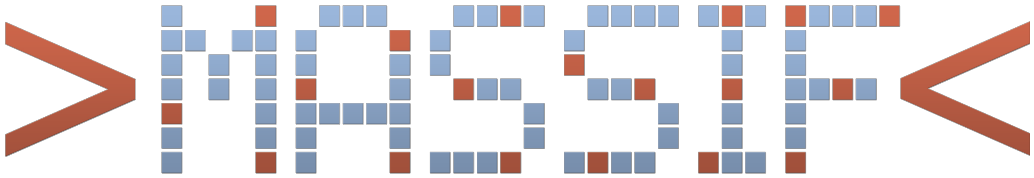

MAnagement of Security information and events in Service InFrastructures

MASSIF
FP7-257475

D4.1.2 - Multi-level Abstraction Concept

Activity	A4	Workpackage	WP4.1
Due Date	Month 24	Submission Date	2012-10-15
Main Author(s)	Maria Zhdanova (Fraunhofer) Jürgen Repp, Roland Rieke (Fraunhofer)		
Version	v1.0	Status	Final
Dissemination Level	PU	Nature	R
Keywords			
Reviewers	Jouni Viinikka (6cure) Luigi Coppolino (Epsilon)		

Part of the Seventh
Framework Programme
Funded by the EC - DG INFSO

Version history

Rev	Date	Author	Comments
V0.1	2012-06-04	Roland Rieke (SIT)	initial version
V0.2	2012-10-04	Maria Zhdanova (SIT)	draft version
V0.9	2012-10-09	Maria Zhdanova (SIT)	review version
V1.0	2011-10-15	Elsa Prieto (Atos)	final review and official delivery

Glossary of Acronyms

ADAS	Automated Data Acquisition System
ADM	Architecture Development Method
AMS	Asset Management System
APA	Asynchronous Product Automaton
BPMN	Business Process Modeling Notation
CEP	Complex Event Processing
CMDB	Configuration Management Database
CySeMoL	Cyber Security Modeling Language
DCS	Distributed Control System
DoS	Denial of Service
DoW	Description of Work
DSR	Decision Support and Reaction
EA	Enterprise Architecture
EPA	Event Processing Agent
EPN	Event Processing Network
ERP	Enterprise Resource Planning
GET	Generic Event Translation
GUI	Graphical User Interface
HMI	Human Machine Interface
IDS	Intrusion Detection System
IF-MAP	Infrastructure-Metadata Access Point
IMS	Identity Management System
IM	Indicator Meta-Model
IT	Information Technology
LAN	Local Area Network
MASSIF	MAnagement of Security information and events in Service InFrastructures
MCU	Monitoring and Control Unit

MTU	Master Terminal Unit
NIST	National Institute of Standard and Technologies
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation
OrBAC	Organization-based Access Control
OSSIM	Open Source Security Information Management
PLC	Programmable Logic Controller
PSA	Predictive Security Analyser
RFID	Radio Frequency Identification
RMU	Remote Monitoring Unit
RTU	Remote Terminal Unit
SABSA	Sherwood Applied Business Security Architecture
SAMEA	Security-Augmented Multilevel Enterprise Architecture Meta-Model
SCADA	Supervisory Control and Data Acquisition
SeSA	Security Strategy Agent
SIEM	Security Information and Event Management
SSPC	Security Strategy Processing Component
TOGAF	The Open Group Architectural Framework
UML	Unified Modeling Language
VPN	Virtual Private Network
WAN	Wide Area Network
XML	eXtensible Markup Language

Executive Summary

Security Information and Event Management (SIEM) technology provides a centralized viewpoint for security-related information. Current SIEM systems demonstrate extensive event monitoring capabilities that enable proactive security incident management, automated log management and compliance reporting. Being able to interface large diversity of event sources, they can collect, store and analyze various security events, but almost purely restricted to the level of Information Technology (IT) infrastructure making it practically impossible to estimate effects of detected incidents for business processes consuming IT services. SIEM solutions maintain and utilize simple contextual information about users and assets within an enterprise for the purposes of risk calculation, security events prioritization and incident investigation, at the same time, the representation of security requirements is scarce and does not allow to monitor and validate critical system properties. The lack of mechanisms for multi-level security monitoring and interpretation of security-related events on business impact view poses a serious problem to current SIEM technology.

In order to enable SIEM systems to relate events received from sources that belong to different abstraction levels and application domains and derive information relevant from security perspective multi-level abstraction techniques and methodologies are required. This work proposes Security-Augmented Multilevel Enterprise Architecture Meta-Model (SAMEA), a flexible concept of interrelated abstraction levels for the various types of security-related events as a conceptual basis for multi-level security monitoring and escalation of low level alerts to human-understandable warnings associated with critical business assets and security properties. Once implemented, the solution will allow to infer valuable statements about potential threats to the monitored system from a business perspective, as well as detect and predict security issues for business processes using security information from application and technology layers.

Contents

List of Figures

List of Tables

1 Introduction

Security Information and Event Management (SIEM) systems are combined solutions that perform real-time monitoring of security events. According to Gartner's[1] Magic Quadrant, SIEM deployments address three main scenarios: compliance, threat management and incident response [24]. Therefore real-time collection, analysis and storage of events from multiple sources in relation to contextual information about users and assets are the key system requirements of this technology.

Existing SIEM solutions offer wide reporting capabilities and extended compatibility to capture large diversity of security alerts generated by network hardware, security appliances, systems and applications. But the scope of current SIEM systems is almost purely restricted to the level of IT infrastructure. SIEM solutions exhibit some intelligence features that are enforced by a so-called correlation engine and usually depend on rules that the system uses. For example, the correlation engine of Open Source Security Information Management (OSSIM) from AlienVault[2] performs complex situation analysis combining four levels of correlation (attack, vulnerability, network status and service availability, and inventory correlation) with risk assessment for critical IT assets [20]. To some degree correlation rules make it possible to link observed events to security requirements and business processes running on top of the IT infrastructure, by grouping and filtering events based on given criteria. However, OSSIM and other SIEM systems currently provide insufficient mechanisms for multi-level security monitoring and representation of security-related events on service or business impact view, leaving the correlation and interpretation of incidents detected throughout multiple layers of the system hierarchy beyond the scope of existing SIEM technologies.

SIEM experts' opinions confirm the necessity to bridge this gap. They point out the growing demand to pull additional events from sources like Enterprise Resource Planning (ERP) systems, that are not directly related to security and thus usually not supported by SIEM vendors [23]. Supplementary data can substantially improve risk calculation, security events prioritization and incident investigation. For instance, SIEMs currently use correlation of security events with vulnerability and assets' value data to identify critical events. The integration of SIEM solutions with other systems such as Identity Management Systems (IMSs), Asset Management Systems (AMSs) and Configuration Management Databases (CMDBs) would predictably enhance their function, since information about the monitored environment available in these systems can provide context for interpretation of security events and increases awareness [22]. It also can help to single out false positives caused by legitimate changes in the infrastructure.

In order to be able to relate events received from additional sources belonging to different abstraction layers and application domains and derive information relevant from security perspective suitable abstraction techniques and methodologies are critical. The aim of this work is to develop a flexible concept of interrelated abstraction levels for the various types of security-related events as a conceptual

[1] http://www.gartner.com/
[2] http://www.alienvault.com/

basis for multi-level security monitoring and the escalation of low level alerts to higher structural levels. Once implemented, the solution will allow to detect and predict security problems for complete business processes using security information from application components and infrastructure, and increase the efficiency and accuracy of security analysis and control [30].

1.1 Purpose & Scope

The document describes a specification of interrelated abstraction levels for various types of security-related events that has been developed in the work package 4.1 of the "MAnagement of Security information and events in Service InFrastructures (MASSIF)" project [25]. In particular, this deliverable introduces the Security-Augmented Multilevel Enterprise Architecture Meta-Model (SAMEA), a meta-model that defines relations between different abstraction levels and different domains and thus enables escalation and interpretation of security-related events from lower to higher levels, e.g. from Denial of Service (DoS) attacks on technical infrastructure that can impair business processes due to violation of security requirements. This meta-model provides a method to transform multiple specific events into few human-understandable alarms associated with previously defined critical assets and security properties, and allows to infer valuable statements about potential threats to the monitored system from a business perspective. Moreover security mechanisms implemented on separate abstraction levels can benefit from mutual exchange of security state information for increasing efficiency and accuracy of incident detection even in situations when individual security monitors produce incomplete or uncertain information.

This deliverable explains our model-based approach to multi-level security monitoring which integrates a security information model and an Enterprise Architecture model to enhance the security analysis capabilities of existing SIEM systems. Specifically our concept adopts the Indicator Meta-Model (IM) introduced in the deliverable D4.1.1 [27] for multi-level security event modelling. The IM facilitates the definition of security probes on different levels in relation to security threats and requirements. As an enterprise modelling language we selected ArchiMate[3], an official standard of the Open Group that is widely used in industry to model and communicate multi-level Enterprise Architectures. SAMEA links together security, business and technical views on the enterprise and allows users to establish continuous security management process and get security relevant information at the required level of abstraction.

1.2 Guidelines Analysis

The analysis of the four validation scenarios for MASSIF that were identified in public deliverable D2.1.1 [26], besides typical issues like dependability, redundancy and fault tolerance, revealed the need for enhanced *security-related* features of future SIEM platforms. These features go beyond functional capabilities supported by existing solutions in many respects, especially concerning the capability to model incidents at an abstract level and exchange multi-level security-related information.

From the observed SIEM limitations, provided scenario requirements and guidelines of [26] we identified a set of requirements that are relevant for the contributions made within this deliverable. The following Table 1.1 summarizes these requirements.

[3] http://www.opengroup.org/archimate/

Table 1.1: Guidelines Analysis

Guideline	Description
G.S.1. Correlation across layers of security events	Advanced SIEM systems need to support enhanced correlation across layers, from network and security devices as well as from the service infrastructure. In this deliverable we propose a multi-level abstraction concept that enables escalation of low level security-related events to higher structural levels and facilitates accurate security incident detection even in the presence of uncertain information.
G.S.2. Multi-level security event modelling	Multi-level security event modelling should provide more holistic solutions to protect the respective infrastructures as the different abstraction layers of the enterprise architecture are not considered independently, but being related to each other. The proposed multi-level abstraction model allows to understand the effects of technical events on the user or process level of the system as well as consequences of security incidents occurring in business infrastructure for business security objectives.

In terms of *trustworthiness* and *legal* considerations, we assume that data we use for analysis are pre-processed and provided by trustworthy MASSIF components in a form compliant with the legal requirements stated in deliverable D2.1.1 [26].

1.3 Terms & Definitions Adopted in this Document

As agreed by the MASSIF Consortium, the main reference of security terms and definitions is provided by National Institute of Standard and Technologies (NIST) [17]. For definitions related to security of industrial control systems including Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCSs), and other control system configurations we refer to [37].

1.4 Structure of the Document

The remainder of this document is organized as follows. In Section 2 we analyze existing approaches to modelling Enterprise Architecture (EA), security information and events, and introduce the multi-level abstraction concept in order to enable multi-level security monitoring and security-related event escalation. In Section 3 we describe an extension to the Indicator Meta-Model that provides explicit connection

of the meta-model to the EA, and a conceptual architecture that defines corresponding continuous security management process. In Section 4 we demonstrate how to apply the proposed multi-level abstraction concept to the use case “On demand power production with supervision” from the dam scenario. Concluding remarks are given in Section 5.

2 Concept of Interrelated Abstraction Levels

In this chapter a Security-Augmented Multilevel Enterprise Architecture Meta-Model (SAMEA) is introduced. The main goal of the SAMEA is to define a flexible concept of interrelated abstraction levels for the various types of security-related events and enable event escalation through the level hierarchy. Existing SIEM systems do not allow to escalate and interpret multiple low level alerts on higher abstraction levels, such as a business impact view. A traditional SIEM solution works at a specific level of abstraction (i.e. network or application layer), and shows little support for multi-layer correlation and risk assessment. It usually cannot link detected incidents to business goals and system security requirements and is therefore unable to analyze their consequences on wider scale.

In order to relate events on different levels we need an abstraction concept that meets the following requirements (see also [25]):

- *Multi scale*. The concept should provide different views for the same set of security events depending on the task of analysis.
- *Elasticity*. The concept should overcome fixed abstraction level concepts that usually constrain flexibility of (multi-)hierarchy event classification and abstraction.
- *Cross cutting*. The concept should support interrelations between abstractions on different levels and in different domains and perform their comparative analysis.

In order to define abstraction levels and relationships between them we utilize the *notion of Enterprise Architecture (*EA) as the fundamental organization of an enterprise, including its information systems, and a specific domain within the enterprise, as well as the principles governing its design and evolution [12]. EA concentrates equally on business relevant elements such as business goals and processes, as well as IT platforms, software components and applications.A variety of EA frameworks are established in practice and research, for a comparison one can refer to [3, 33]. These frameworks define a formal and a highly structured approach of viewing and modelling an enterprise. Most frameworks encompass the following abstraction levels [40]:

- The business architecture - represents information on an enterprise from the business perspective.
- The process architecture - represents information on enterprise services, including business processes, organizational units and data flows.
- The software architecture - represents information on applications, data objects and software services that support business processes.
- The technology architecture - represents information on computing and telecommunication hardware and computer networks that support the software architecture layer.

