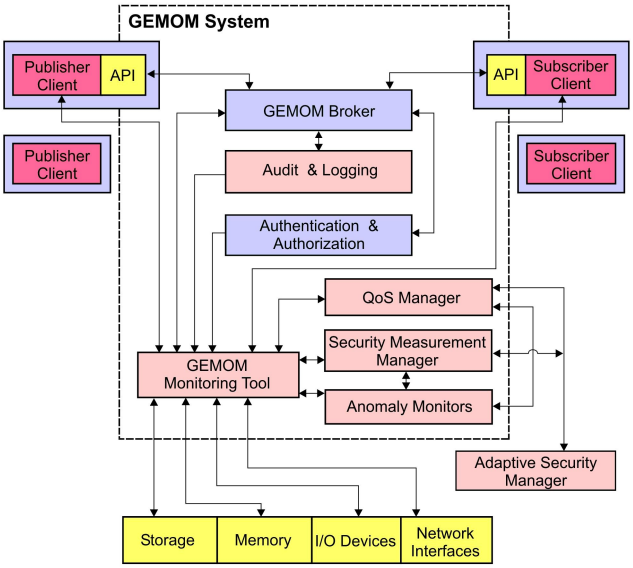

Figure 1. An example GEMOM subnet architecture [1]

Figure 1 also visualizes the connections to and from an MT [1]. The main entity responsible for adaptive security management in GEMOM, the Adaptive Security Manager node resides in the Overlay Manager and can be used to manage several subnets. In addition to the different types of nodes, the MT interfaces directly to platform resources, such as storage, memory, Input/Output (I/O) devices, and network interfaces. The Audit and Logging node provides internal functionality information at method or function level. For example, failures in methods and function calls, and user action logging can be monitored using this node.

B. Topics and Namespaces

GEMOM uses a publish/subscribe paradigm for message communication: authorized nodes can subscribe to *topics* and *namespaces* (see Figure 2) [1]. Publisher and Subscriber Clients have a core role in the content management approach of GEMOM. Communication architecture based on the publish/subscribe principle supports flexibility and scalability objectives well.

The topic contains published data in ⟨*key*, *value*⟩ format. The published data can be delivered to a Subscriber Client or another authorized G-Node, such as a node used in monitoring. Namespaces are mainly used for classification, with a namespace being a prefix for each topic. Namespaces are a higher level hierarchical construct compared with topics. For example, a measurement namespace can be reserved for measurement purposes in monitoring. Similar or associated topics belong to the same category, represented by the namespace. A topic can be shared by several Brokers or assigned to just one [1].

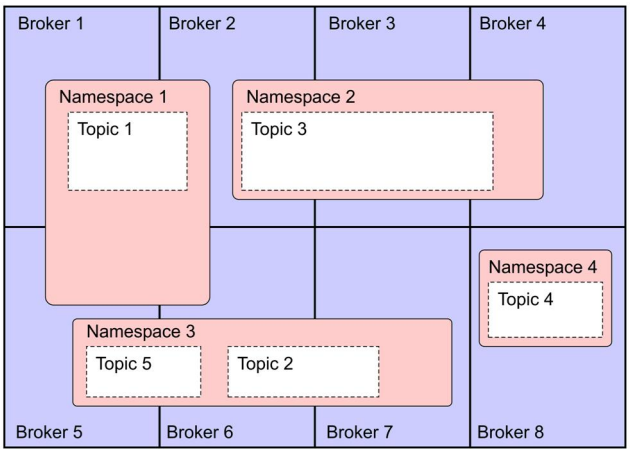

Figure 2. A visualization of topics and namespaces [1]

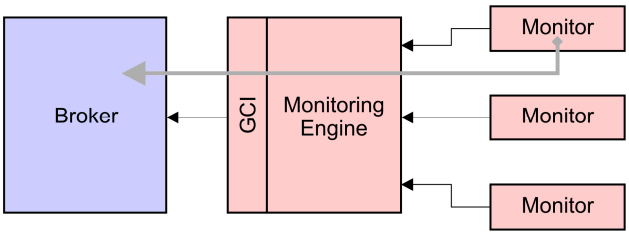

Figure 3. Communication between the Broker and the Monitoring Tool [1]

The actual physical locations of namespaces and topics are transparent to application users. A Subscriber can make a subscription to a topic or a namespace. The GEMOM Authentication and Authorization module manages the access rights to them. Namespace changes to a namespace

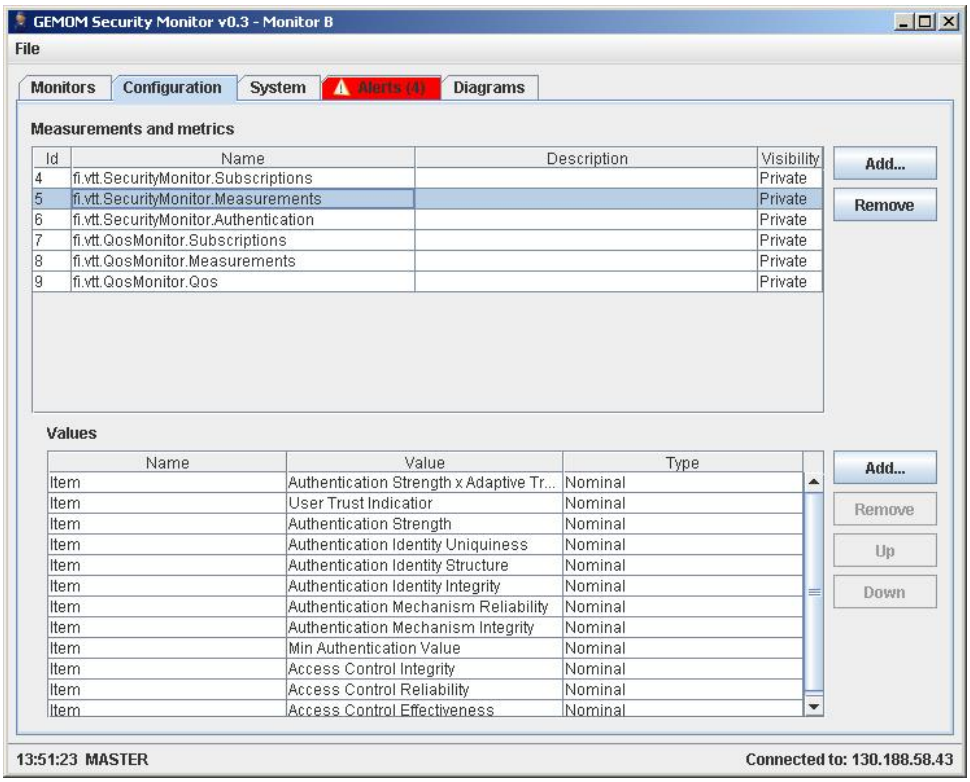

Figure 4. Configuration of authentication and authorization metrics in the MT

subscribed to by a Subscriber Client, such as deletions, name changes, or additions, or removals of topics, will continually be reported by the Broker to it. Topic changes (contents change, name changes, addition or removal of topic) to a subscribed topic are also reported [1]. The depth of reporting can be configured based on needs. Table I shows examples of namespaces and topics connected to the MT functionality.

TABLE I. EXAMPLES OF NAMESPACES AND TOPICS CONNECTED TO MT FUNCTIONALITY

Namespace, topic	Explanation
`/smon/modules, SecurityMonitor_x`	Under the `modules` namespace, there is a list of topics that contains module configurations for each security monitor.
`/smon/alerts/ SecurityMonitor_x, AnomalyMonitor`	`Alerts` namespace contains subnamespaces for each monitor. Subnamespaces contain topics for alerts sent from each Monitor module.
`/smon/metrics/ SecurityMonitor_x, Metric_y`	`Metrics` namespace contains subnamespaces for each monitor. Under them, there are metric configurations.

The GEMOM Broker is a core module of the whole system. It is a functional entity that consists of an application server, numerous plug-and-play objects, configuration files, and database objects [5].

C. Monitoring Tool and Monitor Modules

An MT controls the collection and further processing of security and QoS evidence and manages associated metrics and measurement databases. The main functional components of MT are the Monitor Engine and Monitor modules. The Monitor Engine implements Monitor Core software process functionality with the database and messaging service running in the background. The Monitor Engine does not carry out any monitoring itself but offers basic monitoring support services to the Monitor modules. The Monitor modules can either be pre-configured or added dynamically during the runtime operation [1].

In a GEMOM subnet, an MT is connected directly to the GEMOM Broker. The connection between the Monitor modules and Brokers is arranged via the GEMOM Client Interface (GCI) (see Figure 3) of the Monitoring Engine. GCI is a special interface component optimized for the GEMOM environment. The interface to other modules of the subnet is arranged differently: other modules use the GEMOM publish/subscribe mechanism to communicate and measure, publish and subscribe to relevant topics in a *measurement namespace* [5]. MTs use this mechanism to connect to Authentication and Authorization modules, QoS Managers, Anomaly Detector modules, and Security

Measurement Managers, as well as relevant used and free memory entities, storage (hard disks, memory sticks), network interfaces, and input/output devices (e.g., keyboard) [1]. In addition to logs and direct measurement results, the measurement data can include messages and metadata relevant for the measurement, as well as reports from associated security assurance tools.

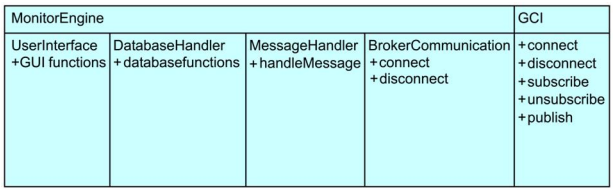

Figure 5. Sub-modules of Monitor Engine [1]

The monitoring system allows Monitor tools to be configured for different purposes, such as security, QoS, availability, and content monitoring. Only metrics, interfacing, and measurement spaces differ depending on the monitoring objectives. In the GEMOM environment, the QoS Monitor, Security Monitor and Anomaly Monitor tools have been implemented to be used for monitoring needs. Figure 4 shows a screenshot from the GEMOM MT, depicting the configuration management of the authentication and authorization metrics. The MT is in master mode. Alerts can be investigated by clicking the 'Alerts' tab during monitoring. Graphical visualization of monitoring results is obtained from the 'Diagrams' tab.

