Aggregation of security attributes based on the granularity level of the system

Thesis Submitted in Partial Fulfilment of the Requirements for the Degree

of

Master of Science

to the Department of Computer Science of Freie Universität Berlin

by

Artemij Voskobojnikov
Student ID: 4557770
voskobojnikov.artemij@gmail.com

First Reviewer: Prof. Dr. Jörn Eichler
First Reviewer: Prof. Dr. Marian Margraf

Berlin, <Datum>

Affirmation of independent work

I hereby declare that I wrote this thesis myself without sources other than those indicated herein. All parts taken from published and unpublished scripts are indicated as such.

Berlin, <date>

(Artemij Voskobojnikov)

Acronyms

List of Figures

Abstract

Contents

1 Introduction

Security concepts can be used to capture the interacting system components, potential threats and countermeasures.
For large information systems such concepts can become very large because of the number of the involved sub-systems/components. Interconnectivity and interdepence amongst components may increase the overall system complexity and it might be therefore difficult to detect all potential impacts [2]. Methods for system abstraction that address this problem already exist. The abstraction here is the creation of representation layers which only reflect relevant properties of a system and therefore provide a better level of understanding for the respective user [9]. This for example can be achieved by different projections which reflect different granularity levels of a system displaying different levels of details [13].
In the security context such projections could be used to focus on the security or insecurity of certain sub-systems. Security attributes of components could thus be viewed separately and the security risk for a respective component could be derived. This might become especially useful when the security concept is incomplete or only partially available. An aggregation of security attributes based on the chosen projection will become possible. The system structure could then be used to derive security attributes for components that previously had none. Thus, potentially new information might become processable.
Aggregation methods for security attributes have already been suggested by researchers, e.g. transformation rules for security requirements by Menzel et al. [6] or aggregation rules for attack graphs by Noel et al. [7]. None of those methods take granularity levels or general system hierarchy into account whereas the goal of this thesis is to provide an approach which makes it possible for a user to select a sub-system of interest, i.e. a projection which reflects a certain granularity level and provides the corresponding security attributes. The relevant attributes as well as dependencies and possible aggregations will be shown to ensure an overall complete picture of the selected sub-system. This information can then be used to assess and improve the security level of the selected projection or its dependencies.

2 Background

Prior to addressing the actual approach and implementation some concepts and terms have to be introduced. Firstly, the term *security concept*, as it is used throughout the thesis, is being described. A definition of *granularity levels* and system abstraction follows. Lastly, a section covers *model transformations* and *aggregation rules* on security attributes.

2.1 Security

Morrie Gasser [3] published a book in 1988 providing solutions for computer scpecialists interested in computer security. The term *Computer Security* usually dealt with three aspects:

- Prevention of theft or damage to the hardware
- Prevention of theft or damage to the information
- Prevention of service disruption

With the invention of the Internet the (personal) computers got interconnected and the sharing of data became prevalent and thus data security.
Nowadays security is seen as a process. No combination of products are guaranteed to make a system secure since they are only as secure as the people configuring them [14].
Information systems consist of hardware, software and data and it is of high importance for the respective stakeholder to secure the object of interest. When speaking about securing systems or components we look at two different terminologies, *Vulnerabilities* and *Threats*. A *vulnerability* is a system weakness, be it in implementation or design, that can be potentially exploited by an adversary. Without the intent of exploiting it the vulnerability has no further effect on the system.
A *threat* however is a set of circumstances that has the theoretical potential to cause harm by exploiting a specific vulnerability [14]. An example to demonstrate the differences follows:

- **Vulnerability**: Sending sensitive data over an unsecured channel
- **Threat**: Exploiting the unsecured channel by eavesdropping and gathering the sensitive data

The definition above only covers information security, which in the scope of this thesis, is insufficient. Here, we define security as the preservation of confidentiality, integrity and availability of assets, where assets can be either physical or logical.