In previous work, the Zachman Framework [41], Sherwood Applied Business Security Architecture (SABSA) [35] and The Open Group Architectural Framework (TOGAF) [10] were further evaluated and used for security engineering. The Zachman Framework consists of a two dimensional matrix based on the intersection of six question columns ("What", "Where", "When", "Why", "Who" and "How") with six rows according to reification transformations, and proved to enable traceability of requirements along the different layers of the enterprise architecture [14]. SABSA was developed independently from the Zachman Framework, but has a similar structure. It provides a model and a methodology specifically for developing risk-driven enterprise security architectures based on the business requirements for security. The Open Group[1] in cooperation with the SABSA Institute[2] has also released a white paper on the integration of security and risk management into enterprise-level architectures [11]. The different aspects of security risk management with consideration of the business perspective are also addressed by Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [2] and CORAS [8] approaches, however, their application requires a complex examination of an enterprise involving expert teams.

For the purpose of this work, the most applicable approach is a model-based approach to security information management as introduced in [16], though, like the other concepts, does not particularly consider the problem of security-related event escalation. The authors use dependencies and interrelationships of business and technical objects of an enterprise derived from a EA extended with security relevant information as a basis for systematic assessment and analysis of IT risks in organizations. We selected this approach as a starting point of our multi-level abstraction concept SAMEA introduced in the following sections.

2.1 Security-Augmented Multilevel Enterprise Architecture Meta-Model

2.1.1 The Concept & Structure

The Security-Augmented Multilevel Enterprise Architecture Meta-Model consists of two interrelated parts: an *architectural part* represented by an EA model and a *contextual part* expressed with a security information model. Both component models are related to the same enterprise and linked together through model artefacts that specify assets of an enterprise under consideration, such as business processes, applications, or technical nodes. The EA model describes concepts and dependencies between them that reflect the organization of different domains of an enterprise using multiple structural layers. Each layer of the model refers to an abstraction level, the number and contents of such layers should be defined in accordance with the activity field and security objectives of the enterprise. In most cases the following three layers are relevant: a *business layer*, which encompasses business processes, roles, services and objects; an *application layer*, which represents application components and data objects that provide informational support to business operations; and a *technology layer* defining the technical infrastructure required to run business applications. The artefacts within each layer as well as layers themselves are connected by different intra-layer and inter-layer relationships, respectively. As discussed above there exist many approaches to model an EA. In this deliverable we utilize the ArchiMate modelling language due to the flexibility it provides in tailoring the model to a particular use-case scenario (see Section 2.2).

[1] http://www.opengroup.org/
[2] http://www.sabsa-institute.org/

The EA model helps to reveal sensitive assets within the enterprise infrastructure and to identify (multi-level) dependencies between the assets. But from the perspective of security analysis and monitoring this information needs to be enriched with security concepts, specifically security requirements, threats and response actions. A security information model describes the concepts pertaining to protection and preservation of the confidentiality, integrity, authenticity, availability, and reliability of information. This model is usually highly specific to its application domain, as illustrated by the Cyber Security Modeling Language (CySeMoL) for network attack detection and prevention [36] or Organization-based Access Control (OrBAC) [1]. For the purpose of security-related events escalation the model should provide definitions on different abstraction levels in relation to security threats and requirements. In this regard IM introduced in [27] serves the purpose in the best way.

In general the architectural and contextual part of the SAMEA can be replaced depending on the application domain and business security goals, on conditions that the link between critical assets, security requirements and generated security alerts is available. Sometimes it is sufficient to define a proper *model view*, a specific perspective of the repertoire of EA models that presents a set of model elements and relationships of the certain type or intended for particular problem domain, system aspect or stakeholder as well as interrelation of those.

The SAMEA defines the following steps of multi-level security monitoring process:

1. Definition of the general security objective as a high-level security property of a business process. The security objective can be derived from enterprise security policies standards established at an executive level of the business and is not tied to any specific technology.

2. Creation of an EA model and selection of a proper model view in accordance with the defined security objective, application field and selected EA modelling approach.

3. Identification of dependencies and definition of dependency graphs using inter-layer relationships between elements of the EA model.

4. Security requirements propagation and creation of security probes based on the IM. The propagation of security requirements implies tracing high-level security properties through the EA model dependencies and represents a mapping of a property to the specific domain (model element). The creation of security probes (indicators) we discuss in detail in Section 3.1.

5. Escalation of security-related event generated by security indicators. Escalation routes are derived from requirement hierarchies and the EA model dependencies.

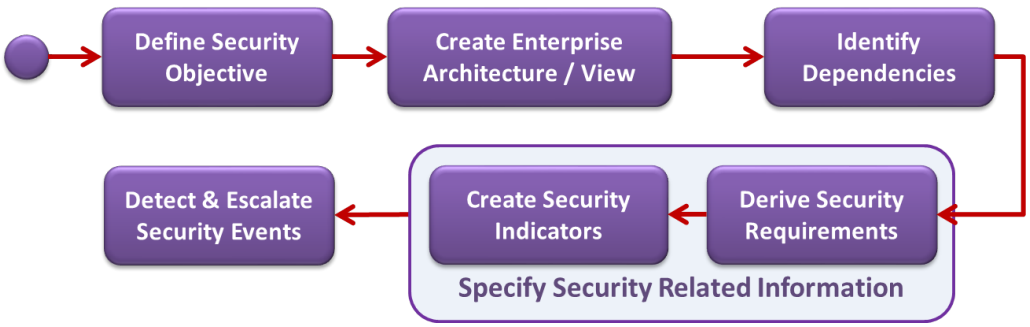

Figure 2.1: Security Monitoring and Event Escalation Process

The schematic representation of the SAMEA-based security monitoring and event escalation process is given in Figure 2.1. The steps related to security requirements propagation and event escalation are discussed below.

2.1.2 Requirements Propagation & Events Escalation

Dependency Graph Definition An EA model uses different types of relationships to express dependencies between model elements. They can represent the control or information flow (dynamic relationships), structural relations (structural relationships) or sharing of some common characteristics (typification relationships). These dependencies direct propagation of security relevant information, such as security requirements, alerts and countermeasures between different model domains. In some scenarios only the presence of a link between two elements is relevant. For such cases the SAMEA includes a derived relationship with the definition provided by the ArchiMate standard [9]. In order to replace relationships of the same type connecting two model elements with the derived one, the direction and the strength of the initial relationships joining at an intermediate element is examined. The weaker relationship can be used as a replacement. In the same way a dynamic relationship between two behavioral elements can be transferred to the active structural elements associated with them or services realized by them. The corresponding rule is given below.

Definition 1. *Assume that P, Q, R are three types of relationships between model elements. If two structural relationships $p(a, b)$ and $q(b, c)$ exist between elements a, b, c such that $p \in P$ and $q \in Q$, then a structural relationship $r(a, c)$ also exists, where $r \in R$ and type R is the weakest of P and Q.*

Currently the SAMEA does not make use of specific subtypes of structural, dynamic or typification relationships taking into account the derivation rule given in Definition 1 and supports only one type of a dependency relationship defined in the following way:

Definition 2. *A dependency relationship between any two model elements exists if these model elements are connected with any of the relationships of an enterprise architecture model. The dependency relationship has the same direction as the initial relationship.*

This definition can be easily extended for individual types of dependency relationships, eventually forming the dependency graph:

Definition 3. *A dependency graph is a set of model elements connected with a dependency relationship. The dependency graph is directed and weighted, meaning that the dependency relationships are consistently directed towards higher or lower layers, and that each relationship is associated with a numerical strength.*

The directed dependency graph guides the tracing of security-related events and security requirements. The weights assigned to the arcs of the dependency graph can be used for filtering and correlation of escalated events.

Security Requirements Propagation A specific security requirement is a set of constraints on an element of the EA model aimed to protect it from security risks. After a dependency graph is defined, the high-level security objectives can be propagated through the graph and translated into specific security properties of the respective dependent model elements. The security requirement propagation is

done top-down and results in a hierarchy of requirements starting from the main security objective. A requirement defined for higher layers elements is inherited by dependent elements in lower architectural layers.

Security Event Escalation The IM in combination with the dependency graphs and requirement hierarchies defined on the basis of the EA model gives a possibility for security event escalation. The IM describes a security probe which gets environmental events, correlates them with the current security context and carries out response actions (generates an alert) in case it is triggered. Thus indicators can be considered as one of the sources of security-related events. Dependency graphs related to the detected incident can be easily identified using the connection of the indicator to the EA model. The escalation of security events produced by other security systems (antivirus software, Intrusion Detection System (IDS) systems, SIEM systems, etc.) can be performed in the same way if the involved assets are included in the alert. Otherwise a security incident can be traced through a hierarchy of security requirements built upon a dependency graph.

Security-related events can be escalated both in bottom-up and top-down directions depending on the place of involved model elements in the EA using an event escalation route. The definition of the event escalation route is based on the notion of a simple path in graph theory.

Definition 4. *An event escalation route is an acyclic path in a dependency graph with one or more start vertices representing model elements that experienced a security incident.*

2.2 Enterprise Architecture Modelling (ArchiMate)

To realize the architectural part of the SAMEA we chose ArchiMate 2.0[3], a public standard of the Open Group to model and communicate EAs. It has several important advantages regarding the concept of interrelated abstraction levels. First of all ArchiMate offers an integrated architectural approach that allows to describe and visualize different architecture domains, their relationships and dependencies [18]. It implements a service-oriented approach to relate different domains. Another important feature of ArchiMate is its formal foundation that enables unambiguous interpretation and automated analysis of the models [13].

ArchiMate Standard supports the TOGAF Architecture Development Method (ADM) and uses terms defined by the TOGAF standard [10], it can be mapped to such widely used modelling notations, as Unified Modeling Language (UML) [15], Business Process Modeling Notation (BPMN) and Petri Nets [39].

ArchiMate specifies a lightweight and scalable EA visual modeling language based on service-oriented approach, which provides uniform representation of EA diagrams depicting architecture domains, layers and aspects [9]. This language helps to identify the multi-level dependencies between different layers or aspects of an architecture like those between business processes and supporting applications or technical infrastructure [19]. The elements of ArchiMate can be organized in a two dimensional matrix: elements (active, behavioral, passive) and layers (business, application, technology). This matrix represents the architectural framework of ArchiMate.

[3] http://www.opengroup.org/archimate/

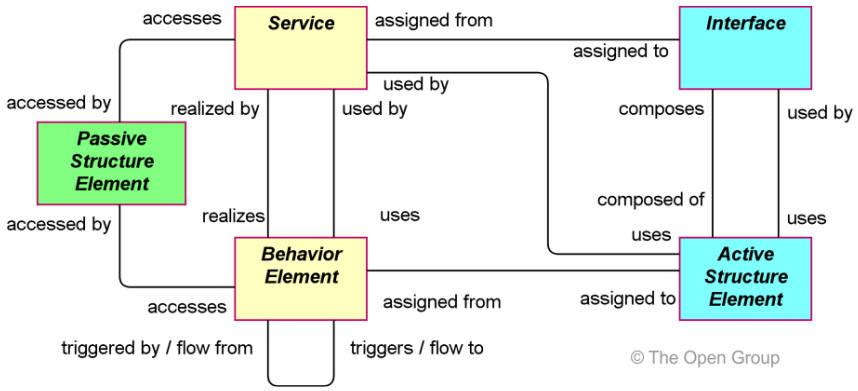

Figure 2.2: The Core Concepts of ArchiMate

The language uses active structure elements to model acting subjects in EA such as business actors and application components, passive structure elements to model objects manipulated with actions, and behavior elements to represent activities. Figure 2.2 illustrates the core concepts of ArchiMate, passive structures are colored with the green, behaviour structures - with the yellow, and active structures are blue. The set of concepts is further extended by the internal/external view and the individual/collective view. The structure elements can act in collaboration and show collective behavior (interaction). ArchiMate also introduces services and interfaces to model externally available functions and points of access to these functions respectively. The service concept represents a unit of functionality that some element, e.g. an application or a computer node provides to other elements.

Besides that the elements can be interrelated. ArchiMate defines three kinds of relationships:

- Structural relationship - models the structural coherence of concepts of the same or different types. Some concepts, such as structural “composition”, “aggregation”, and “association” are adopted from UML 2.0. Additional structural relationships are introduced for behavior elements and objects (“access”), behavior elements and subjects (“assignment”), services or interfaces and behavioral elements (“used by”). “Realization” links an element with other element that realizes it. “Association” is the weakest structural relationship; “composition” is the strongest.

- Dynamic (temporal) relationship - models (temporal) dependencies between behavioral concepts. Dynamic relationships are used for modelling data exchange (“flow”) and temporal connections (“triggering”) between such behavioral elements as processes, function, interactions and events.

- Other (typification) relationship - does not belong to the first two categories, and contains “specialization” (UML 2.0), “junction”, and “grouping” relationships. Apart from “specialization” ArchiMate allows to associate relationships of the same type with “junction” and objects having common characteristics with “grouping”.

The important concept of ArchiMate is *derived relationships*. The abstraction rule for structural relationships analyzes the direction and the strength of two relationships joining at an intermediate element, combines them and replaces by the weaker one [9]. A dynamic relationship between two behavioral elements can be transferred to the active structural elements associated with them or services realized by them.