The MT Monitor Engine starts its operation as a Windows service. The sub-modules of the Monitor Engine and GCI interface are listed in Figure 5. After starting, the Monitor Engine will run in the background and be ready to establish a connection to a Broker using the BrokerCommunication sub-module by using the GCI interface. This sub-module also acts as a mediator for all publish and subscribe calls that originate from the MT [1].

Once the connection has been established, the Master MT switches to online mode. The first MT in the GEMOM subnet will obtain Master status. In the case of a machine crash or reboot, every MT is able to start up automatically and switch to online mode after the machine is up and becomes operational again. The MessageHandler sub-module handles messaging between the Monitor Engine and Monitor modules. The DatabaseHandler offers database services for Monitors. The UserInterface sub-module implements the user interface that is used for the configuration and management of MTs and the entire monitoring system. All events, statistics, status,

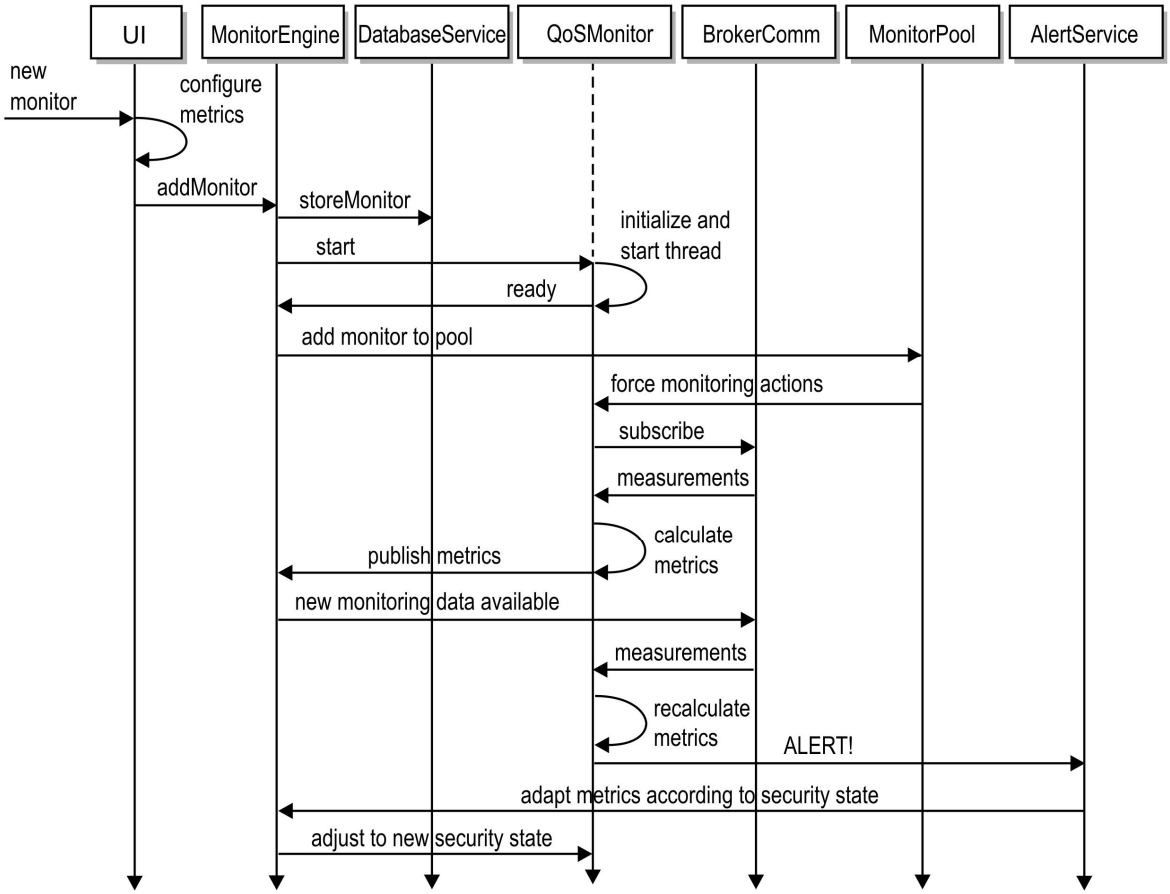

Figure 6. Adding a Monitor module to the system

functionality, and alarms are reported and visualized by the UserInterface sub-module. In addition, all information is available in special namespaces and topics [1]. Monitor modules that communicate with other MTs are connected to other Brokers and modules can be added to support multi-point monitoring needs. All distributed MTs have up-to-date monitoring information at their disposal. The MTs that reside close to the measurement points gather data from these points and make them available to other MTs.

D. Addition of New Monitor Modules

Security monitoring is typically carried out in co-operation with several Monitor modules, as completeness and meaningfulness of the measurements often require information from several system components and stakeholders.

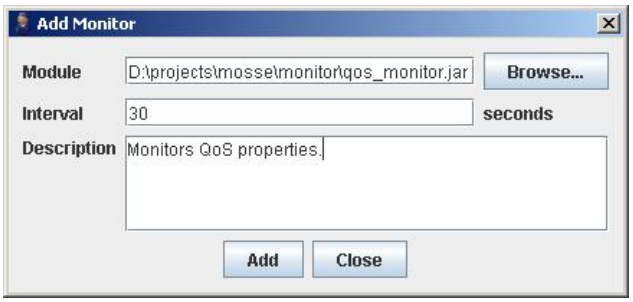

Figure 7. Adding new monitors is straigthforward in GEMOM. It is crucial for security-measurability to pay attention to ease of use

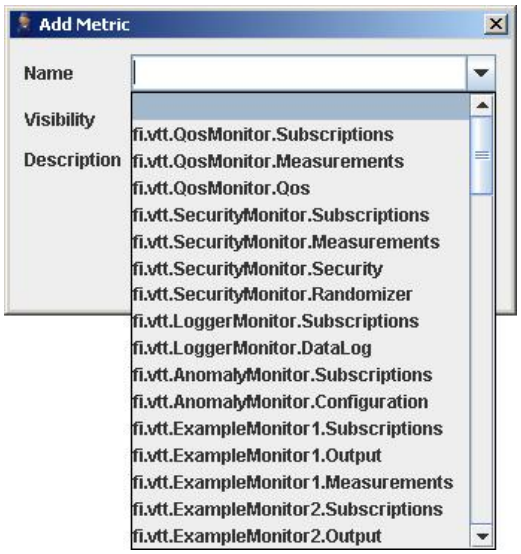

Figure 8. Adding metrics in a Monitor

Adding new Monitors is relatively straightforward; the main task in this context is to define appropriate decision-making algorithms and metrics. Figure 6 shows the functionality of adding a Monitor module to the system and Figure 7 a dialog box for end-user interaction during this operation. Once the metrics have been configured from the User Interface (UI), the Monitor can be saved and started. The Monitor initializes its own parameters and starts a thread. After this, it is a shared resource for the whole monitoring system. The Monitor carries out preliminary analysis and then receives measured data from the system. The measured data ('raw data') are used by the configured metrics collection and the metrics results are published for use by authorized subscribers, such as the Adaptive Security Manager component. If criteria for anomalies or other critical situations are met, alarms are raised. After this, all relevant Monitor metrics are adapted to the current situation. Adaptation can be carried out by, e.g., tightening selected requirements and criteria for new alarms, changing the frequency of measurements, or reducing the processing load.

E. Addition and Configuration of Metrics

Metrics can be added to the GEMOM Monitors with an easy-to-use user dialog box (see Figure 8). Metrics consist of logical expressions with either raw input data or results from other metrics. For each metric, the following parameters need to be configured: Metric ID, input data, output data, metric expression formula, threshold values, and timing. The following configuration presents an example of a Security Monitor metrics configuration:

```
fi.vtt.SecurityMonitor Metrics:
  fi.vtt.SecurityMonitor.Subscriptions

Topic@Namespace:130.188.58.43:7891@/_sys/clients
  fi.vtt.SecurityMonitor.Measurements
    Item:nAUTH
  fi.vtt.SecurityMonitor.Security
    MinimumAuthenticationStrength:0.3
    InitialNumber:0.7
    Lowest:0
    Highest:1
    Speed:0.05
```

According to the above configuration, `SecurityMonitor` subscribes to a topic that is updated by a Broker. It measures `nAUTH` (normalized authentication strength, *AS* in Eq. 1). If `nAUTH` drops below 0.3, `SecurityMonitor` will send an alarm. The following configuration is an example of QoS monitoring:

```
fi.vtt.QoSMonitor Metrics:
  fi.vtt.QoSMonitor.Subscriptions

Topic@Namespace:130.188.58.43:7891@/_sys/clients

Topic@Namespace:130.188.58.43:6148@/_sys/clients
  fi.vtt.QoSMonitor.Measurements
    Item:iCPU
    Item:aMEM
    Item:dBR
    Item:pBR
  fi.vtt.QoSMonitor.QoS
    MinimumAvailableMemory:500
    MinimumIdleCPU:50
    MaximumDataRate:1000
    MaximumPublishRate:10
```

With this configuration, `QoSMonitor` subscribes to the topics that are updated by Broker and GAgent and is able to read variables from them. It is configured to measure `iCPU` (CPU idle time, IT_{cpu} in Table IV), `aMEM` (available memory in Megabytes, *AM* in Table IV), `dBR` (data rate, bytes per second), and `pBR` (publication rate, publications per second). It also has a QoS metric that indicates when values are not acceptable. If a measured value is not in the acceptable range, `QoSMonitor` will send an alarm. The above configuration enables `QoSMonitor` to plot the measured values in a diagram.

III. Using Security Metrics for Adaptive Security Monitoring

The following section briefly discusses the security metrics that form the basis for security monitoring. It also provides an example of their use in adaptive security decision-making.