2.1.1 Security Architecture

2.1.2 Common Criteria

The overview of TOGAF showed the importance of security in corporations. The following section will present a way of modeling security concerns for an asset of interest.
Common Criteria proposes an evaluation by using a so called *Security Target* (ST), a construct that encapsulates the *Target of Evaluation* (TOE), threats to the TOE and countermeasures [12]. The goal of the evaluation is to show that the used countermeasures are sufficient to counter potential threats and thus implying that the TOE is sufficiently protected.

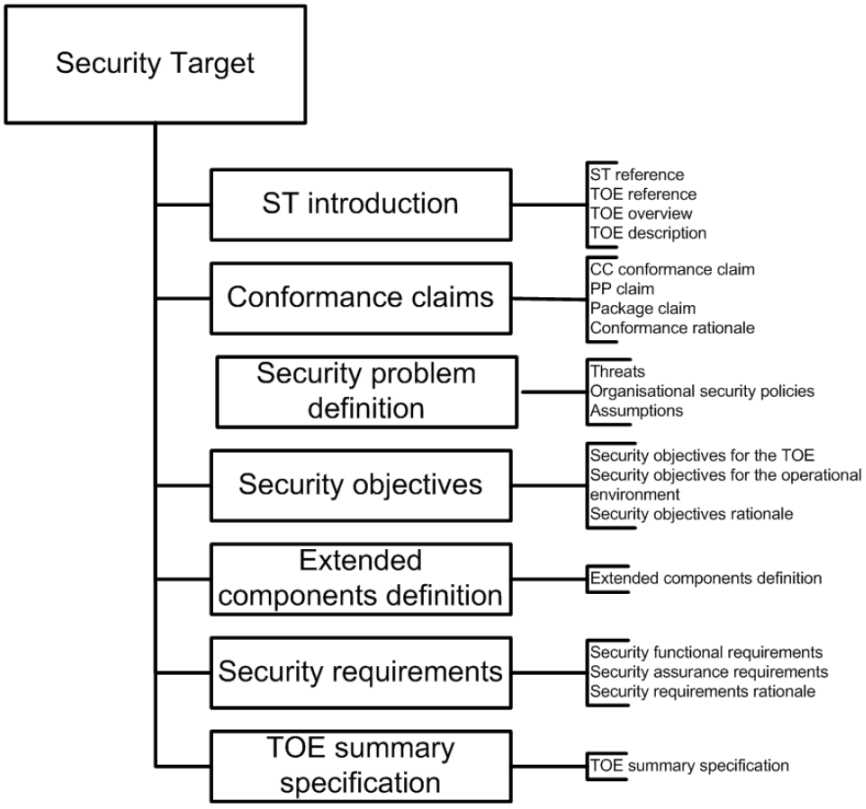

Figure 1: Overview of the Security Target contents

A description of all the contents of a ST is unnecessary here and only the key security attributes of a ST that will be used to construct a *Security Concept* (Subsection 2.1.3) are being introduced.
The *Security Problem Definition* defines, as the name suggests, the security

problem that is being addressed. Apart from containing guidelines and assumptions it contains *Threats* which are *„[...] adverse actions performed by a threat agent on an asset“* ([1], p. 66).
A *Security Objective* is an abstract solution to the previously defined security problem. There exists a possibility to divide the *Security Objectives* into part wise solutions, one being the *Security Objectives for the TOE* and the other being the *Security Objectives for the Operational Environment*. Moreover does the ST contain traces showing which objectives address which threats, guidelines and assumptions and a set showing that all threats, guidelines and assumptions are addressed by the security objective.
Security Functional Requirements (SFR) are a more detailed translation of the previously defined *Security Objective*. Despite being more detailed, SFR have to be still independent from specific technical solutions.
Lastly, STs contain a TOE summary specification where it is stated how the TOE meets all the SFRs and how exactly those requirements are met on a technical level.