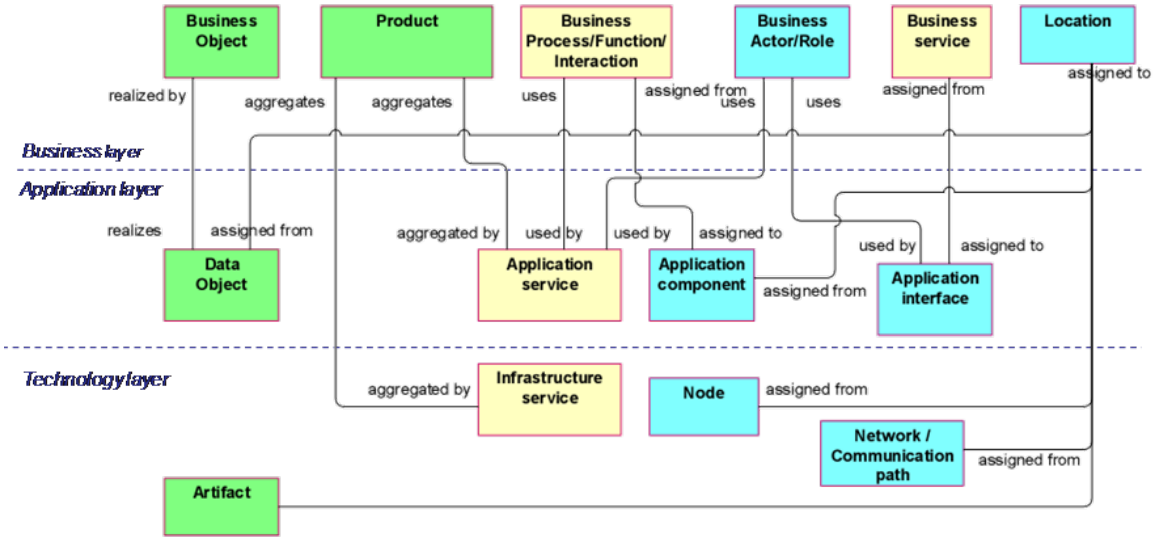

Figure 2.3: Relationships between EA Layers in the ArchiMate

ArchiMate considers an enterprise as a layered system and distinguishes three layers when modelling the EA:

1. The Business Layer, which defines services and products created by internal business processes and offered to external parties.
2. The Application Layer, which defines applications supporting internal business processes as well as their interfaces and data types.
3. The Technology Layer, which defines infrastructure services enabling business applications, including corresponding hardware and software.

The elements of each layer realize the same types of concepts and relationships. Relations between different layers can be expressed using services (service orientation and “used by” relationship) and “realization” relationships, when elements in lower layers implement elements in higher levels. The relationships between the three layers are shown in Figure 2.3. The detailed description of the structure of each layer can be found in [9].

3 Integrating Security Event Modelling & Multilevel Abstraction Concepts

This chapter describes integration of the security event modelling concept or Indicator Meta-Model (IM) introduced in public deliverable D4.1.1 [27] with the multilevel abstraction concept SAMEA proposed in Section 2.1. The SAMEA meta-model enables analysis of dependencies between elements that are associated with one business process but belong to different abstraction levels, e.g. technological nodes and business activities. The SAMEA consists of two main parts: an EA model and a security information model. The IM, in turn, is a security information model establishing a format for multi-level security analysis and reporting that allows to define security probes on different levels and relate them to security requirements. Due to these features the IM can facilitate security events escalation and therefore needs to be linked to the EA part of the SAMEA. Below we describe an extension to the IM which realizes the reference between the hierarchies of EA model elements (assets) and security concepts, such as the security requirements. This allows a systematic tracing of security incidents along dependencies from low level infrastructure security issues to higher structural levels where we can estimate the effect of incidents on business goals. The explicit connection to defined assets can also facilitate the creation of indicators as well as security requirements propagation. In the following we introduce an extension to the IM and propose corresponding conceptual architecture for security event processing.

3.1 Extended Indicator Meta-Model

The Indicator Meta-Model (IM) was introduced in D4.1.1 [27] as a structure and process to detect and manage security incidents. In particular, the IM describes a way to create new security monitoring probes based on Complex Event Processing (CEP) or database queries. The model consists of *abstract indicators*, high level structures describing security probes in plain text, *real indicators* instantiating abstract definitions, and the process of indicator creation and deduction. One or more real indicators defining configuration of CEP-based security monitoring related to certain security properties (threats) can be derived from an abstract indicator.

The IM follows an *"on-if-do-why"* approach and can be interpreted in the following way: *on* detection of the attack pattern in the event stream and *if* the monitored system is in the given condition, the response action is *do*ne on the reason that (*why*) security property is violated (or security threat is realized).

The IM has four logical parts[1]:

`EventStreamProperty` ("on"): models event patterns that indicate a security incident. This part

[1] All named elements of the IM are written in `typewriter font` beginning with an upper case

specifies misuse signatures or anomalies based on parameters extracted from the event channels and functions or operators applied to them together with criterion, evaluating extracted parameters and triggering the Indicator.

`Condition` ("if"): denotes context or a system state condition to be validated whenever an event pattern is found. A security incident is detected and a response action is triggered only in case the condition is met. [27] does not restrict the format of the condition part, specifying it as a condition query on the repository including the context and domain specific information which can comprise multiple results (if-then-else).

`Action` ("do"): models executable response actions to be carried out when indicator is triggered and condition is met. Actions can be atomic or complex, consisting of multiple context-dependent reactions, internal or external. Internal actions update the knowledge base, set system state parameters or trigger another indicator. External actions imply plugins that can notify administrators or perform more advanced countermeasures.

`SecurityPertinence` ("why"): provides a reference to the security issue addressed by the indicator, i.e. a security property or attack/anomaly description. Basically this part explains why certain actions have been performed and helps to estimate the impact of the incident.

In Section 2.1 we proposed an approach to escalate events from different abstraction layers of a multilevel EA using dependency graphs that represent relationships between elements of the model. But from a security management perspective not all environmental events generated in lower layers of the architecture are relevant for security analysis on higher layers. The relevance of an event is determined by the high-level business security goal and the corresponding set of security concepts (requirements, threats and countermeasures) that should be defined for every (critical) element of the EA model. In our case it is the IM that formulates important security concepts. Hence, to enable escalation of critical security events based on the asset relationships we need to link the IM to the multilevel EA, by extending the previously described IM with explicit asset references. Note that this extension does not contradict the definition of the indicator. Preconditions for the creation of indicators given in [27] name a set of security requirements together with relation to the critical assets of the system as a required input for the deduction of the indicators. Our extension explicitly integrates this relation in the indicator. The structure of the extended indicator is presented in Figure 3.1. The new structural element `AssetReference` extending the IM is specified below.

`AssetReference` ("what") is either a single element, a group or a list of elements of any type that belong to an EA model. We assume that operational structure of the enterprise might be described with several architectural models. Therefore the *Name* field must unambiguously identify the reference EA model.

The `AssetReference` comprises the following elements:

- *Name* [1] is the identifier of the `AssetReference`.
- *Type* [1] is the type of the `AssetReference`. We distinguish a single, a group and a list type. The single type designates an atomic element of the EA, such as an activity or a sensor. The group refers to a set of elements linked together with a "grouping" relationship of an EA. Within the extended IM the group is treated as the integrated whole, (i.e. it can have only one location and owner). The `AssetReference` of the list type contains a list of assets that can be distributed over several locations or/and owned by different stakeholders.

- `Instance` [1...n] specifies the content of `AssetReference` and refers to a particular EA model element(s) such as a process activity, an application component or a device.

`Instance` has three parameters:

- *Name* is the identifier of the `Instance`.
- `Location` [1] models the placement of the model element and denotes a geographical, physical or logical location. It can refer to a site inventory, domain framework, network map, etc. depending on the kind of the element and the abstraction layer. Any `Instance` should have one `Location`.
- `Owner` [1...n] is a stakeholder or a role that bears full responsibility for a model element. There must be at least one `Owner` to control a security state of the element.

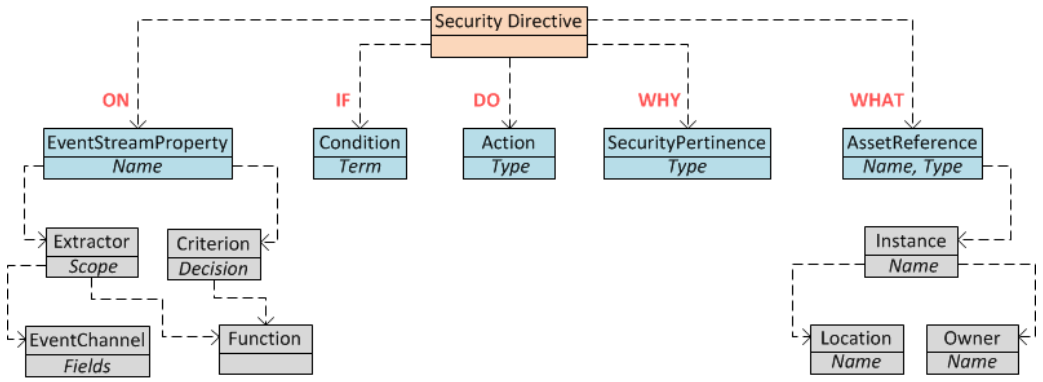

Figure 3.1: Structure of the Indicator

The proposed extension to the IM does not disturb the process of indicator creation and application introduced in [27]. The "what" part allows to refine some steps earlier left to the discretion of security administrators. We give the explanation below:

1. The relation of security requirements to the critical assets of the system which is stated as one of the preconditions to create an indicator will be stored as a part of the definition of the extended indicator. It enables easy identification of *dependent indicators* and *escalation of security events* through asset dependency graphs. An activation of one of the dependent indicators can cause a cascade triggering of other security probes in the chain even if it is not explicitly defined in the "do" part of the indicator. The latter means that an incident is detected even if initial sensors (implementing the "on" part) are compromised.

2. A collection of critical security properties that the monitored system should possess is necessary to create an indicator. So far it was not discussed how to ensure that a set of indicators is complete and covers all critical properties. Combining the EA with the IM allows to analyze indicators using security requirement trees built out of the dependency graphs, and in this way to reveal indicators which are missing or redundant. By aligning indicators with requirement trees helps to detect conflicting response actions which can block operations within the enterprise if realized.

3. Having an asset reference as a part of the indicator helps to identify the set of measurement points, including domain specific field sensors and physical sensors, to formulate context conditions related specifically to the particular asset, and to update this information if the EA changes, e.g. new sensors appear or security systems are deployed.

To sum up, the introduced extension to the IM allows to increase overall security awareness as well to counteract security incidents in more robust and adequate way due to closer semantic relations and mutual information exchange between security concepts and actual enterprise structure.

3.2 Conceptual Architecture for Security Event Processing

In this section we introduce a conceptual architecture for security event processing based on the previously described extended IM. The architecture can be used to realize an event processing system aimed to identify and respond to events related to violation of security requirements or security incidents occurring across different abstraction layers of the EA.

The proposed architecture develops an Event Processing Network (EPN) framework offered in [34, 21] to describe the structure of event processing systems. The framework consists of four components: The event producer, event consumer, Event Processing Agent (EPA) and an event channel modelling connection medium between EPN components. An event producer or event source collects environmental events and forwards them to EPAs. An event consumer receives events produced by EPAs and uses them according to the purposes of the component, e.g. for further analysis, reporting or visualization. An EPA carries out different processing operations on the events received from a source. Depending on its capabilities EPAs can be classified into filter agents, pattern detection agents and transformation agents [7]. A filter agent discards events that do not meet a filtering condition. Pattern detection agents identify event patterns in input stream, i.e. performs event correlation, while transformation agents input events into output events using a built-in conversion function. EPAs are connected to event producers and consumers as well as other EPAs by means of event channels.

The IM defines a sequence of event processing operations aimed to support event driven security analysis and therefore perfectly fits into the EPN framework. The process of indicator application described in [27] uses the notion of security probe to represent components implementing IM. State transitions of security monitoring with an indicator-based security probe are illustrated by the UML state diagram in Figure 3.2. From an input event stream, a security probe selects events that pertain to the monitored asset(s) and evaluates `Criterion` using extracted parameters. In case the security event condition is met the indicator is triggered and validation of the context or system state `Condition` is required to classify and react to the detected incident. To validate context condition the security probe analyzes results returned by a condition query. Based on the context condition met the probe identifies the violated security property and countermeasures to be taken. The security probe is not supposed to carry out response action itself, but it should define which reaction is necessary and activate corresponding components.

As performance and scalability can be critical for security monitoring applications we propose to enhance a security probe with a distributed architecture that consists of a set of dedicated EPAs. An EPA has interface to an event bus for translation and processing of transmitted environmental events as well as for receipt and adoption of configuration commands being distributed within the IM dispatching. An EPA can also generate output events and/or alarms which are disseminated via common communication medium. The internal logic of an EPA is implemented in its core and corresponds to the type of the EPA.