A. Development of Security Metrics in a Hierarchical Way

The term *security metrics* has become standard when referring to the security level, security performance, security indicators or security strength of a system under investigation – a technical system, product, service, or organization [6]. The general motivation for security measurements is the common argument that an activity cannot be managed well if it cannot be measured [7]. The above is particularly applicable to adaptive security management. Security solutions with varying strength levels are required in distributed networked systems such as GEMOM so that they can manage security in an adaptive manner according to the needs of varying situations like the context and dynamicity of security threats. Security metrics provide the means with which to score different solutions during the system's operation [7]. 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 [8]. Security metrics can be used for decision support in assessment, monitoring, and prediction.

Our earlier work [7][9] analyzed the collection of security metrics heuristics developed to measure the correctness and effectiveness of the security-enforcing mechanisms of the GEMOM system. Security metrics have been developed for adaptive security, authentication, authorization, confidentiality, integrity, availability, and non-repudiation mechanisms. Extensive surveys of available security metrics can be found in [10][11][12]. The earlier mentioned research [7] introduced an iterative methodology for security metrics development that has been simplified here:

1. Carry out prioritized threat and vulnerability analysis of the system under investigation.
2. Use suitable security metrics taxonomies and/or ontologies to further plan the measurement objectives and metrics types.
3. Develop and prioritize security objectives.
4. Identify Basic Measurable Components (BMCs) from the security requirements using a decomposition approach. BMCs are leaf components of the decomposition that clearly manifest a measurable property of the system. Similarly, decompose the system architecture into components.
5. Define measurement architecture and evidence collection**.** Match the BMCs with the relevant system components with attainable measurable data.
6. Integrate metrics from other sources and select BMCs based on feasibility analysis.
7. Develop an appropriate balanced and detailed collection of metrics from the BMCs.

BMCs are identified by security objective decomposition [6][7]:

1. Identify successive components from each security requirement that *contribute to its security correctness, effectiveness and/or efficiency* [6], or another security property in question;
2. Examine the subordinate nodes in order to determine whether further decomposition is required. If so, repeat the process with the subordinate nodes as current goals, breaking them down into their essential components.
3. Terminate the decomposition process when none of the leaf nodes can be decomposed further or when further analysis of these components is no longer necessary.

Originally, the idea of security objective decomposition was proposed by Wang and Wulf [13]. Note that the mechanism of developing security metrics discussed above is one example of systematizing this task. There are so many other possible ways to develop metrics.

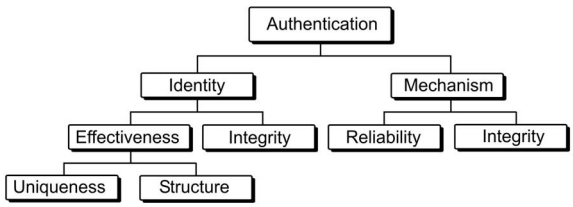

Figure 9. An example of authentication decomposition [13]

TABLE II. BMCs for Authentication [7]

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

B. An Example of Authentication Decomposition

Figure 9 provides an example of high-level decomposition of main authentication objectives, while Table II shows the associated BMCs identified from this decomposition. The identity concept and authentication mechanism essentially contribute to the security strength of authentication. A more detailed explanation of the listed BMCs can be found in [7].

The metrics can be aggregated in the form of a weighted sum, resulting in Authentication Strength *AS* [7]:

$$AS = w_{AIU} \cdot \overline{AIU} + w_{AIS} \cdot \overline{AIS} + w_{AII} \cdot \overline{AII} + w_{AMR} \cdot \overline{AMR} + w_{AMI} \cdot \overline{AMI}, \quad (1)$$

to quote, "where w_x is the weighting factor of component x and '¯' denotes normalization and uniform scaling of the component metrics."

C. An Adaptive Authentication Management Example

The following example illustrates the use of the authentication metrics of Table II in adaptive security monitoring.

The GEMOM system requires a minimum Authentication Strength, min(AS_{usr}), for each user, *usr*. This value differs between the normal mode and the mode during suspicion of a Denial of Service (DoS) attack ('DoS alarm mode'). The conditions for min(*AS*) are [1]

$$\min(AS_{USR}): \quad AT_{usr} \cdot AS_{usr} \geq \theta_1 \wedge \quad \min(AIU_{usr}, AIS_{usr}, AII_{usr}, AMR_{usr}, AMI_{usr}) \geq \theta_2, \quad (2)$$

to quote, "where AT_{usr} is an adaptive trust indicator, θ_1 is the general Authentication Strength threshold and θ_2 is the threshold for each component metric. For instance, thresholds could be set as follows: during normal operation, $\theta_1 = 0.5$ and $\theta_2 = 0.2$, and during DoS alarm mode, $\theta_1 = 0.6$ and $\theta_2 = 0.3$." More details on parameters can be found in [7].

Below, we show a highly simplified scenario of how the authentication metrics discussed above can be used in an adaptive manner in GEMOM. The example consists of seven steps that represent the GEMOM security monitoring system in different authentication situations (or 'steps') [1]:

1. Authentication of the user *usr* authenticating himself/herself for the first time in an office environment using a smart card.
2. Identification of *usr* in an office environment using a user name/password pair. There are several weeks between Steps 1 and 2, and the value of the trust indicator has been increased.
3. Availability has dramatically decreased (increased delay and decreased QoS), possibly due to a DoS attack.
4. Identification of *usr* in the office environment using the user name/password pair. The Authentication Strength of *usr* falls under the threshold, causing an alarm.
5. Identification of *usr* in an office environment using an X.509 certificate. The Authentication Strength is now over the required threshold level and the authentication is successful.
6. Normal mode is resumed after the DoS attack mode. The adaptive trust indicator has now been increased, but *usr* has attempted to read a topic without rights to do so. Consequently, the adaptive trust indicator must be decreased by a certain amount.
7. Identification of *usr* in a home office environment, with a GEMOM smart card and user name/password pair in use. The authentication is successful.

TABLE III. STEPS 1–7 OF AN AUTHENTICATION MONITORING EXAMPLE [1]

Param.	**St. 1**	**St. 2**	**St. 3**	**St. 4**	**St. 5**	**St. 6**	**St. 7**
delay	0.2	0.2	0.9	0.9	0.9	0.3	0.2
AT_{usr}	0.5	0.7	0.2	0.2	0.3	0.6	0.4
QoS ind.	0.9	0.9	0.1	0.1	0.1	0.8	0.9
AIU_{usr}	0.7	0.6	0.6	0.6	0.8	0.7	0.6
AIS_{usr}	0.7	0.7	0.7	0.7	0.7	0.7	0.7
AII_{usr}	0.7	0.7	0.7	0.7	0.7	0.7	0.7
AMR_{usr}	0.4	0.2	0.2	0.2	0.4	0.3	0.6
AMI_{usr}	0.4	0.2	0.2	0.2	0.4	0.3	0.6

Figure 10 [1] shows a screenshot of the MT window that depicts the Authentication Strength in the above-mentioned steps. Note that the time scale shown in the screenshot does not correspond to the real timing between the steps. Real timing from step to step can be days or weeks. The threshold level is shown as a red line. If the Authentication Strength is below the threshold, the authentication process – controlled by the Adaptive Security Manager – will not continue until the strength moves above the threshold. This can be achieved using stronger authentication mechanisms, or the system alarm mode is over. The values of the core metrics associated with the above steps are shown in Table III. Note that the values are only informative and are not based on real measurements. All values have been normalized and scaled to the interval 0...1. Note that the correlation of different events, such as publish and subscribe information and meta-data, is required to distinguish between normal system peak loads and increased traffic due to a DoS or a Distributed DoS attack.

In practice, Authentication Strength is affected by a large number of dimensions that can be depicted by a hierarchy of sub-metrics for *AS*. The above example therefore only shows how, in principle, authentication monitoring based on metrics can be used. Moreover, all data cannot be obtained from the administration domain of the metrics users: data originating from the different stakeholders' part of the authentication process, such as Identity Provider, is needed.

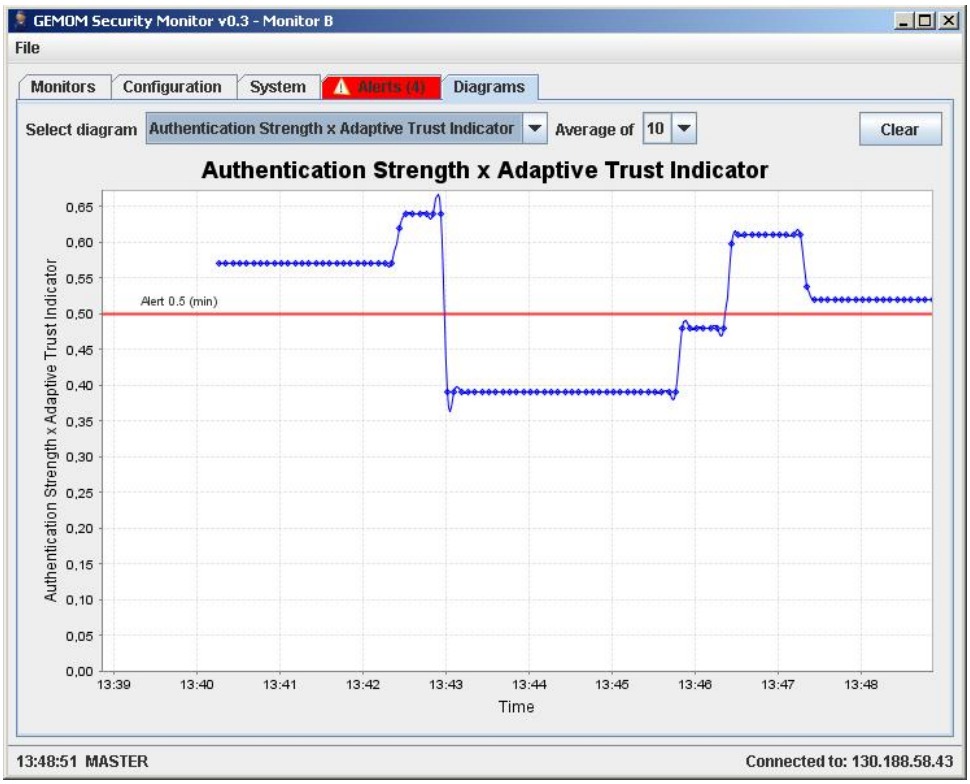

Figure 10. A screenshot of the GEMOM Monitoring Tool window depicting the seven steps of the authentication monitoring example [1]

IV. Proposed Security-Measurability-Enhancing Mechanisms

Although systematic and practical approaches to security monitoring based on metrics are generally desired, they are quite rare. One notable reason for this is that systems do not support security measurability very well. Measurability means that the metrics should be capable of having the dimensions, quantity, or capacity ascertained [14] in their measurement approaches [1].

The following section analyzes security-measurability-enhancing mechanisms that can be used to enable systematic and practical security monitoring in telecommunications and software-intensive systems. The mechanisms were introduced in our earlier work [1]. The mechanisms have been implemented in the research prototype of the GEMOM system [1].

A. *Flexible Communication Mechanism and/or Probing*

GEMOM nodes use the publish/subscribe mechanism to report their status and desired measurements to certain topics reserved for that purpose, while other authorized nodes can subscribe to this information. This mechanism is flexible, scalable, and increases the effectiveness of security measurements in the system. An example of how this mechanism can be used is presented in Figure 11 [1].

The performance of the actual system functions and communication can be measured and used for monitoring the design. For instance, a Broker can publish statistics of publish/subscribe activity, such as messages per second or kilobytes per second, or the delay between different nodes. An authorized node can subscribe to the associated topic or namespace. The monitoring system obtains measurement data from other nodes using the publish/subscribe mechanism. An indicator of monitoring delay, the relational 'distance' value of an MT part of the monitoring system, can be measured by comparing the delay data from Brokers. The delay data can be used for self-protection and resiliency management: if a Master MT crashes, an MT with the next highest priority automatically becomes a new Master MT. The prioritization can be partially based on the delay values of each MT.

If the publish/subscribe mechanism is not used, specialized *measurement probes* can be used to deliver measured data from the system components to the monitoring system. Different types of probes can be available or not available at different time instants and can be managed in a dynamic way. The probes should have a standardized language, yet be abstract, not related to any specific model. During monitoring, the measurement requirements have to be matched dynamically to the available and attainable probes that can deliver the required

measurement results [15]. The probes in use can reside in different parts of the system and in clients and servers.

The communication mechanism cannot fully solve the needs for gathering evidence from the platform resources, such as storage, memory, Input/Output (I/O) devices and network interfaces. Special measurement probes should be developed to manage this kind of evidence gathering.

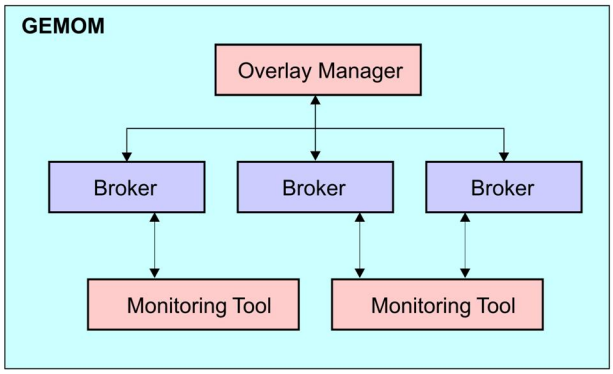

Figure 11. An example of using the publish/subscribe communication mechanism [1]

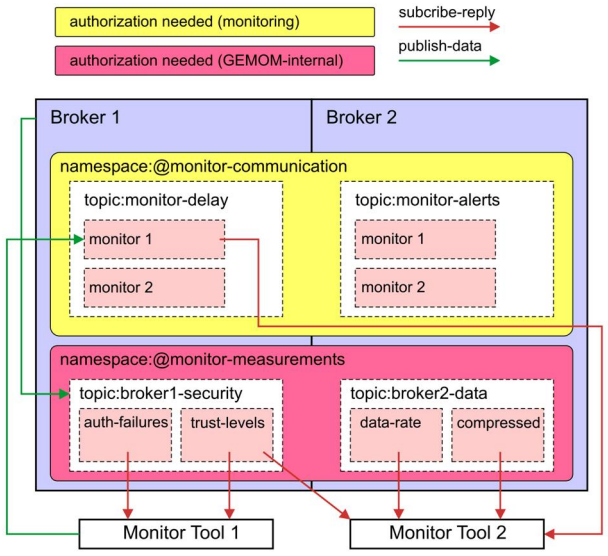

Figure 12. An example set-up of Monitoring Tools in an multi-point GEMOM monitoring environment [1]

It must be noted that the aggregation of individual security metrics results and measurements can be troublesome. Practical experience among industrial practitioners on aggregation has shown that the higher the abstraction level, the 'greener' the results tend to be, assuming a traffic lights approach ('red' meaning low level of security assurance or level, 'yellow' mediocre, and 'green' good) [15].

B. *Security Measurement Mirroring and Data Redundancy*

In general, GEMOM uses redundancy techniques to secure continued, uninterrupted operation in case of failures, overload, or a Denial-of-Service (DoS) attack [2]. As part of GEMOM's resilience management, redundant functionality, message feeds, and delivery paths will be maintained in the system in order to support switch-over to them in the event of a failure without information loss. Measurement data are also part of the redundancy functionality, and in addition to the nodes in operation, the measurement data reside in mirror nodes [1].

If a Broker crashes, valuable measured data about the failure event are available from its redundant counterpart, the *Mirror Broker* [1]. A Mirror Broker can be accessed separately for collection of data about the failure. Mirroring in security monitoring can be used for system security development purposes and learning from the 'what went wrong' evidence. A Mirror Broker and associated mirrored measurement data should use resources with no or only minor dependencies to the main Broker and measurements. If there are too many dependencies between the brokers, a failure in the main Broker can affect the Mirror Broker.

The effectiveness of mirroring can be increased by adding more than one Mirror Broker and/or mirrored data resources. In general, if the system contains N copies of the data, the resulting *redundancy level* is $N - 1$. The more mirrored functionality and data, the more processing resources are needed to carry out copying functionality associated with the mirroring procedure. Resources can be saved if smart mirroring procedures are used. For instance, mirroring can be done in a prioritized way in which most of the essential functionalities and/or data are kept up-to-date more frequently than less essential ones. Moreover, snapshots, raw presentations of the data, can be used.

C. *Multi-Point Monitoring*

A GEMOM subnet includes an Overlay Manager, n Brokers, and k MTs. An MT can monitor up to n Brokers (see Figure 12 [1]), which enables *multi-point monitoring*. In this subnet, one MT acts as a master and the other MTs act as slaves. The Master MT is responsible for synchronizing communication and data exchange between different MTs. While the status information of each MT is available for every MT, only the Master MT is allowed to manage others [1].

Figure 13 shows a GEMOM MT screenshot of the management of a Master and a Slave Monitor.

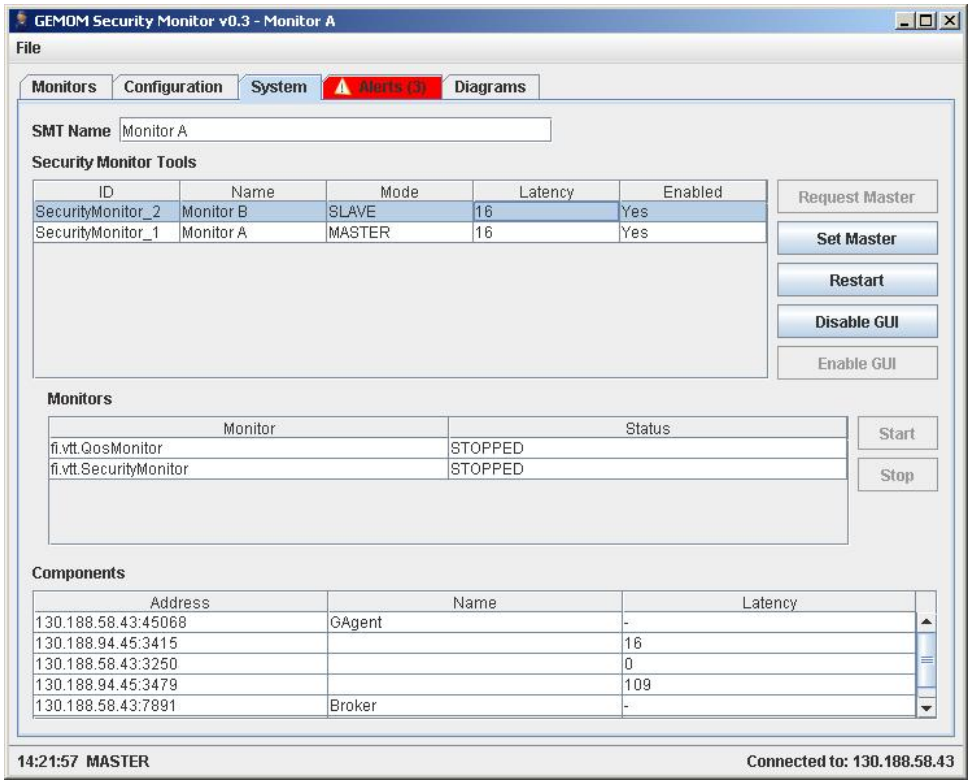

Figure 13. An MT window screenshot displaying a simple monitoring set-up for a Master Monitor with one Slave Monitor

In security-related measurements, multi-point monitoring is needed in most cases. Security issues in different parts of the overall system under investigation (e.g., a technical system, service, product) contribute to the overall security performance. Moreover, security evidence from stakeholders residing outside the administration scope of the system can be needed. For instance, the Identity Provider's identity establishment and management processes, and the client-server authentication protocols interfacing to it affect the Authentication Strength. Measured evidence is not always available and attainable from outside the scope of the system administration. In these cases, available or assessed evidence information can be configured directly to the Monitor modules. Such evidence is usually relatively static, and does not need to be updated frequently.