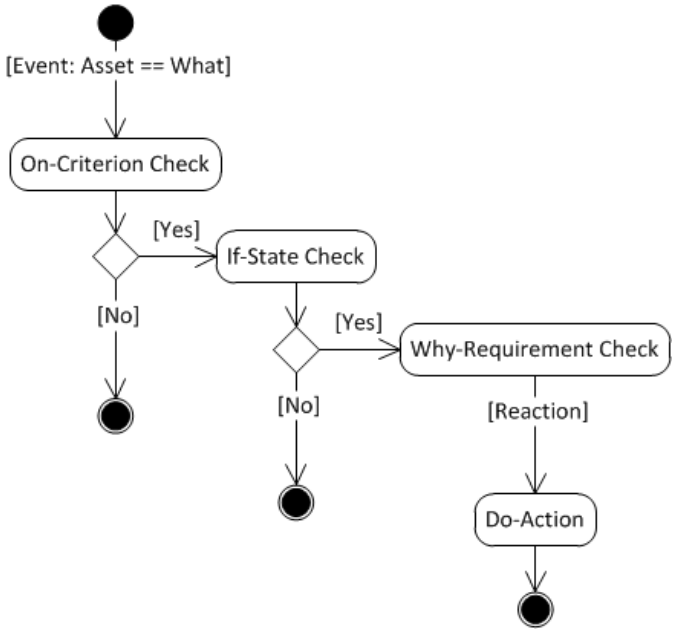

Figure 3.2: State Diagram for Event Driven Security Monitoring

Such approach will also enable flexible combination and re-use of event processing components. The components of the proposed conceptual architecture for security event processing based on the extended IM are presented in Figure 3.3.

The architecture consists of several EPAs and one distinguished agent, the Security Strategy Agent (SeSA), which is similar to Security Strategy Processing Component (SSPC) introduced in [32]:

Event evaluation EPA: The on-EPA performs filtering, correlation or transformation of events according to the `Criterion` directive given in the "on"-part of the IM and generates an on-event (alert) if result is positive. An on-event means that a misuse or an anomaly is detected in the input event stream and serves as a trigger to other IM-based components. In a SIEM architecture on-EPA refers to a CEP engine.

Context validation EPA: An if-EPA is activated by an on-event and validates the condition set in the "if"-part of the IM using events produced by the corresponding condition query. In case of positive validation an if-EPA generates an if-event (alert) designating a suspicious system state. A SIEM can use several different context validation EPAs responsible for different aspects or domains, such as process state, application state or network state, real or predicted [6].

Asset Reference EPA: The what-EPA realizes the "what"-part of the IM and provides connection to the Enterprise Architecture required to determine critical assets involved in the security incident. The what-EPA takes asset information from on- and if-events and returns dependency graphs to facilitate requirement analysis and event escalation.

Security information EPA: A why-EPA implements the "why"-part of the IM and is responsible for determination of violated security properties. The why-EPA listens to on- and if-events and uses their together with provided asset information to identify a broken security requirement. The resulting security pertinence data are sent in a why-event (command) to trigger response actions.

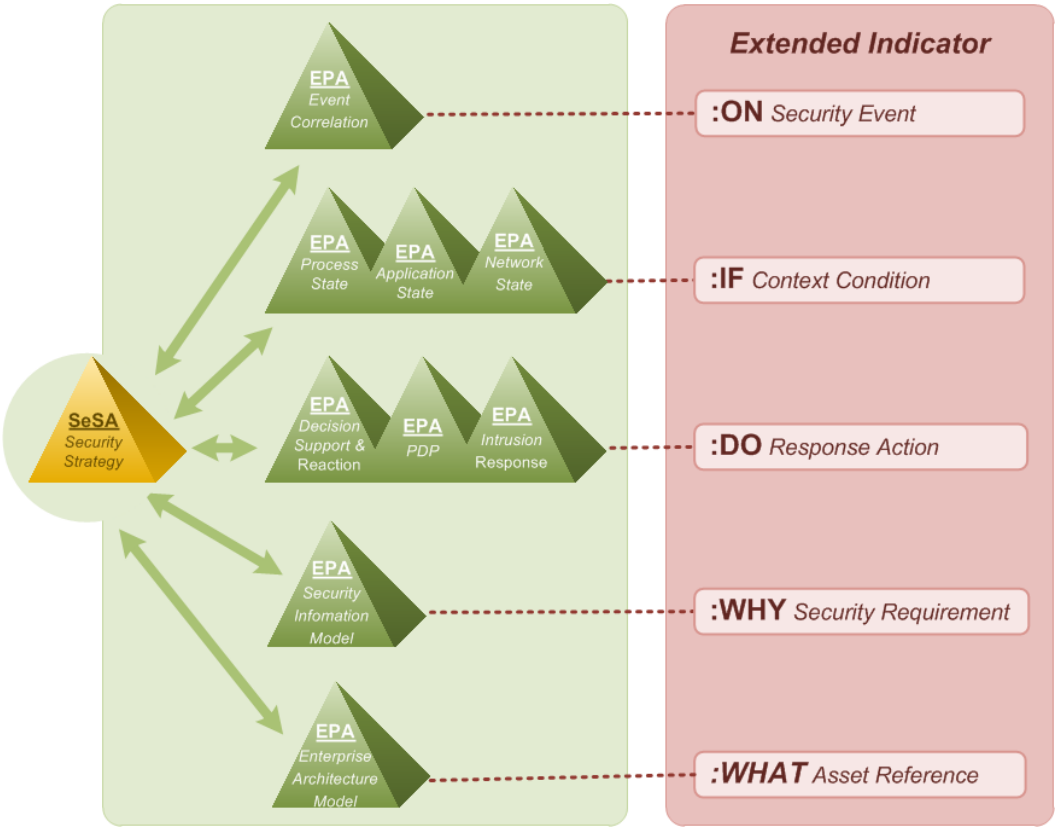

Figure 3.3: Conceptual Architecture for Security Event Processing

Reaction EPA: The do-EPA is responsible for taking countermeasures in response to the detected security incident in view of the observed system state and implements the "do"-part of the IM. The do-EPA is activated by a why-command and implements e.g. a policy decision point.

Security Strategy Agent: The SeSA controls the execution of the indicators and orchestrates EPAs. It can execute an indicator or parts of it directly or delegate the tasks to specialized components. The SeSA initially gets the IMs from the why-EPA. The SeSA parses the indicators, identifies the responsible EPAs for each subtask and distributes a respective configuration directive to the EPAs.

The proposed conceptual architecture can be used for implementation of a distributed security event processing system that integrates specialized subsystems responsible for IM-based security management tasks. The architecture can also form a basis to determine the component structure of monolithic systems that monitor and analyze environmental events from a security perspective. As an example of the first approach we can name a SIEM system that embodies inventory subsystems, event sensors, analysis subsystems (for risk assessment, data mining, etc.) and intrusion reaction subsystems. The second approach we demonstrate using the architecture of the Predictive Security Analyser (PSA), the model management component of MASSIF SIEM [29]. The PSA is fed with events by the CEP or Generic Event Translation (GET) through the Common Repository, performs close-future process behavior simulation and prediction of possible security violations and generates additional security symptoms (security alerts), which are fed back to the Repository for the Decision Support and Reaction (DSR) and Visualization components.

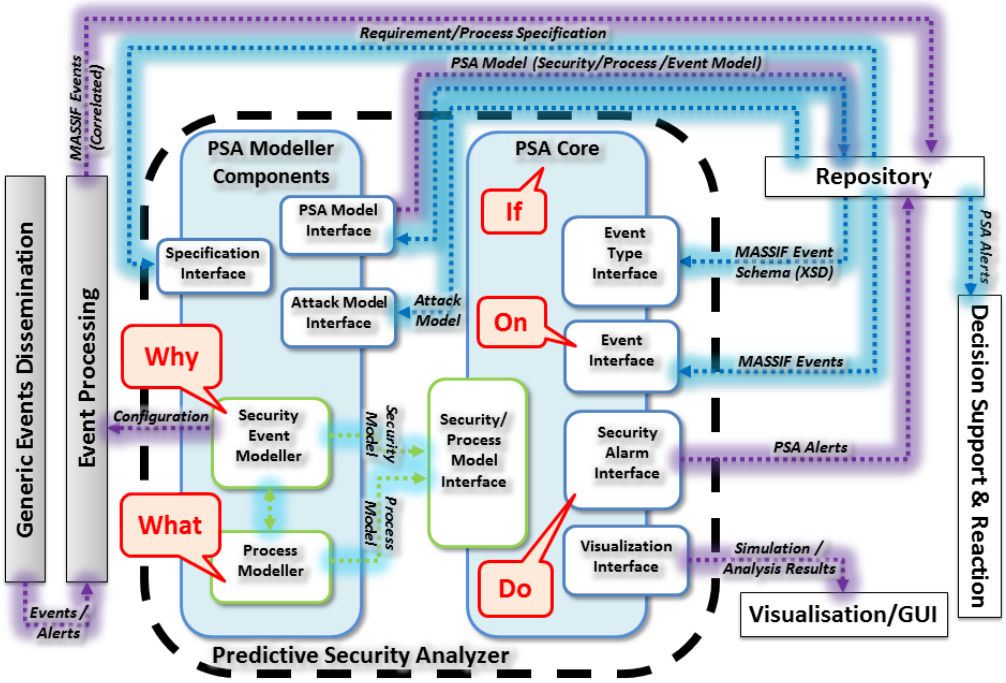

Figure 3.4: "What-On-If-Do-Why" Concept Mapping to the PSA

Figure 3.4 provides the mapping of the "What-On-If-Do-Why" concept to Predictive Security Analyser (PSA) components. The PSA Modeller Components implement "What" and "Why" parts of IM. The Process Modeller receives the process descriptions including associated asset information and transforms them into an internal PSA model (Asynchronous Product Automaton (APA)) which is used for the continuous real-time analysis and close-future simulation of the process. The Security Event Modeller performs security requirements elicitation and creates an internal security model based on security goals and properties related to the monitored process to enable detection and prediction of critical process states. The PSA Core Components realize the "On", "If" and "Do" parts of the indicators. The Event Interface communicates through the MASSIF repository with the CEP, the source of correlated events. The correlation rules are created by the Modeller Components, thus observation of an event automatically means that on-criterion is triggered. The parameters of the events are defined using the Event Type Interface, which delivers definitions of the format and the content of all events processed by the PSA. The PSA Core executes the models generated by the Modeller Components and estimates the monitored process state conditioned by the observed events. If the current process state leads to violation of a security requirement or to some critical state in future, the if-condition is met. Note that the PSA has only one type of response actions for selection - it sends a PSA alarm to notify other MASSIF components about the detected critical situation. The "do" component is realized by the Security Alarm Interface that delivers the identified security violation to the MASSIF Repository.

The conceptual architecture for security event processing incorporating the extended IM and the EPN concepts increases usability during design of corresponding systems and gives better understanding of the structure of the system, thus benefiting system efficiency and performance. Additional advantages to scalability and performance can be gained by implementation the EPAs in a distributed architecture.

4 Security-Augmented Multilevel Meta-Model Specialization Example for Dam Scenario

In this chapter we introduce an instantiation of Security-Augmented Multilevel Enterprise Architecture Meta-Model that deals with the critical infrastructure (hydroelectric power plant) control activity. Our specialization example explores use and misuse cases characterizing the validation scenarios for MASSIF identified in public deliverable D2.1.1 "Scenario Requirements" ([26], Section 6) by "EPSILON", the dam scenario provider.

The basic idea is illustrated by Figure 4.1. The storage dam stores water upstream in the reservoir and feeds the hydropower station to generate hydroelectric power. When gates open on the dam gravity forces the water down through the penstock. Pressure increases as water flows through the pipeline. Dammed water delivered through the penstock drives the water turbine and the generator in the powerhouse. The turbine blades produce rotation of magnets inside the generator generating alternating current which is transformed to higher-voltage current by the transformator. Electricity flows to customers via power lines coming out of the power plant. Used water is delivered through outflow pipelines to the river downstream. Dam is a complex system consisting of natural and artificial parts, units and subsystems, including foundation, reservoir, the operating organization and the powerhouse. To monitor and control such complex system a large number of heterogeneous devices is deployed. Additionally dam system has wide geographic spread that implies highly distributed monitoring and control operations. As the dam control system gets more complex the failure probability increases [31]. That is the reason why buildup security monitoring provided by MASSIF SIEM is very important in such critical infrastructures to estimate the relevance of the events occurring on a dam and to detect anomalies or malfunctions at both the infrastructure and the process level.