D. Auto-Recovery on Error

The MTs save their state information for later use in the case of a system crash due to a failure or attack to support flexible investigation of the situation. When the MT is rebooted, it activates auto-recovery functionality with the error mode and automatically initiates connection to the system. In this kind of situation, the crashed MT will always operate as a slave, because the error situation should be investigated before resuming normal operation, even if the crashed MT was a Master MT before the failure. If a new Master MT is started, it can easily disable the crashed and rebooted MT [1].

Auto-recovery functionality can be automated in different ways: (i) no auto-recovery, (ii) attempt auto-recovery for *N* times or *M* minutes, or (iii) attempt auto-recovery indefinitely. From a security monitoring perspective, the automation depends on the nature of the failure of the attack. It is important, especially after an attack situation, to investigate the situation and validate the monitoring configuration before, during, and after the attack in order to ensure that every MT plays its role as fairly as possible. Root-cause analysis techniques can be used to investigate the attack situation in detail based on the evidence provided by the MT.

Below is an example of an internal monitoring tool messaging and synchronization topic structure. The `/smon` namespace has two topics: `Watchdog` and `Control`. The `Watchdog` topic is updated by the Master MT and it is read by all Slave MTs. They should reply with the watchdog time to `/smon/smt` namespace under the `Alive` topic. If any Slave MT does not update its watchdog time, it is considered to be unavailable. The `Hierarchy` topic shows which monitor is playing the master role. In case the Master MT crashes, other monitors decide which will be the new master. The new master takes over the master role and starts controlling the

watchdog timer and the control messaging. Control messages are sent via the `Control` topic in the `/smon` namespace.

The `/smon/modules` namespace is used to inform other MTs of the kind of modules that are running in each monitor. The `/smon/metrics` namespace shows the actual metric configuration of all monitor modules.

```
/smon

  T: Watchdog
  Time=289

  T: Control
  from=SecurityMonitor_xx
  to=ALL|SecurityMonitor_xx
  time=12334214
  cmd=MASTER_MODE
      REQUEST_MASTER_MODE
      RESTART:<delay>
      ENABLE_GUI:true|false
      START_MONITOR:n@@xx.xxx.XxxXxxx
      STOP_MONITOR:m@@xx.xxx.XxxXxxx
      LASTID:xx


/smon/modules

    T: SecurityMonitor_0
    1@@fi.vtt.QoSMonitor=RUNNING
    2@@fi.vtt.AnomalyMonitor=STOPPED
    CurrentConfig=1,2

    T: SecurityMonitor_1
    1@@fi.vtt.ActivityMonitor=RUNNING
    CurrentConfig=1

    T: SecurityMonitor_2


/smon/smt

    T: Alive
    SecurityMonitor_0=289
    SecurityMonitor_1=289
    SecurityMonitor_2=289

    T: Names
    SecurityMonitor_0=Machine in Room E272

    T: Hierarchy
    SecurityMonitor_0=Master
    SecurityMonitor_1=Slave

    T: Latency
    SecurityMonitor_0=14
    SecurityMonitor_1=37


/smon/measurements

    /aMEM

      T: <ip>:<port>
      <timestamp_long>=<aMEM_average>

    /iCPU

      T: <ip>:<port>
      <timestamp_long>=<iCPU_average>


/smon/alerts

    /SecurityMonitor_0

      T: 1@@fi.vtt.QoSMonitor
      From=SecurityMonitor_0
      Discriminator=aMEM
      Status=NEW|OLD|ACK|DEL
      OldStatus=NEW|OLD|ACK
      Time=142412455
      Severity=CRITICAL|MAJOR|MINOR|WARNING
      Text=Something went terribly wrong!

      T: 2@@fi.vtt.ActivityMonitor

    /SecurityMonitor_1

      T: 1@@fi.vtt.QoSMonitor


/smon/metrics

    /SecurityMonitor_0

      T: fi.vtt.QoSMonitor.Broker
      MinimumAvailableMemory=335
      Type=Decimal
      Visibility=Public
```

E. Integrity, Availability and Configuration Correctness Checks

The integrity and availability metrics, part of the security metrics collection of GEMOM, are based on the results from the integrity and availability check functionality built into selected critical parts of the system. Integrity and availability checks are carried out by special program code constructs and algorithms at run-time and in connection with software security assurance activities (such as testing and analysis). In addition to code constructs, tools such as Tripwire [16] can be used for periodical and triggered integrity checks of files and data. The reports and potential alarms of integrity checks carried out by integrity check code constructs and tools are visible in the MT, and this evidence is used as part of the integrity metrics.

The integrity and availability checks address typical software weakness and vulnerability problems. The following widely known checks increase the security quality and can be used to support the security measurability [1]:

1. Validation of input data in all relevant interfaces. It is possible to prevent most injection attacks (such as Cross-Site Scripting, Structure Query Language SQL injection, and null injection) with by proper input validation.
2. Buffer and memory overflow checks
3. Storage and database checks
4. Error recovery, self-protection and resiliency checks

Configuration correctness is one of the main aims of integrity checks, along with high quality of software and a lack of critical vulnerabilities. Metrics depicting configuration correctness are based on design and implementation requirements, reference standards, and best practice information. The wrong configuration and deployment of security controls can result in severe security problems. Moreover, wrong system configuration in other parts than those directly linked to security functionalities can potentially turn into security problems.

Further checks should be developed, such as focusing on the issues pointed out by the results of the threat and vulnerability analysis of the SuI, and available public vulnerability information, such as OWASP [17] classifications and applications of them, e.g., [18].

It must be noted that it is possible to develop a wide collection of integrity and availability metrics, and the checks can degrade the processing performance greatly. Prioritization of the results of threat and vulnerability analysis is therefore important.

F. State of Security Framework

The timing of security-related measurements needs to be managed carefully. Risk assessment is essentially a prediction of security risks that can cause problems in the future. When the security controls that are implemented and deployed in the system are based on the output from risk assessment, it is important to handle history information, current measurements and future predictions separately. The MTs and the Adaptive Security Management functionality in the Overlay Manager use the State of Security (SoS) framework [5] (see Figure 14 [1]). This concept can be seen as a security-measurability-enhancing mechanism, implementing a timing framework for security measurements.

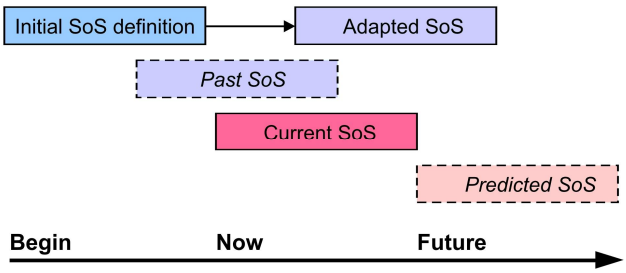

Figure 14. The high-level State of Security concept in GEMOM [1]

SoS is a time-dependent estimate of the system's security level and performance based on an appropriate integrated and aggregated balanced collection of security metrics. In short, SoS describes the high-level *security health index* of the operational GEMOM system. SoS contains an aggregated value of individual metrics components. The estimate is initially offered by the MT and whenever it is triggered. Metrics with different time scale properties (lagging, coincident, or leading) are used depending on the situation [1].

Note that valuable information is always lost in the aggregation process of metrics results. A graphical representation of metrics in the MT can therefore be used when investigating the SoS in more detail.

There are five different phases in the estimation of SoS [1]:

1. The *initial SoS* is calculated based on the collection of initial values of security metrics (lagging and coincident metrics). Weights are associated with different component metrics in order to indicate their relative importance, based on the results of risk-driven prioritization. In practice, a 'close to correct' weight assignment is used, as analytical results are often unavailable [19]. Moreover, practice has shown that the aggregated results tend to show results that are too optimistic compared with reality. The initial weights are assigned based on up-to-date threat, vulnerability, trust, and reputation knowledge. The metrics components that need to be balanced in the collection are adaptive security, authentication, authorization, confidentiality, integrity, availability, and non-repudiation.
2. The *current SoS* is based on a coincident metrics calculation whenever it is triggered by a timer, an attack alarm, an anomaly alarm, or a manual request. The weighting is adjusted based on updated data on threats, vulnerabilities, the context, or other relevant parameters.
3. Past and current SoSs can be *compared* in the monitoring system in order to identify trends and potential fault situations. Trend analysis is an important application of security monitoring results.
4. The SoS can be *adapted* according to decisions made by the Adaptive Security Management functionality in the Overlay Manager. The adapted SoS is the result of actions carried out by the Overlay Manager.
5. The *predicted SoS* is based on leading metrics to support proactive estimation. A comparison of past SoS can be used to identify trends affecting the prediction. The predicted SoS can be fed as input to the threat and vulnerability analysis.

Figure 15 depicts the general process from the Current SoS to the Adapted SoS. After the process, the old Current SoS becomes a Past SoS and the Adapted SoS becomes the Current SoS. The trigger mentioned in the figure can be an alarm, incident, or some other type of security effectiveness information. Trends and changes in the security risks affect the updates of metrics and their calculation.

G. Measurability Support from Building Blocks of Security

The architectural design of the system greatly affects the

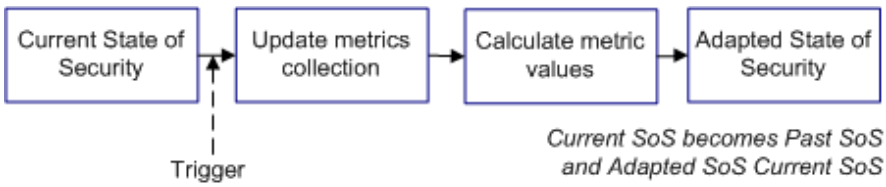

Figure 15. From Current SoS to Adapted SoS.

security measurability of the system during its operation. It is important that the various architectural security building blocks are designed to support seamless co-operation between the different parts of the operational system and its connections. In communication architectures, nodes or modules should be designed in such a way that it is possible to obtain enough data about their operational state in an authorized manner. Measurement probe architecture and the management of probes should be designed hand-in-hand with the actual system architecture. Stage 5 (design of measurement architecture), mentioned in the security metrics development approach discussed above, is tightly connected to the actual architecture of the system.

Several architectural components in GEMOM have been built in such a way that they exhibit internal properties and functionalities that support security measurements: in other words, they can be considered to incorporate *intrinsically security-measurable* [4] or security-measurability-enhancing constructs. These constructs increase the credibility, applicability, and sufficiency of monitoring approaches based on security metrics [1].

In addition to the architectural building blocks, systematic and automatic information exchange in GEMOM has been designed between the different stages of the security metrics development process: threat and vulnerability analysis, security requirements definition, decomposition of requirements, and detailed metrics definition [1].

H. Use of Shared Metrics and Measurement Repositories and Enumerations

Enumerations classify fundamental entities and concepts relevant to security, and repositories allow common, standardized content to be used and shared. Shared security metrics and measurement repositories support the development of credible security metrics and help to focus and prioritize efforts. In the near future, several novel shared security metrics and associated data repositories will be available. Examples of these include the Common Vulnerability Scoring System (CVSS) [20] and the associated baseline security data enumeration Common Vulnerabilities and Exposures (CVE) [21], both of which are part of the Security Content Automation Protocol (SCAP) [22]. Martin discussed their use in [23], along with how they integrate with different stages in the secure system development life cycle. In addition, the Web Application Security Consortium (WASC) and the SANS Institute maintain popular threat and vulnerability classification collections, found in [24] and [25], respectively [1]. Table IV recaptures examples of security enumerations, languages, and repositories [26].

I. Reuse of Available Metrics and Measures Relevant to Security

In many cases, parameters that were originally developed for other measurement purposes can be applied, at least partly, to security measurements. The following section provides some examples of these. Taking into account metrics that are readily available in the system is part of the action to increase security measurability. The available and attainable metrics relevant to security can be matched with the needs of security evidence.

TABLE IV. EXAMPLES OF SECURITY ENUMERATIONS AND REPOSITORIES [26]

Name	Explanation
Common Vulnerabilities and Exposures (CVE)	A standard enumeration of identifiers for known vulnerabilities
Common Weakness Enumeration (CWE)	A standard enumeration of identifiers for software weaknesses
Common Attack Pattern Enumeration and Classification (CAPEC)	A standard enumeration for identifiers for attacks
Common Configuration Enumeration (CCE)	A standard enumeration for identifiers for configuration
Common Platform Enumeration (CPE)	A standard enumeration for platforms, operating systems, and application packages
SANS Top-20	Top critical vulnerabilities list by the SANS Institute
OWASP Top-10	Top critical vulnerabilities list by OWASP
WASC Threat Classification	Threat classification by the WASC consortium
U.S. National Vulnerability Database (NVD)	A U.S. vulnerability database using, e.g., CVE, CCE, SCAP, and US-CERT data
Red Hat Repository	Patch definitiions for Red Hat Errata security advisories
OVAL Repository	OVAL vulnerability, compliance, inventory and patch definitions
Center for Internet Securtiy (CIS)	Security configuration benchmarks
U.S. Department of Defence Computer Emergency Response Team (DoD-CERT)	Information assurance vulnerability alerts ans security implementation guides
U.S. National Security Agency (NSA)	NSA security guides

TABLE V. MACHINE -RELATED COUNTERS [5]

Symbol	Counter
IT_{cpu}, PT_{cpu}	CPU idle time, CPU processing time
AM	Available memory
PA	Paging activity
B_{util}, B_{max}	Bandwidth usage, maximum bandwidth
L_{m2m}, V_{m2m}	Latency, visibility between two machines

TABLE VI. PUBLISH/SUBSCRIBE ACTIVITY-RELATED COUNTERS [5]

Symbol	Counter
PPN, PPT	Publications per namespace, topic
PPB, PPC	Publications per Broker, Client
MPB, MPC	Messages per Broker (publications and subscriptions), Client
DPB, DPC	Amount of data per Broker, Client
PBPC	Number of protocol breaches per Client
N_{ns}, N_{top}	Number of namespaces, topics

TABLE VII. SOURCES OF SECURITY METRICS IN GEMOM [5]

Symbol	1	2	3	4	5	6	7
Countermeasure mechanism performance metrics (by requirement decomposition)	×	×	×	×	×	×	×
Metrics based on anomaly and misuse models (attacker-oriented weakness metrics)		×	×				
Cryptographic strength metrics		×		×	×		×
Availability metrics based on QoS application performance metrics						×	
Trust metrics	×						
Reputation metrics	×						
Software quality metrics	×	×	×	×	×	×	×
Vulnerability metrics	×	×	×	×	×	×	×
System endemic security-relevant metrics						×	

QoS evidence can often be used for security-related availability measurement. Different types of availability threats usually have a high impact in telecommunication systems: the system, or a core part of it, is in its most vulnerable state during availability attacks, such as Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks. Moreover, the actual execution of the application often crashes due to these attacks. An attacker can execute his/her strategy and achieve his/her goals by exploiting the high vulnerability time window, which can potentially also cause other security threats. The intruder can even seize the system using this strategy [27]. Until recently, QoS and security metrics have lived in separate worlds. Some security attacks affect application performance, and the most important objective of QoS management is to ensure application performance. The GEMOM monitoring approach uses metrics for security availability and application performance measurement [6]. Other availability parameters that can usually be obtained from telecommunication systems are delay, delay variation, and packet loss rate [27] [1].

Machine-related and functionality-related counters can be used in security measurements. Table V [5] lists some general machine-related counters that are available in the GEMOM system. Similar counters are available in most systems. Aggregated information from several counters can be used to identify system situations that could potentially be a threat to security. Table VI [5] lists some publish/subscribe activity-related counters that can be used when monitoring abnormal situations. Table VII [5] summarizes the origin of security metrics and other security-relevant metrics in GEMOM. The numbers in the columns represent the security-enforcing mechanisms (1 = adaptive security, 2 = authentication, 3 = authorization, 4 = confidentiality, 5 = integrity, 6 = availability and 7 = non-repudiation). '×' in the relevant box indicates the use of the security metrics in GEMOM [1]. Note that in another types of systems, different types of security metrics can be used, and their origin can be different.

J. Secure Coding – Security-Relevant Software Quality

Software quality affects the resulting security level of the system directly and indirectly. Software quality can be increased by the application of secure coding principles. In addition to input validation, secure coding principles include, for example [28], the defense-in-depth principle, the principle of least privilege, aiming for the simplest system possible, default access denial, data sanitization, effective quality assurance, and use of a secure coding standard (best practice).

It must be noted that some of the secure coding principles seem to decrease the security measurability. For example, default access denial, the principle of least privilege, and data sanitization may have this kind of effect due to the reduced amount of available information. Proper authentication and authorization should be implemented in order to allow the security measurement activity to obtain all relevant data.

Security metrics that address the enforcement of security coding principles are part of the balanced security metrics collection. Knowledge about the principles used increases the security measurability of the system.

V. IMPACT ON FEASIBILITY OF SECURITY MONITORING

The feasibility of security measurement and associated monitoring is at the core of its success. Our earlier work investigated the criteria for the quality and feasibility of using security metrics in software-intensive systems [29] and the core requirements for a secure and adaptive distributed monitoring approach [30]. The following section discusses the impact that the mechanisms introduced in this study have on the feasibility of security monitoring based on security metrics. The metrics criteria of [28] and the monitoring requirements of [30] are deployed as feasibility objectives [1].

The criteria in [29] are classified into three levels, each of which incorporates six criteria. The levels emphasize the credibility of security metrics, their applicability for use together with the measurement approach, their sufficiency for their intended use, and the completeness of the metrics collection. The criteria were originally developed for security metrics. The applicability of the security metrics and measurement approaches, such as monitoring and applying them to the final use environment, is crucial however. Security metrics should be designed 'hand-in-hand' with the measurement approach: metrics cannot be used without measurements, and measurements are useless unless they are interpreted [29].

Table VIII [1] summarizes the feasibility criteria discussed in [29] and provides examples of how they are supported by the security-measurability-enhancing mechanisms of the GEMOM monitoring approach. In the table, FC denotes flexible communication, SMM denotes security measurement mirroring and data redundancy, ARE indicates auto-recovery on error, MPM is multi-point monitoring, IAC represents integrity and availability checks, TF is the timing framework, BBS is the measurability support of building blocks of security, SMMR is the use of

shared metrics and measurement repositories, RAM is the reuse of available metrics and measurements relevant to security, and SC indicates secure coding. In general, the development of security metrics is challenging and metrics that meet all the feasibility criteria are extremely rare [29].

TABLE VIII. FEASIBILITY CRITERIA OF [29] AND DISCUSSED MECHANISMS SUPPORTING THEM [1]

Criterion	Supporting Mechanism(s)
Correctness	IAC, SC
Granularity	All
Objectivity and unbiasedness	TF, SMMR, RAM
Controllability	All
Time-dependability	MPM, ARE, TF
Comparability	FC, MPM, ARE, IAC, SMMR, SC
Measurability	All
Attainability, availability, easiness	All
Reproducibility, repeatability, scale reliability	All
Cost effectiveness	FC, BBS, SMMR, RAM
Scalability and portability	FC, SMM, MPM, BBS, RAM, SC
Non-intrusiveness	FC, SMM, MPM, ARE, IAC, BBS, RAM, SC
Meaningfulness	MPM, TF, BBS, SMMR, RAM
Effectiveness	MPM, ARE, TF, BBS, SMMR, RAM
Efficiency	FC, SMM, MPM, BBS, RAM, SC
Representativeness and contextual specificity	TF, BBS, SMMR, RAM
Clarity and succinctness	MPM, IAC, BBS
Ability to show progression	TF, SMMR
Completeness	FC, TF, BBS, SMMR, RAM

TABLE IX. ADDITIONAL MONITORING REQUIREMENTS OF [30] AND DISCUSSED MECHANISMS THAT SUPPORT THEM [1]

Requirement	Supporting Mechanism(s)
Security	All
Runtime adaptation	FC, SMM, MPM, ARE, BBS, RAM

The requirements identified in [30] can be classified into five categories: scalability, runtime adaptation, correctness, non-intrusiveness, and security. Table IX [1] analyzes how the proposed mechanisms support the requirements that are in addition to [29]: security and runtime adaptation.