In the considered scenario the dam is continuously monitored by an Automated Data Acquisition System (ADAS), coordinating the automatically recorded structural and geotechnical instrumentation data and transmitting information about dam behavior to the control center. In addition SCADA system is used to push automated or operator-driven supervisory commands based on information received from remote stations to configure remote actuator operations. Since ADAS is designed to perform monitoring operations, its components can be adapted to the SCADA system, e.g. to provide sensor data to the SCADA devices [5]. Dam monitoring applications control many different parameters related to dam structure, water flows, and device states, each of which is measured using specialized sensors placed in the field (inclinometers, tiltmeters, piezometers etc.). Measurements produced by field devices are collected by Remote Terminal Units (RTUs) in case of SCADA and Remote Monitoring Units (RMUs) in case of ADAS that convert sensor data and send them over computer network or data bus to the remote Master Terminal Unit (MTU) or remote workstations accordingly. Monitoring and Control Unit (MCU), advanced RTU and Programmable Logic Controller (PLC) devices allow control systems to perform control operations on actuators autonomously. The subsystems composing the dam infrastructure can be

protected using firewalls and VPNs, IDSs, and other off-the-shelf security solutions. The dam scenario

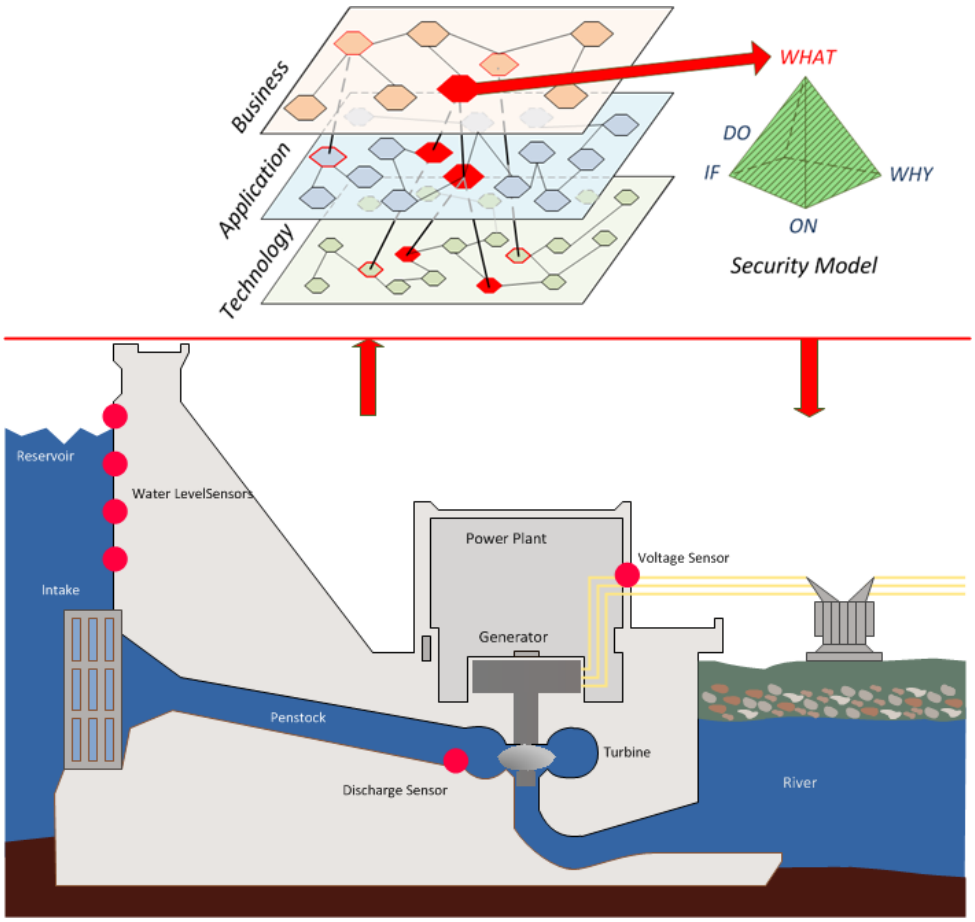

Figure 4.1: Multilevel Modelling of Hydroelectric Powerplant

specifies requirements to a SIEM system that comprise identification of the security events at infrastructure, service and process levels as well as correlation of physical and logical events. The system should consider dam purposes, functionalities and structure as well as established regulations to analyze, predict and react to critical situations. The SIEM for dam security monitoring should be aware of the context in order to correctly interpret the detected symptoms.

In this deliverable we described the Security-Augmented Multilevel Enterprise Architecture Meta-Model (SAMEA) model which integrates a security information model and an Enterprise Architecture (EA) model to provide conceptual grounds for multi-level security monitoring and security-related event escalation necessary to satisfy the requirements posed by the dam scenario providers. In particular the model exploits the Indicator Meta-Model (IM) introduced in [27] specifically for multi-level security event modelling and ArchiMate [9] enterprise modelling language for creation multi-level EA models. Figure 4.1 shows the process of the SAMEA application to the dam scenario that consists in definition of EA abstraction layers and design of EA meta-model views that most fully reflect the business security objective, and construction of interlevel dependency graphs using relationships between elements of the meta-model for requirements propagation and refinement. For each critical element of the EA meta-model IM model that monitors system compliance with the given security requirements is speci-

fied. Finally events escalation from lower abstraction levels to higher ones is performed based on derived indicators that are triggered by observed events and interlevel dependency graphs expressing asset hierarchies. The SAMEA concept is intended to facilitate security analysis and sound decision on response actions launched within the dam infrastructure. The latter stands behind the arrow from the model part of the picture to its dam part.

In the following we demonstrate how to implement SAMEA concept for one of the dam processes and produce respective eXtensible Markup Language (XML) metadata descriptions. The specialization example uses XML notation compatible with the Infrastructure-Metadata Access Point (IF-MAP) specification [4]. We chose this notation because it is the only standardized and industry supported solution known to us which is focused on security-device integration and stateful runtime metadata sharing and correlation for security related decision making and policy enforcement. Adoption of this standard allows easier integration with many existing SIEM systems.

4.1 Dam Process Description: On demand Power Production with Supervision

For the purpose of demonstration we consider a process of on demand power production with supervision following the use cases[1] within the dam control activity. In our example we assume that all operations performed on the dam control station must be supervised by authorized dam personnel (a control station operator). The SIEM should generate an alert if an actuator sends a command to be executed by the dam control station and the operator is not present at the dam control center to supervise its execution.

The process workflow is illustrated in Figure 4.2, Figure 4.3 and Figure 4.4 using UML notation[2] to express process's dynamic. The hydroelectric turbine of the *power plant* (*PP*) is connected to the dam by means of penstocks. To produce electric power on demand the *PP* sends a request on water discharge through the penstocks to the dam *control station* (*CS*) which feeds the turbine. The requested water discharge v_{dis} may vary, depending on the demanded hydroelectric power production. The *CS* displays the request at the dashboard and waits for the confirmation from the *control station operator OP*. In case the *OP* is not present at the *CS* at the moment or does not posses privileges to give such confirmation, the discharge request cannot be executed. On receiving the confirmation the *CS* triggers the discharging operation and sends a command to open the gate (*GT*) serving the connected turbines. The *GT* processes the command by comparing its current state with the requested state and changing the current state if the states differ. At the end of the operation *gate status monitors* send the information about the current state of the *GT* to the *CS*. The *CS* displays the new *GT* status at the dashboard.

The control station *CS* evaluates the discharging value v based on the measurements produced by the *discharge sensors* in the penstocks and opens the gate until the discharging water on the penstocks satisfies the *PP* request. When the measured discharging value meets the requested threshold the *CS* displays an alert to stop opening the *GT* and upon the confirmation sends a command to stop the gate *GT*. The command is processed by the *GT* in the same way as described earlier resulting in the *GT* state "opened".

[1] "Sending commands from the dam control station with legitimate rights" (use case 8); "On demand electric power production" (use case 2)

[2] http://www.omg.org/spec/UML/2.4.1/

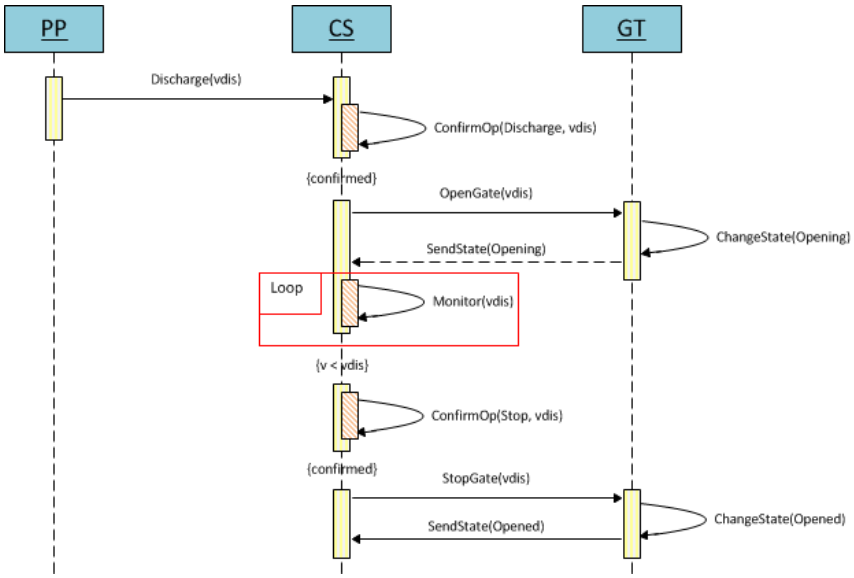

Figure 4.2: *PP* discharge request workflow

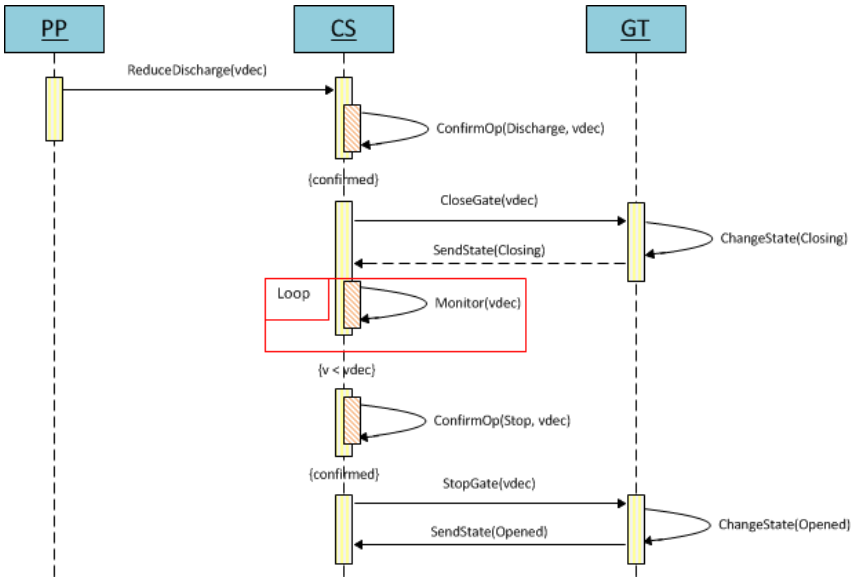

Figure 4.3: *PP* reduction request workflow

We consider the following three alternatives for process development, in the main following the procedure discussed above. If the water discharge towards the turbines is not enough for satisfying the demanded electric power *PP* sends a request to increase the discharge to v_{inc}. The states of the gate *GT* correspondingly change to “opening” and “opened”. If electric power demand decreases the power plant *PP* sends a request to reduce the discharge on the penstocks to the specified value v_{dec} or to stop the discharge on the penstocks (then the discharging value is the system constant v_{min}). In the first case the *CS* sends supervised commands to partially close the *GT* and to maintain the discharging value v under the requested value. The *GT* states sequentially takes on the values “closing” and “opened”. In the last

case the *CS* closes the gate *GT* to stop water discharge through the penstocks, bringing the discharging value to the minimum v_{min}. The *GT* changes its states from "closing" to "closed".

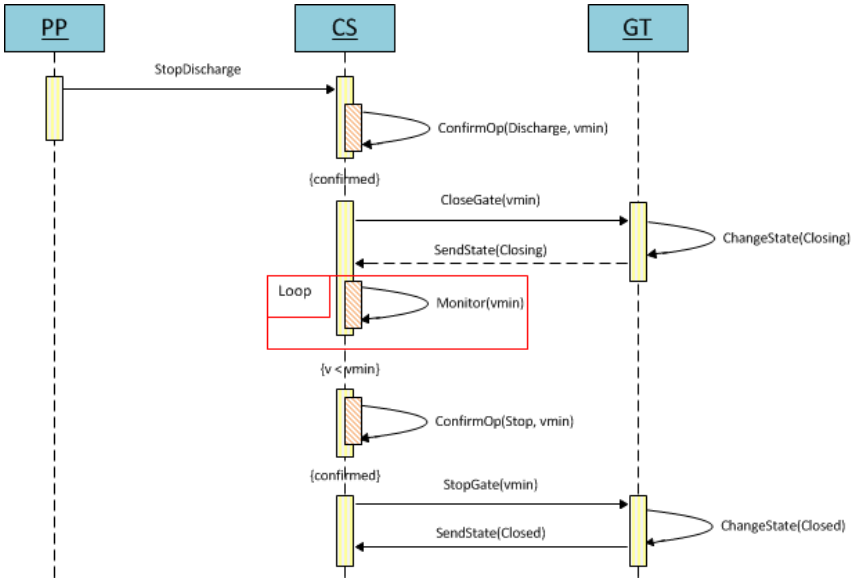

Figure 4.4: *PP* stop discharge request workflow

The authorized control station operator *OP* must supervise all operations performed in the *CS*. To access the *CS* premises the dam *CS* personnel must produce an RFID badge that carries personal identifier id_{op} granting access to the dam control center. A dam *access control system* (*ACS*) monitors access requests and logs access events. After entering the control center the *OP* can log on to the *CS* system using *OP* credentials $cred_{op}$. The *CS* implements role-based access control and distinguishes two user roles: *privileged user* (administrator) and *normal user*. We assume that *OP* require dam administrator rights $cred_{adm}$ to confirm discharge requests of the power plant. Figure 4.5 illustrates the respective workflow. The *OP* logs in as a dam administrator, checks the displayed discharge request and sends a command cmd to dam actuators to start the discharge. When the task is finished the *OP* logs off out the *CS* system. To allow the *OP* to leave the control center *ACS* verifies his/her RFID identifier again.