Scalability is a key property in any distributed monitoring system. Real applications and systems are often quite dynamic and the scale of use can change rapidly. Scalability refers not only to scaling from a small system to a large one but also to scaling from large to small, and to scaling in geographic coverage [31]. Scalability goals can also concern the use of a large number of applications in the same system. All scalability goals imply the need for flexibility in terms of how the monitoring tool and 'measurer nodes,' and their communication, are constructed. GEMOM basic solutions, which comprise the MOM approach and the inherently encapsulated publish/subscribe communication mechanism, support scalability well [1]. More effort to ensure scalability is needed in other kinds of measurement architectures with complex probe management.

If necessary, the hierarchical architecture of Monitors and MTs can be created to support a large number of measurements. It is easy and straightforward to configure and initiate new Monitors. In principle, the chosen solutions do not set constraints on the scale of the system, in terms of the number of nodes or from a geographic coverage perspective. If high-volume measurements are needed, however, a special type of measurement channel solution is required.

The performance of the overall monitoring system is constrained by the underlying network solutions, typically the Internet networks in use. Network management is a problem for all networks as they grow in size. It may also be challenging to use a huge number of topics or namespaces, with proper authentication and authorization management. The number of measurement topics and namespaces can be kept under control with informed and adaptive planning and configuration. A large number of overlapping publish and/or subscribe actions result in a need for sophisticated mutual exclusion management solutions. Chockler et al. [32] suggested techniques for scalable solutions for publish/subscribe activity with many topics [1].

As GEMOM and its applications are run on various platforms that range from desktop computers to smart mobile phones, the portability of the monitoring solutions is important. The underlying MOM middleware solution enables applications to establish communication and interaction without having knowledge of the platform on which the application resides within the network. The monitoring approach is built using the same communication approach as the system in general. This means that the portability of the monitoring approach is good; only appropriate interfaces between the GEMOM nodes and their repositories and the services of the platform need to be implemented [1].

Interoperability of the GEMOM monitoring solution with other security tools is desirable in order to obtain a more holistic 'security picture' of the environment. The target of security monitoring can be applications, hosts, and the network. The GEMOM monitoring solution emphasizes those applications that operate in the GEMOM system, as well as host-based performance, resilience, and self-protection information [1].

Tools that focus on Internet-based traffic include vulnerability scanners, packet sniffers, various kinds of traffic analyzer tools, and application-specific scanners. A variety of these tools is available as commercial and open products. These tools can be connected to the MT in a straightforward manner using the publish/subscribe

mechanism. They often do not support interpretation of the data or correlation of input or logs from different sources however. In other words, they concentrate on the raw measurements. Almost all of the tools in the area of Internet Protocol (IP) traffic measurement and analysis perform only a small subset of the functionalities required to capture, file and classify, store, analyze traffic, and prepare the results for graphical display or for export into a database or other framework [33]. Consequently, the seamless integration of these tools to the GEMOM monitoring approach requires the development and configuration of specific Monitors that are capable of interpreting and correlating data produced by the tools [1].

The Open Web Application Security Project (OWASP) [17] and Insecure.org [34] list and analyze different security tools, some of which are potentially useful for complementing the GEMOM adaptive security management environment.

VI. Related Work

The US National Institute of Standards and Technology's (NIST) security metrics report [4] believes that the development of security-measurability-enhancing mechanisms is a promising research direction. Few research results are available in this new field however. In the following, we discuss related research that suggests mechanism for enhancing security measurability.

Martin [23] focused on the issues highlighted in Section III.H of this paper in his discussion on security-measurability-enhancing mechanisms based on SCAP enumerations and scoring systems. He elaborated on the mechanisms from an organizational perspective and showed how these mechanisms can be integrated into risk management, operations security management processes and enterprise networks. He does not deal with security-measurability-enhancing mechanisms from a technical perspective however.

Ciszkowski et al. [35] described an end-to-end quality and security monitoring approach for a Voice-over-Internet Protocol (VoIP) service over Peer-to-Peer (P2P) networks, providing adaptive QoS and DoS/Distributed DoS attack detection. They also introduced an associated trust and reputation framework to support routing decisions. The standalone modules in this architecture include Security, QoS, Reputation Management, and Intrusion Detection functionalities, the communication of which is arranged via channels. These choices do not allow as much flexibility and scalability as the GEMOM dynamical monitoring.

Jean et al. introduced a distributed and adaptive security monitoring system based on agent collaboration in [36], using a special algorithm to classify malicious agents based on their execution patterns. They also used the notion of an agent's security level but did not provide further details of the parameters of the security level calculation. They also defined a trust and confidence notion for the hosts of the agents.

Spanoudakis et al. [37] introduced a runtime security monitoring system based on confidentiality, integrity, and availability patterns. Their architecture contains a Monitoring Manager that takes requirements as an input and controls Monitoring Engine. Measurement Agents deliver the measured data to an Event Catcher, communicating with the Monitoring Engine. This architecture is interesting because it can be mapped directly to the more flexible GEMOM monitoring architecture. A special monitoring pattern language is used to define the pattern metrics for the Monitoring Manager. This work is limited to the basic abstract dimensions of security – confidentiality, integrity, and availability.

Kanstrén and Savola described five-layer reference architecture for a secure and adaptive distributed monitoring framework in [28], as developed in the BUGYO Beyond Research project. The main approach of this reference architecture is to increase non-intrusiveness by isolating the monitoring framework from the observed system. The architecture uses abstraction layers for the management of different practical measurement management objectives. In systems using measurement probes, this kind of architecture is usual, but for publish/subscribe-based systems like GEMOM, it can bring too much complexity.

Evesti et al. [38] proposed a preliminary environment for runtime security management that consists of service discovery, service adaptation, measurement, and decision-making functionalities. Vulnerability databases are used as a basis for measurements. The decision-making functionality can be seen as carrying out functions that are similar to those that the Security Measurement Manager and Overlay Manager perform in GEMOM.

Bejtlich [39] discussed network security monitoring at length and overviewed some tools and research solutions used, especially in traffic and packet monitoring. If the GEMOM environment is used over the Internet Protocol (IP), these tools can complement the GEMOM security monitoring environment by offering more details of IP traffic. Bejtlich claimed that traditional Intrusion Detection Systems (IDSs) do not deliver the value promised by vendors and that detection techniques that view the alert as the end goal are doomed to fail. Like GEMOM's monitoring and adaptive security management approaches, Bejtlich views alert data as an indicator and as the beginning of the actual decision-making process. Intrusion Detection and Prevention Systems (IDS/IPS) must also adapt to changes in the threat and vulnerability environment.

Bulut et al. present a measurement framework for security assurance in [40]. The framework collects information about the security-enforcing mechanisms of the system under investigation. The framework contains three different types of components: probe-agents, multiplexing agents and server-agents. Multiplexing agents are responsible for multiplexing data over subnet boundaries, and server-agents handle centralized processing of the measured data.

Vandelli et al. present a measurement framework for scientific experiments in [41] that are capable of capturing massive amounts of data. Standardized architectures and protocols are used between the measurement components. A separate data stream is used to pass high-volume data to the measurement system, and a separate channel is used to pass

control requests. This kind of high-performance measurability-enhancing solutions can be used in data gathering of security-relevant log information. In most security measurements, high-volume data do not need to be transferred.

A survey of approaches to adaptive application security and adaptive middleware can be found in [42] and [43], respectively.

VII. DISCUSSION

Security is a multi-faceted socio-technical challenge, and despite the long record of academic research and wide experience of practical issues, it still suffers from systematic and widely accepted design and management techniques. Security measurements introduce systematic thinking to security issues. However, nowadays, the use of security metrics and measurements is hard due to the lack of information and enough support from developed systems. It is clearly a 'chicken or egg' problem: in order to advance the field of security metrics, you need evidence from the actual system, and in order to be able to gather that evidence (and justify the gathering effort), you need metrics. Security-measurability-enhancing mechanisms aim to make evidence gathering effective and cost-effective, and they contribute especially to improved availability and attainability of the evidence. If these mechanisms become more widely accepted and applied, the field of security metrics can definitely make big steps forward. The fact that public/subscribe communication was chosen as the underlying communication paradigm is an important design choice supporting the measurability goals well. Flexibility in measurements is needed to 'pave the path' for wider acceptance of the use of security metrics and measurements.

We emphasize that the collection of security-measurability-enhancing mechanisms discussed in this study is preliminary. It is obvious that specific security measurement needs will raise the need for new and different kinds of mechanisms. In the future, it makes sense to also carry out standardization efforts in this field; it is easier to support the wide acceptance of the mechanisms if there are commonly agreed ways to do so. Recent advances in information security management system (ISO/IEC 27000 series) standardization have shown that there is interest in incorporating security metrics and measurements in security management standards. The current work in a standardization world regarding security metrics is quite vague. Nowadays, there are still big gaps between security management and engineering-oriented standards, the former concentrating on a top-down approach, and the latter mainly on bottom-up check-lists. Metrics, with the help of enough support of security-measurability-enhancing mechanisms, can play the role of bridging this gap. Wide acceptance of security metrics and measurement approaches can make remarkable advances in the whole security field.