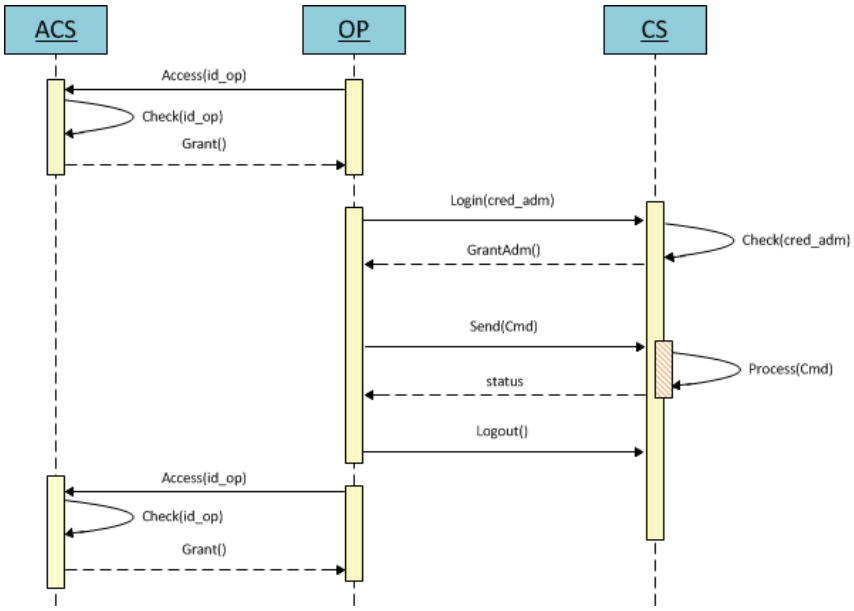

Figure 4.5: Control station operator authorization workflow

4.2 Multilevel Enterprise Architecture View for Dam Process

In this section we present an EA developed for the dam scenario using ArchiMate modelling language. We utilized the notion of views and viewpoints implemented in ArchiMate [38]. Views represent different abstractions on the repertoire of models composing the EA and are intended for particular problem domain, system aspect or stakeholder as well as interrelation of those. A view is specified with a viewpoint which determines respective concepts, analysis and visualization techniques. To specify multilevel architectural abstraction for event escalation we selected the *layered viewpoint*, which embodies all model layers considered in ArchiMate and expresses coherence between layers in terms of services they use and provide. The main goal of the layered viewpoint is to provide EA model overview in one diagram.

Figure 4.6 depicts the EA model view for our sample process of on demand power production with supervision. To produce the model we used Archi[3], a free, open source, cross-platform tool and editor to create ArchiMate models. There are two types of the layers in the diagram: the service layer and the dedicated layer, that includes the infrastructure, the application, the process, and the actors/roles layers. The layers essentially are groups where elements are joined together using the "grouping" relationship. Each dedicated layer offers a set of services (integrated in a corresponding service layer) to the upper dedicated layer. Structural elements within a dedicated layer are connected to the services they provide by means of "realization" relationship, and with "used by" relationship to the services they use.

As input for the model development we employed the dam case study architecture described in Section 6.2 of [26]. Considering that the goal of this example is to expose multi-layered view of the hydroelectric power plant system rather than to produced a detailed EA for the dam infrastructure, we kept the reasonable level of detail allowing us to reveal links between multiple layers. On the upper layer we placed the dam control station operator as the main system user. Services provided by the process layer

[3] http://archi.cetis.ac.uk/

(external business services) permit the operator who has administrative privileges to initiate a discharge request, authorize requests generated by other actors such as power plant from the previous section and supervise dam operations, in our example, related to the change of gate's state. These services are facilitated by the business process "On demand supervised electric power production" which is triggered by an external event when the demand for power load in the power plant alters (increases or decreases). The process actions follow the description in Section 4.1: (1) the plant requests the water discharge towards the turbines required to produce electric power; (2) the control station displays the discharge request for conformation awaiting authorization from the operator, (3) issues corresponding gate command (OpenGate, CloseGate) related to the confirmed request, (4) monitors the gate executing the command and communicates status information to the control station operator. The power plant, the control station and the gate represented in the diagram as roles associated with the process.

The process layer is supported by external application services provided by application components and internal services. In accordance to the available scenario description we singled out four main aggregated systems: control system, visualization system, SCADA system and physical access control system. The control system covers various system components and applications running at the dam control station that allow the operator to monitor the state of controlled processes, modify settings and supervise automatic control operations. The control system enables services of user authentication, device control, online processing and analysis of real-time and stored data. Similarly the visualization system aggregates components located at the visualization station of the dam that are responsible for accumulating and displaying historical information. The SCADA system retrieves sensor data from the monitored infrastructure and provides a two-way information and data delivery service for the control and the visualization system including communication of commands to lower-level control devices via command service. The physical access control system realizes services required for Radio Frequency Identification (RFID)-based access control in the dam control center.

The lower layers of the architecture concern infrastructure and infrastructure services used by dam applications. Elements situated in this layer are connected with non-specific "association" relationship as the focus of this example is the interlayer dependencies. For the same reason internal services are not shown in the diagram. The needs of the control system are served by the control node where Human Machine Interface (HMI) server, applications server, analysis server and online database are located. The visualization node is dedicated to running historian and user interface services based on apache server, visual applications server and historical database. The control node and vizualization node are associated with the internal LAN used for data communication. In contrast with the case study architecture shown in [26] we allocated in the internal Local Area Network (LAN) a separate node (access control node) hosting access control system and used in RFID validation. The internal LAN is isolated from the external LAN. The firewall element implements segregation strategy and protects devices on the heterogeneous network of the dam by filtering communication packets. Network and firewall services are considered as internal infrastructure services. The external LAN connects the MTU or the SCADA server to the control and visualization nodes. The MTU in its turn communicates with the monitored infrastructure by means of the control network designated in the figure as WAN. The control network connects the supervisory control level to RTUs and other lower-level control modules. RTUs located at remote field sites enable data acquisition and control of sensor deployment monitoring critical physical parameters to support SCADA and control system operations. The chain of communications between the MTU, the RTUs and field devices is the basis for the physical measurement service. The command execution is provided by the SCADA server.

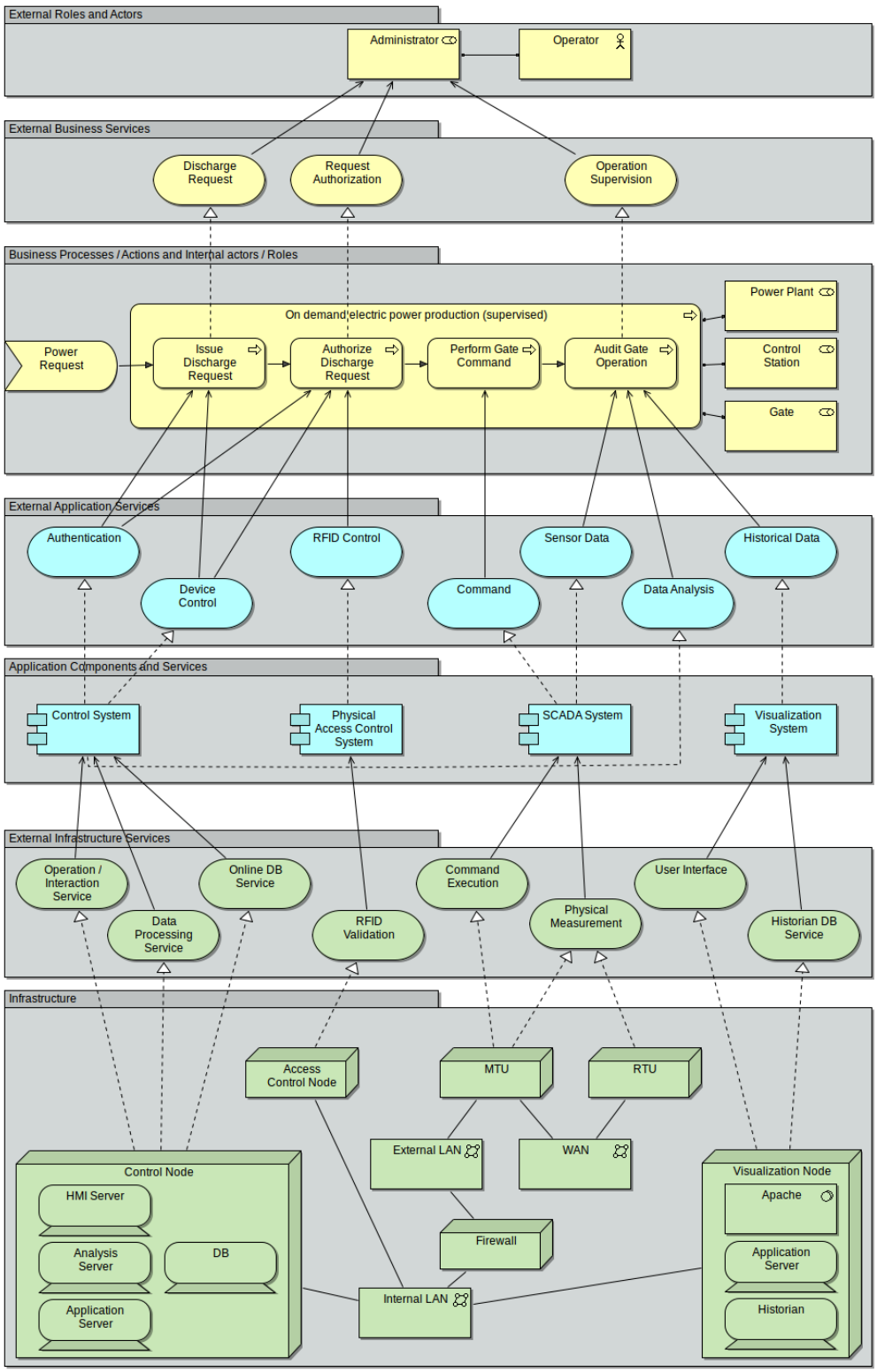

Figure 4.6: Enterprise Architecture Model for the Dam Scenario

4.3 Events Escalation using Dependency Graph & Extended Indicators

In this section we show how to escalate security-related events using extended indicators and dependency graphs on the example of the dam process "On demand power production with supervision" described in Section 4.1. For the sake of consistency with the previously reported results we continue to explore misuse cases discussed in [27] and in [28]. We consider the dam sabotage scenario that deals with an insider attack realized by a maintenance employee (an attacker). The aim of the attacker is to sabotage the dam system and blackmail the provider. He/she can enter the control center producing his/her personal RFID-badge and log in into the dam visualization and control system with stolen dam administrator's credentials. The attacker accesses historical database and downloads confidential data, causing abnormally high traffic rates. The unusual user behavior is used to detect unauthorized database access.

We formulate the general security objective for the considered scenario as *authenticity*, i.e. any operation within the hydroelectric power plant must be carried out by authorized personnel and result from genuine data. We can verify authenticity through *authentication* which includes the proof of identity. In our case a dam administrator must produce an RFID-badge and administrator's password to get authorized access to the control center and the control station respectively. In order to derive security requirements for entities involved in the dam process the security objective needs to be propagated through abstraction levels of the EA model. Using dependencies between elements of the model we built a hierarchy of authentication requirements that represents a mapping of the security objective to the specific domain (model element). To identify the dam sabotage we use the extended IM. The detailed description of the described steps is presented in the following sections.

4.3.1 Creation of Extended Indicators for Dam Process

In creation of extended indicators for the unauthorized access we follow the procedure from [27].

Extended Abstract Indicator The informal description of problem statement.

- `nameAndDescription` **Name**=UnauthorizedAccess
  Description=Monitoring IP traffic rates for critical servers identified with IP addresses (IP1 for the historical database, IP2 for the dam control system).
  $\longmapsto$ If it is triggered by exceeded traffic threshold, the condition "An administrator is present" is verified.
  $\longmapsto$ When the threshold is exceeded, an alarm is triggered, or the event is written to a log file.
- `eventStreamProperty` Monitor whether the traffic to the historical database is normal.
- `condition` Verify whether an administrator is present.
- `action` If `condition` is true $\longmapsto$ trigger alarm, otherwise write the event to log.
- `securityPertinence` Confidentiality of dam historical data, integrity of dam configuration, authorized access.
- `AssetReference` Historical database, dam control system

Extended Real Indicator The formal specification of a real indicator derived from the abstract indicator to be parsed by a security system component (e.g. SeSA, Section 3.2). The Listing 4.1 defines the indicator that enables detection of abnormally high access rates to the historical database.

```
Indicator Name=UnauthorizedAccessHistorian {
    :on EventStreamProperty Name=HistorianAnomaly
        :Extractor
            :EventChannel Name=IPStreamToIPX Fields=[(sourceIP, ipaddress), (
              FIXEDdestIP,ipaddress), (traffic,double)] ;
            :Parameter Name=trafficTuple Type=Tuple(double,ipaddress) Value=(
              IPStreamToIPX.traffic,sourceIP);
            :Function Operator=avg(trafficTuple[0],3600s) InputParam=[
              trafficTuple] OutputParam=avgTraffic;
            :Parameter Name=avgTraffic Type=Tuple(double,ipaddress)
        ;
        :Criterion
            :Function Operator=>(avgTraffic[0],FIXEDmaxthreshold) InputParam=[
              avgTraffic[0]] OutputParam=aboveThreshold;
            :Parameter Name=aboveThreshold Type=boolean;
            :Decision exceedThreshold;
        ;
        :ReturnValue avgTraffic;
    ;
    :Parameter Name=FIXEDmaxthreshold Type=double Value=??;
    :Parameter Name=FIXEDdestIP Type=ipaddress Value=IP1;
    :if Condition TypeOfQuery=SQL
        :ConditionQuery IF SQL="SELECT Employer FROM PhysicalPresenceTable WHERE
           Role='Admin'" != Null THEN Action[0] ELSE Action[1];
        :DatabaseInformation IPAdress=a.b.c.d PortNumber=e Type=mySQL;
    ;
    :do Action [NotificationAction=mailto:alarms@damcompany.com("Alarm=Traffic
      to IP=a.b.c.d exceed threshold."), KnowledgeUpdateAction=
        writeToLogFile("Traffic to IP=a.b.c.d exceed threshold: "+ avgTraffic)]
    ;
    :why SecurityPertinence
        SRequirements=[confidentiality(dam history data, administrator group),
          authenticity(administrator)]
        Assumptions=["Admin credentials required to access database.", "Logging
          in into system at control center requires physical presence."]
        Threats=["Unauthorized access to historical database."]
    ;
    :what AssetReference Name=DamModelD412 Type=List
        :Instance Name=Historian
            :Location Name=VisualizationNode;
            :Owner Name=Epsilon;
        ;
        :Instance Name=ControlSystem
            :Location Name=ControlCenter;
            :Owner Name=Epsilon;
        ;
    ;
}
```

Listing 4.1: Extended Indicator to Monitor Unauthorized Access to Historical Database

4.3.2 Definition of Dependency Graphs for Dam Process

The general definition of a dependency graph derived from the multi-level EA model is introduced in Section 4.2. In the dam example depicted in Figure 4.6 we do not distinguish escalation routes dependent on different types of relationships. Instead we employ the concept of derived relationships in ArchiMate and replace "used-by" and "realization" relationships with "association" by application of the abstraction rule for structural relationships [9]. Example graphs defined using this rule are presented in Figure 4.7.