VIII. CONCLUSIONS AND FUTURE WORK

We have discussed solutions to enhance the security measurability of telecommunications and software-intensive systems. The presented solutions have most value if they are built into the system under investigation already during the architectural design of it. The solutions were discussed in the context of the distributed messaging system GEMOM incorporating adaptive security management functionality. Solutions proposed and discussed in this study:

- *A flexible communication mechanism* is crucial to the security measurements. In GEMOM, the use of the publish/subscribe mechanism for measurement reporting increases the flexibility, scalability, and effectiveness of security measurements. In other types of communication architectures, measurement probes should be designed hand-in-hand with the architectural choices of the system.
- *Data and functionality redundancy* can help in obtaining valuable security-relevant data from fault situations. Redundancy can be implemented by smart maintenance of Mirror Brokers and mirrored measurement data, as independently as possible from the main Brokers and measurement data.
- *Multi-point measurement support* is often needed in security measurements, as only holistic security evidence is meaningful and the functionalities and processes of various system components and stakeholders contribute to that evidence. Holistic security measurability can be implemented by deploying several monitoring tools that are able to communicate continuously with each other.
- *Auto-recovery on error mode* helps in the investigation of failure before resuming normal operation. It is a helpful functionality for deeper analysis of attacks, such as root-cause analysis.
- *Configuration correctness, integrity and availability checks* of software constructs and input, buffers, memory, storage, and databases are integral parts of the security measurability solution.
- *State of security framework* supports categorization of the metrics and measurements with respect to time. Timing of different measurements and metrics is also important; it is necessary to define the state of security in order to establish proper time-dependency for the measured information.
- *Seamless co-operation of the security building blocks* at system architectural and process level should include a systematic and automatic information exchange. In order to ensure enough security measurability support, these issues should be taken into account already during the architectural design phase of the system and during the risk management and security assurance activities.
- The use of *shared metrics and measurement repositories and enumerations* supports the development of security metrics and ensures that the monitoring system has up-to-date threat and vulnerability knowledge. Security effectiveness metrics, in particular, can be based on repositories and enumerations.
- *The use of measures developed for other measurements* can be useful for security measurements. Some parameters that were originally developed for other measurement purposes can be applied to security

measurements. These include QoS, performance indicators, delay, delay variation, and packet loss rate, along with other indicators that reflect abnormal operation. Adequate authentication and authorization mechanisms ensure that the necessary data are available from modules for which secure coding principles are thoroughly enforced.

Our future work includes using the monitoring system for adaptive security management in experimental GEMOM system application use case investigations in critical information systems. This experimentation work will analyze the feasibility of the monitoring approach, the security metrics that are used, and the identified security-measurability-enhancing mechanisms.

ACKNOWLEDGMENTS

The main part of the work presented in this study was carried out in the GEMOM FP7 research project (2008–2010), which was partly funded by the European Commission. Some of the analytical work in study was carried out in the BUGYO Beyond CELTIC Eureka project (2008–2011). The authors acknowledge the contributions to the GEMOM system description made by various GEMOM partners, and collaboration with persons working in the BUGYO Beyond project.

REFERENCES

[1] R. Savola and P. Heinonen, "Security-Measurability-Enhancing Mechanisms for a Distributed Adaptive Security Monitoring System," SECURWARE 2010, Venice/Mestre, Italy, July 18-25, 2010, pp. 25-34.

[2] H. Abie, I. Dattani, M. Novkovic, J. Bigham, S. Topham, and R. Savola, "GEMOM – Significant and Measurable Progess Beyond the State of the Art," ICSNC 2008, Sliema, Malta, Oct. 26–31, 2008, pp. 191–196.

[3] H. Abie, R. Savola, and I. Dattani, "Robust, Secure, Self-Adaptive and Resilient Messaging Middleware for Business Critical Systems," ADAPTIVE 2009, Athens/Glyfada, Greece, Nov. 15–20, 2009, pp. 153–160.

[4] W. Jansen, "Directions in Security Metrics Research," U.S. National Institute of Standards and Technology (NIST), NISTIR 7564, Apr. 2009, 21 p.

[5] R. Savola and H. Abie, "Development of Security Metrics for a Distributed Messaging System," AICT 2009, Baku, Azerbaijan, Oct. 14–16, 2009, 6 p.

[6] 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.

[7] R. Savola and H. Abie, "Development of Measurable Security for a Distributed Messaging System," International Journal on Advances in Security, Vol. 2, No. 4, 2009, pp. 358–380 (March 2010).

[8] G. Jelen, "SSE-CMM Security Metrics," NIST and CSSPAB Workshop, Washington, D.C., USA, June 2000.

[9] R. Savola and H. Abie, "Identification of Basic Measurable Security Components for a Distributed Messaging System," SECURWARE '09, Athens/Glyfada, Greece, Jun. 18–23, 2009, pp. 121–128.

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

[11] A. Jaquith, "Security Metrics: Replacing Fear, Uncertainty and Doubt," Addison-Wesley, 2007.

[12] N. Bartol, B. Bates, K. M. Goertzel, and T. Winograd, "Measuring Cyber Security and Information Assurance: a State-of-the-Art Report," Inf. Assurance Tech. Analysis Center IATAC, May 2009.

[13] 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.

[14] J. R. Williams and G. F. Jelen, "A Framework for Reasoning about Assurance," Arca Systems, Inc., 1998, 43 p.

[15] T. Kanstrén et al., "Towards an Abstraction Layer for Security Assurance Measurements (Invited Paper)," MeSSA '10, Proc. 4th European Conf. on Software Architecture: Companion Volume, pp.189-196.

[16] G. H. Kim and E. H. Spafford, "The Design and Implementation of Tripwire: a System Integrity Checker," Computer and Communications Security, Fairfax, VA, Nov. 2–4, 1994, pp. 18-29.

[17] OWASP, Open Web Application Security Project. [On-line]. Available: www.owasp.org [Accessed June 20, 2011].

[18] E. A. Nichols and G. Peterson, "A Metrics Framework to Drive Application Security Improvement," IEEE Security & Privacy, Mar./Apr. 2007, pp. 88–91.

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

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

[21] R. A. Martin, "Managing Vulnerabilities in Networked Systems," IEEE Computer Society Computer Magazine, Vol. 34, No. 11, Nov. 2001.

[22] M. Barrett, C. Johnson, P. Mell, S. Quinn, and K. Scarfone, "Guide to Adopting and Using the Security Content Automation Protocol (SCAP)," NIST Special Publication 800-177 (Draft), U.S. National Institute of Standards and Technology, 2009.

[23] R. A. Martin, "Making Security Measurable and Manageable," MILCOM '08, Nov. 16–19, 2008, pp. 1–9.

[24] Web Application Security Consortium (WASC), "Threat Classification," Version 2.0. [Online]. Available: www.webappsec.org [Accessed June 20, 2011].

[25] SANS Institute, "The Top Cyber Security Security Risks." [Online]. Available: www.sans.org/top-cyber-security-risks/ [Accessed June 20, 2011].

[26] R. A. Martin, "Making Security Measurable and Manageable," MILCOM '08, San Diego, California, Nov. 16-19, 2008, 9 p.

[27] R. Savola and T. Frantti, "Core Security Parameters for VoIP in Ad Hoc Networks," WPMC '09, Sendai, Japan, Sep. 7–10, 2009, 5 p.

[28] R. Seacord, "CERT Top 10 Secure Coding Practices." [Online]. Available: www.securecoding.cert.org [Accessed June 20, 2011].

[29] 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.

[30] T. Kanstrén and R. Savola, "Definition of Core Requirements and a Reference Architecture for a Dependable, Secure and Adaptive Distributed Monitoring Framework," DEPEND 2010, Venice, Italy, July 18–25, 2010, 10 p.

[31] P. Venables, "Security Monitoring in Heterogeneous Globally Distributed Environments," Information Security Technical Report, Vol. 3, No. 4, 1998, pp. 15-31.

[32] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Constructing Scalable Overlays for Pub-Sub with Many Topics – Problems, Algorithms and Evaluation," PODC '07, Portland, Oregon, USA, 2007, pp. 109–118.

[33] J. Quittek, A. Bulanza, S. Zander, C. Schmoll, M. Kundt, E. Boschi, and J. Sliwinski, "MOME Project – State of Interoperability," MOME Project WP1 Deliverable 11, Technical Report, 2006.

[34] Insecure.org [On-line]. Available: www.insecure.org [Accessed June 20, 2011].

[35] T. Ciszkowski, C. Eliasson, M. Fiedler, Z. Kotulski, R. Lupu, and W. Mazurczyk, "SecMon: End-to-End Quality and Security Monitoring System," 7th Int. Conf. on Computer Science, Research and Applications (IBIZA '08), Kazimierz Dolny, Poland, Jan. 31–Feb. 2, 2008. Published in Annales UMCS, Informatica, AI 8, pp. 186–201.

[36] E. Jean, Y. Jiao, A. R. Hurson, and T. E. Potok, "Boosting-based Distributed and Adaptive Security-Monitoring through Agent Collaboration," IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology 2007 – Workshops, pp. 516–520.

[37] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, "Towards Security Monitoring Patterns," ACM Sympos. on Applied Computing, Seoul, Korea, 2007, pp. 1518–1525.

[38] A. Evesti, E. Ovaska, and R. Savola, "From Security Modelling to Run-time Security Monitoring," SEC-MDA, Enschede, Netherlands, Jun. 24, 2009, pp. 33–41.

[39] R. Bejtlich, "The Tao of Network Security Monitoring – Beyond Intrusion Detection," Addison-Wesley, 2004, 798 p.

[40] E. Bulut, D. Khadraoui, and B. Marquet, "Multi-Agent Based Security Assurance Monitoring System for Telecommunication Infrastructures," Proc. Communication, Network and Information Security, 2007.

[41] W. Vandelli et al., "Strategies and Tools for ATLAS Online Monitoring," IEEE Transactions on Nuclear Science, Vol. 54, No. 3, pp. 609-617, June 2007.

[42] A. Elkhodary and J. Whittle, "A Survey of Approaches to Adaptive Application Security," SEAMS '07, May 20–25, 2007.

[43] S. Sadjadi, "A Survey of Adaptive Middleware," Technical Report MSU-CSE-03-35, Michigan State University, East Lansing, Michigan, Dec. 2003.