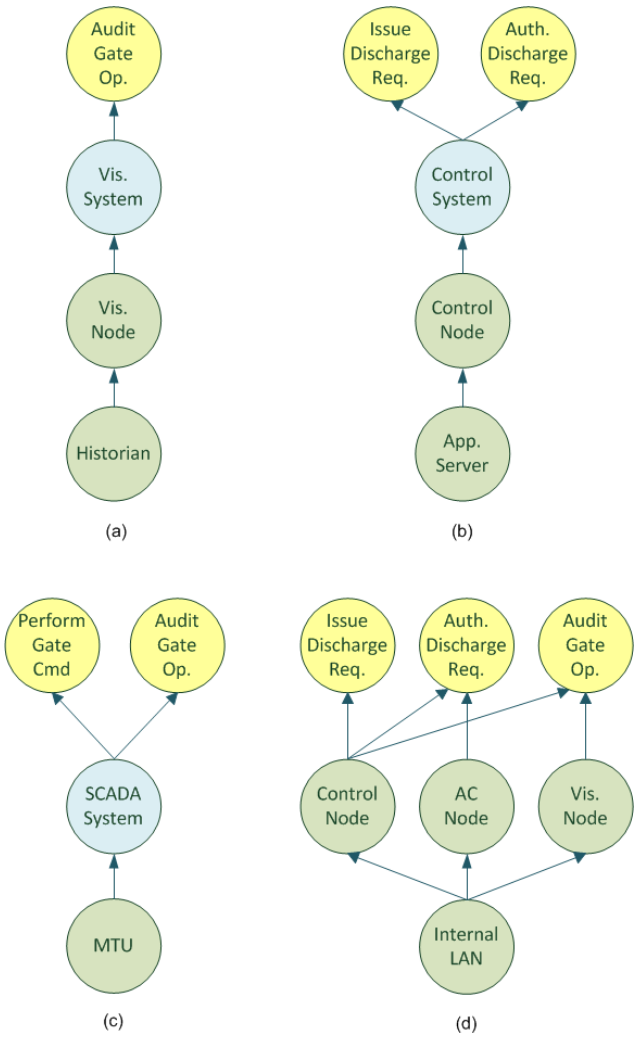

Figure 4.7: Examples of Dependency Graphs

To create the dependency graph (a) we derived a dependency relationship between the historian and the visualization system by replacing "realization" and "used-by" relationships joining at the service element "Historian DB Service". The dependency between the visualization system and the audit gate operation activity was derived through the historical data service. Other dependency graphs arise from the similar course of reasoning. In the graph (b) the application server provides the operation service, i.e. execution of authorized commands. The audit gate operation activity is served by the analysis server and

thus is not included in the graph. The graph (c) reveals that two business activities depend on the SCADA system. A more complicated case involving network environment is presented in (d). From security point of view it implies that the disruption of the internal LAN might paralyze the power production process. Note, that the graphs (b), (c) and (d) produce several escalation routes for security-related events.

4.3.3 Event Escalation for Dam Process

We demonstrate event escalation procedure introduced in Section 2.1 with two examples of unauthorized access. In the first example we show how a security incident can be traced through a hierarchy of security requirements built upon a dependency graph. The second uses a dependency graph to propagate an alert generated by a triggered indicator created in Section 4.3.1 between entities in different abstraction levels.

Exampe 1 We utilize a dependency graph depicted in Figure 4.7, (b). The security objective defined for the dam process of on demand power production is authenticity. Using the dependency graph we propagate this objective to lower abstraction levels of the EA. Thus for activity "Issue discharge request" we define the corresponding requirement as "discharge request is authentic", and for the control system and control node the requirement "administrative authorization" must be met. Figure 4.8 illustrates one of the escalation routes resulting from the graph (b) with the described requirements hierarchy. We assume that the application server within the control node is infected with a backdoor trojan program which allows to get remote administrative access to the host. This simple network attack violates the security requirements: a user (attacker) can log in into the server without administrator's credentials.

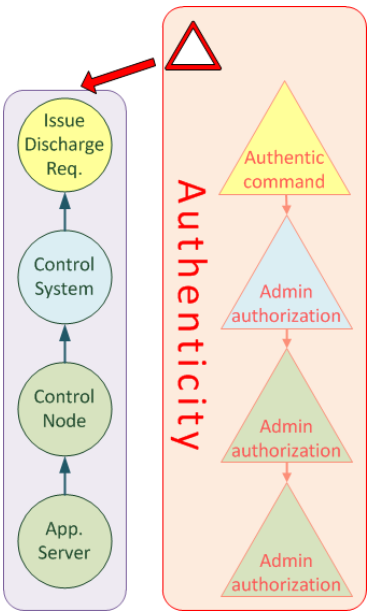

Figure 4.8: Dependency Graph and Requirements Hierarchy

An antivirus software detects the trojan and generates an alert. The alert is escalated through the dependency graph and causes cascade violations of authenticity property on all abstraction levels. As the access to the control system application cannot be considered secure the validity of the discharge request

issued using this application is not ensured. Before execution of this command the incident needs to be investigated and the command source must be verified.

Exampe 2 As an input for this example we take the indicator *UnauthorizedAccessHistorian* described in Section 4.3.1 and a dependency graph depicted in Figure 4.7, (a). We assume that the security incident defined by the indicator occurred, a disgruntled employee without privileges logged in into the historical database with stolen credentials and downloads confidential information. As both "on"- and "if"-part are positively validated due to the observed unusual activity the indicator gets triggered and generates an alert that historical database is compromised. To estimate the impact of the detected incident we analyze the dependency graph related to the historian. Escalation of this alert to the level of the visualization system means that the data displayed by this system and used by the audit gate operation activity are not trusted as the historical database is compromised. On the activity level this fact implies that thresholds required to control gate opening might be not valid and interference of dam personnel is necessary.

4.4 SAMEA Metadata Descriptions in XML notation

We developed the metadata definitions for the example dam process "On demand power production with supervision" using an XML notation compatible with the IF-MAP specification [4]. The selected notation allows to introduce a database service that contains comprehensive information (metadata) about systems (and users) communicating through a network delivered by a publish/subscribe model, where all of the network and security applications can participate in updating the service. Therefore to base the realization of the SAMEA concept on such database service is a very straightforward approach. Indeed it represents a real-time view of an EA, enables automatic validation of "on" (an event stream signature) and "if" (a context condition) parts of the IM and storage of security requirements and possible countermeasures as the metadata information. The notation offers two types of data: identifier and metadata, and one type of relationship: link. All operations and data types are represented as XML documents. The identifier is a unique value of a given type, e.g. network address. The link is an unnamed, bi-directional binding relationship between two identifiers. The metadata is any shared, real time data about network devices, policies, status, behavior and relationships between various systems (e.g. security events, network identity, and network location). The current state of the monitored enterprise can be represented with a graph, where identifiers are depicted as ovals, links as lines, and metadata as squares.

The exemplary graph in Figure 4.9 illustrates the power plant discharge request workflow. For the dam scenario we developed dedicated XML definitions of identifiers, metadata and links described in detail below. This set could be extended to support other operations carried out in a hydroelectric power plant. By presenting the current state of the database service to a supervising employee via a representation of the graph in a GUI, inconsistencies and security violations can be highlighted directly. The requirement described in 4.1 can be expressed by a cascaded condition regarding identifiers and links: "A command-request may only be generated-on a control station with an address that is also associated with a access-request (via an access-request-address link) that is associated with a dam operator identity (via an authenticated-as link)". Metadata can also be utilized by a condition: "A resource-request may not be linked to an unknown power plant address via a generated-by". If a condition in an expression is not matched (e.g. an access-request is not associated with an operator via an authenticated-as link), this violation can be highlighted in the graph presented to the supervising employee. This methodology also enables the supervisor to assess the state of related metadata in case of a violation or inconsistency ad-hoc.

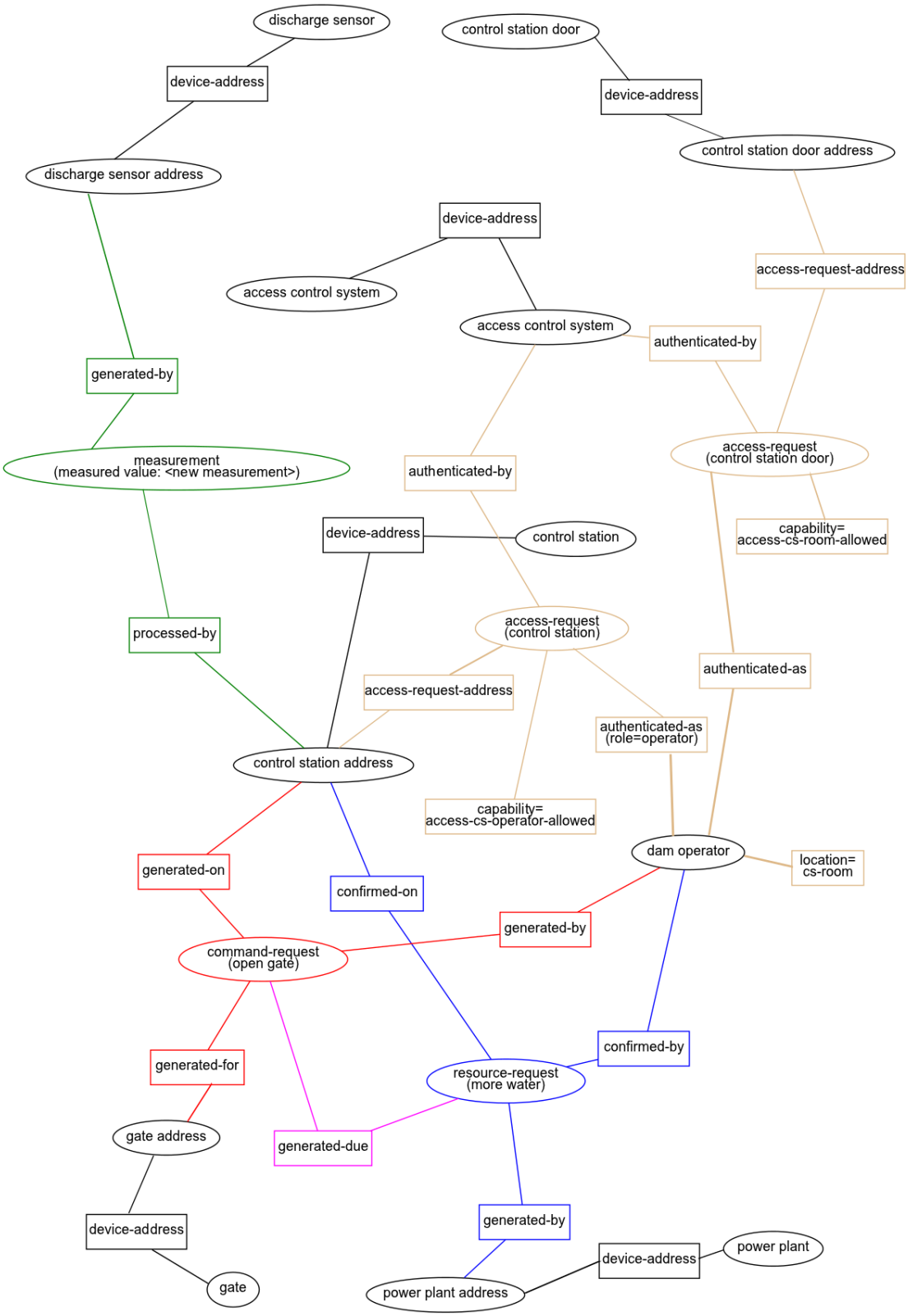

Figure 4.9: Exemplary Graph for Power Plant Discharge Request

4.4.1 address

The names of specific addresses identifiers include the type of asset they are identifying (e.g. control station address), the type of address is implicitly defined by the corresponding asset.

```
<xsd:complexType name="AddressType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
  <xsd:attribute name="type" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="IPv4"/>
        <xsd:enumeration value="IPv6"/>
        <xsd:enumeration value="MAC"/>
        <xsd:enumeration value="other"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="othertype" type="xsd:string"
use="optional"/>
  <xsd:attribute name="layer" use="optional">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="2"/>
        <xsd:enumeration value="3"/>
        <xsd:enumeration value="4"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

Figure 4.10: `address` identifier definition

If type is “other”, the attribute othertype must be used. The layer attribute must be used if type is “other” and should not be used if type is not “other”. The exemplary graph utilizes the general address identifier as proposed in Figure 4.10.

4.4.2 command-request

This identifier represents a command generated-by a dam operator (generated-due to a resource-request, proposed below). If type is “limited” a value must be provided. If a value is provided a unit should be provided. (e.g., “limited”: open by 10%, “unlimited”: open until fully opened). It represents a command generated-by a dam operator (generated-due to a resource-request, proposed below).

```
<xsd:complexType name="CommandRequestType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="action" type="xsd:string" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="open"/>
        <xsd:enumeration value="close"/>
        <xsd:enumeration value="start"/>
        <xsd:enumeration value="stop"/>
        <xsd:enumeration value="move"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="type" type="xsd:string" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="unlimited"/>
        <xsd:enumeration value="limited"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="unit" type="xsd:string" use="optional"/>
  <xsd:attribute name="value" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Figure 4.11: `command-request` identifier definition

The exemplary graph utilizes the command-request identifier as proposed in Figure 4.11.

4.4.3 `resource-request`

This identifier represents a specific resource request generated-by a power plant. A unit SHOULD be provided. The enumeration used are dam-specific. To enable a greater scope of usage for this identifier, additional types of resources could be added to the list or the restriction to the resource attribute could be omitted.

```
<xsd:complexType name="ResourceRequestType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="resource" type="xsd:string" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="water"/>
        <xsd:enumeration value="power"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="type" type="xsd:string" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="absolut"/>
        <xsd:enumeration value="increase"/>
        <xsd:enumeration value="decrease"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="unit" type="xsd:string" use="optional"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
```

Figure 4.12: `resource-request` identifier definition

The exemplary graph utilizes the resource-request identifier as proposed in Figure 4.12.

4.4.4 `measurement`

This identifier represents a specific event metadata corresponding to activity of interest detected on the network / by dam sensors.

```
<xsd:element name="measurement">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="0" maxOccurs="1"/>
      <xsd:element name="value" type="xsd:double"
        minOccurs="0" maxOccurs="1"/>
      <xsd:element name="unit" type="xsd:string"
        minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.13: `measurement` identifier definition

The exemplary graph utilizes the measurement link, as proposed in Figure 4.13.

4.4.5 device-address (Link)

This links a device with a known address, the device utilizes and can be identified by. The optional element interface represents meta information about the corresponding interface on the device the address is associated with.

```
<xsd:element name="device-address">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="interface" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.14: device-address link definition

The exemplary graph utilizes the device-address link, as proposed in Figure 4.14.

4.4.6 generated-due (Link)

This links causal related original identifier, e.g., a command-request is generated-due a resource-request generated-by a power plant. This is a process related link to enhance automatic post-processing.

```
<xsd:element name="generated-due">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.15: generated-due link definition

The exemplary graph utilizes the generated-due link, as proposed in Figure 4.15.

4.4.7 generated-for (Link)

This links a request with the address of a device it is generated-for.

```
<xsd:element name="generated-for">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.16: `generated-for` link definition

The exemplary graph utilizes the generated-for link, as proposed in Figure 4.16.

4.4.8 `generated-by` (Link)

This links a request to the identity (manually) or the address of the device (automatically) it is generated-by.

```
<xsd:element name="generated-by">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="creation-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.17: `generated-by` link definition

The exemplary graph utilizes the generated-by link, as proposed in Figure 4.17.

4.4.9 `generated-on` (Link)

This links a request with the address of the device it is generated-on.

```
<xsd:element name="generated-on">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.18: `generated-on` link definition

The exemplary graph utilizes the generated-by link, as proposed in Figure 4.18.

4.4.10 confirmed-by (Link)

This links a request with the identity it is confirmed-by.

```
<xsd:element name="confirmed-by">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="creation-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.19: `confirmed-by` link definition

The exemplary graph utilizes the confirmed-by link, as proposed in Figure 4.19.

4.4.11 confirmed-on (Link)

This links a request with the address of a device it is confirmed-on.

```
<xsd:element name="confirmed-on">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.20: `confirmed-on` link definition

The exemplary graph utilizes the confirmed-on link, as proposed in Figure 4.20.

4.4.12 processed-by (Link)

This links a event or measurement with the address of the device it is processed-by.

```
<xsd:element name="processed-by">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="publish-time" type="xsd:dateTime"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes"/>
  </xsd:complexType>
</xsd:element>
```

Figure 4.21: `processed-by` link definition

The exemplary graph utilizes the processed-by link, as proposed in Figure 4.20.

5 Conclusion

This document describes a model-based approach to multi-level security monitoring that enables definition of relations between different abstraction levels and different domains and escalation of security-related events from lower to higher abstraction levels, e.g. from a network infrastructure to a web-server or from a SCADA system to a dam control process. It also helps to interpret received security information on different levels and trace security incidents linking together critical assets, security requirements and generated alerts.

The concept of interrelated abstraction levels is expressed with a SAMEA which integrates a security information model and an Enterprise Architecture model to enhance the security analysis capabilities of existing SIEM systems. The SAMEA allows to infer dependencies between elements belonging to different abstraction levels of a EA model in the form of dependency graphs. The obtained dependencies provide a basis for (hierarchical) security requirements propagation and identification of event escalation routes. The SAMEA also enables filtering of events according to their relevance from the perspective of a monitored security property.

The proposed multi-level abstraction concept satisfies the requirements specified in the MASSIF Description of Work (DoW). Escalation of events makes it possible to analyze an event within a different context and correlate it with security information that is not available otherwise. Security mechanisms implemented on separate abstraction levels can benefit from the mutual exchange of security information for increasing efficiency and accuracy of incident detection even in situations when individual security monitors produce incomplete or uncertain information. An architectural and contextual part of the SAMEA can be replaced or a EA model view can be changed to better match the business security goals on conditions that the link between critical assets, security requirements and generated alerts is kept.

The proposed approach can enhance existing EA modelling paradigms with security features, give opportunity to analyze security alerts produced by current SIEM systems in respect to global security objectives and predict, increase efficiency and accuracy of security analysis and control. These features can advance SIEM solutions, real-time CMDBs and other event-driven systems aimed to maintain automated business processes.

Bibliography

[1] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03)*, June 2003.

[2] C.J. Alberts and A.J. Dorofee. *Managing Information Security Risks: The Octave Approach*. Sei Series in Software Engineering. Addison-Wesley, 2003.

[3] S. Buckl, E. M. Ernst, J. Lankes, F. Matthes, and Ch. M. Schweda. State of the Art in Enterprise Architecture Management. Technical report, Technische Universität München, 2009.

[4] Trusted Network Connect. *TNC IF-MAP Binding for SOAP, Specification v2.1, Revision 15*. TCG Published, May 2012.

[5] L. Coppolino, S. D'Antonio, V. Formicola, and L. Romano. Integration of a System for Critical Infrastructure Protection with the OSSIM SIEM Platform: A dam case study. In *SAFECOMP*, pages 199–212, 2011.

[6] J. Eichler and R. Rieke. Model-based Situational Security Analysis. In *Proceedings of the 6th International Workshop on Models@run.time at the ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand*, volume 794 of *CEUR Workshop Proceedings*, pages 25–36. 2011.

[7] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Pubs Co Series. Manning Publications, 2010.

[8] R. Fredriksen, M. Kristiansen, B. A. Gran, K. Stølen, T. A. Opperud, and Th. Dimitrakos. The CORAS Framework for a Model-Based Risk Management Process. In *Proceedings of the 21st International Conference on Computer Safety, Reliability and Security*, SAFECOMP'02, pages 94–105, London, UK, 2002. Springer-Verlag.

[9] The Open Group. ArchiMate 2.0 Specification. Technical report, 2012.

[10] The Open Group. TOGAF Standard Version 9.1. Technical report, 2012.

[11] The Open Group TOGAF-SABSA Integration Working Group. *TOGAF and SABSA Integration. Whitepaper*. The Open Group, The SABSA Institute, October 2011.

[12] S. Grunow. Automated Enterprise Service Bus Based Enterprise Architecture Documentation. Master's thesis, KTH, Industrial Information and Control Systems, 2012.

[13] P. Gustafsson, D. Höök, U. Franke, and P. Johnson. Modeling the IT Impact on Organizational Structure. In *Proceedings of the 13th IEEE international conference on Enterprise Distributed Object Computing*, EDOC'09, pages 12–21, Piscataway, NJ, USA, 2009. IEEE Press.

[14] R.R. Henning. Use of the Zachman Architecture for Security Engineering. In *Proc. 19th NIST-NCSC National Information Systems Security Conference*, pages 398–409, 1996.

[15] M.-E. Iacob, H. Jonkers, and M. Wiering. Towards a UML profile for the ArchiMate language, ArchiMate D2.2.3c. Technical report, Telematica Instituut (TI), Leiden Institute for Advanced Computer Science (LIACS), 2004.

[16] F. Innerhofer-Oberperfler and R. Breu. Using an Enterprise Architecture for IT Risk Management. In *ISSA*, pages 1–12, 2006.

[17] Richard Kissel. *Glossary of Key Information Security Terms, NIST IR 7298, Revision 1*. National Institute of Standards and Technologies (NIST), February 2011.

[18] M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication, and Analysis*. Springer, 2005.

[19] M. M. Lankhorst, H. A. Proper, and H. Jonkers. The Architecture of the ArchiMate Language. In *Enterprise, Business-Process and Information Systems Modeling*, volume 29 of *Lecture Notes in Business Information Processing*, pages 367–380. Springer Berlin Heidelberg, 2009.

[20] AlienVault LC. AlienVault Unified SIEM, System description. Technical report, 2011.

[21] D.C. Luckham. *The Power of Events: an Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2002.

[22] SC Magazine. *The Complete Guide to Log and Event Management, Whitepaper*. http://whitepapers.scmagazine.com/, June 2012.

[23] SC Magazine. *The state of SIEM, Ebook*, 2012.

[24] M. Nicolett and K.M. Kavanagh. *Magic Quadrant for Security Information and Event Management*. Gartner RAS Core Research Note G00212454, May 2011.

[25] MASSIF project consortium. *MAnagement of Security information and events in Service InFrastructures - Annex 1*. FP7-257475 MASSIF European project, April 2010.

[26] MASSIF project consortium. *D2.1.1 - Scenario requirements*. FP7-257475 MASSIF European project, March 2011.

[27] MASSIF project consortium. *D4.1.1 - Event Modelling Concept*. FP7-257475 MASSIF European project, September 2011.

[28] MASSIF project consortium. *D4.2.1 - Formal Specification of Security Properties*. FP7-257475 MASSIF European project, September 2011.

[29] MASSIF project consortium. *Architecture Document*. FP7-257475 MASSIF European project, April 2012.

[30] MASSIF project consortium. *Enhancing Security and Trustworthiness with Next-Generation Security Information and Event Management, Whitepaper*. FP7-257475 MASSIF European project, June 2012.

[31] P.J. Regan. Dams as systems - a holistic approach to dam safety. In *30th Annual USSD Conference Sacramento, California*, 2010.

[32] R. Rieke, J. Schütte, and A. Hutchison. Architecting an Security Strategy Measurement and Management System. In *Model-Driven Security Workshop in conjunction with MoDELS 2012 (MDsec 2012), Innsbruck, Austria*. 2012.

[33] J. Schekkerman. *How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating Or Choosing an Enterprise Architecture Framework*. Trafford, 2003.

[34] G. Sharon and O. Etzion. Event-Processing Network Model and Implementation. *IBM Systems Journal*, 47(2):321 –334, 2008.

[35] J. Sherwood, A. Clark, and D. Lynas. *Enterprise Security Architecture: A Business-Driven Approach*. CMP Books, 2005.

[36] T. Sommestad, M. Ekstedt, and P. Johnson. A Probabilistic Relational Model for Security Risk Analysis. *Computers & Security*, 29(6):659–679, 2010.

[37] K. A. Stouffer, J. A. Falco, and K. A. Scarfone. *Guide to Industrial Control Systems (ICS) Security - Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC). NIST Special Publication 800-82*. National Institute of Standards and Technology (NIST), June 2011.

[38] H. ter Doest, M.-E. Iacob, M. Lankhorst, D. van Leeuwen, and R. Slagter. Viewpoints Functionality and Examples, ArchiMate D3.4.1a v2.6. Technical report, Telematica Instituut (TI), 2004.

[39] R. Usmany. Conceptual modeling for business process semantics. Technical report, Radboud University Nijmegen, Faculty of Science, July 2012.

[40] R. Winter and R. Fischer. Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In *Enterprise Distributed Object Computing Conference Workshops. EDOCW'06. 10th IEEE International*, page 30, October 2006.

[41] J. A. Zachman. A Framework for Information Systems Architecture. *IBM Syst. J.*, 38(2-3):454–470, June 1999.