2.1.3 Security Concept

The term *Security Concept*, as it is defined here, is based on the constructs introduced in the previous chapters, namely *Security Architecture* and *Security Target*. An overview follows.
Assets are the to be secured objects of interest, i.e. TOE according to Common Criteria. *Assets* can be either logical or physical and can be grouped to sets, if needed.
A *Security Goal* (SG) is the equivalent to the *Security Objective*. A valid SG must address an *Asset* and a *Security Goal Class* that defines the actual purpose of the SG. In general the set of *Security Goal Classes* consists of *Confidentiality*, *Integrity* and *Availability* but can also be expanded by further classes such as *Authenticity*.
Threats serve the same purpose as proposed by Common Criteria. They are adverse actions performed by an entity against an *Asset*.
This information is all brought together in *Security Requirements* that are defined in natural language and show the interrelationships between elements. A *Security Goal* has to be mentioned as well as an *Asset* and a *Threat* against which the object of interest should be protected.
Lastly, *Controls* are the technical measures that counter or minimize the *Threats*.
The Table 1 depicts the relationships between all the security attributes:

Name	Contains	Description	Example
Asset	-	Digital or physical object of interest that should be secured	Sensible user data
Security Goal Class	-	Defines the purpose of the Security Goal	Confidentiality of sensible user data
Security Goal	Security Goal Class, Asset	Defines the security objective	Confidentiality of sensible user data shall be protected
Threat	Asset	Adverse action against an Asset	Eavesdropping of sensible user data
Security Requirement	Asset, Security Goal, Threat	Security Objective in natural language	The Confidentiality of sensible user data shall be protected against eavesdropping
Control	Threat	Measure to minimize or mitigate the Threat	Encryption of sensible user data with AES-256 to prevent eavesdropping

Table 1: Elements of a Security Concept

2.2 Modeling

To ensure a viable solution one has to think of a representation of real life systems. *Models* can be used to achieve this by depicting the key properties and processes of a certain system. According to Ed Seidewitz [11] a model is a *„set of statements about some system under study“* with the statements being either correct or incorrect.
A system modeled using the Unified Modeling Language (UML) serves as an example. In this case such statements could be made on the relationships between classes and would only be correct if they are consistent with the actual structure of the respective system under study (SUS), i.e. the described (modeled) relationships do indeed exist.
In our case we would try to create a model that reflects the security attributes and their interrelationships in a SUS. This interpretation of a model is key because only then the model is given a meaning [11].
A definition of a model is not enough. A *metamodel* has to be clearly defined to verify whether a model is conform or not, i.e. whether a security concept instance is conform to its security concept metamodel. The following figure shows the interrelationships.

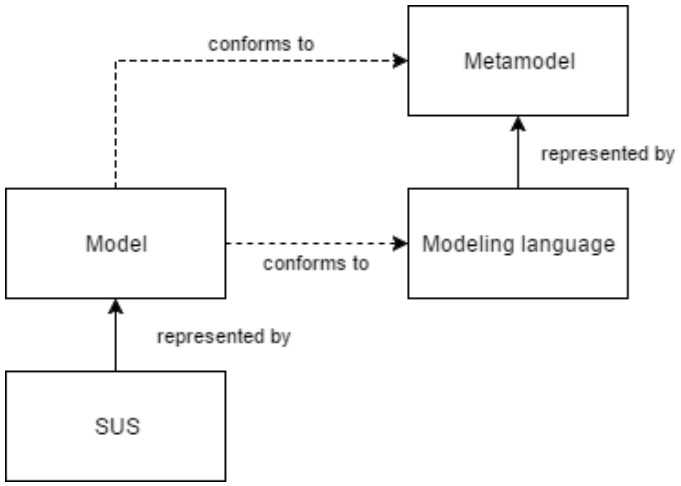

Figure 2: Relationships between Model and Metamodel

A security concept of a SUS would be modeled in a modeling language, e.g. UML which is a representation of its own metamodel. At the same time the security concept would be conform to its metamodel. This conformity, be it the security concept or the modeling language, is needed for a model to be considered valid.

2.2.1 Model Transformation

The Meta Object Facility (MOF) [8] is a standard metametamodel proposed by the Object Management Group (OMG) and captures the relationships between models in a three-layered architecture consisting of M1, M2 and M3. Models (M1) are representations of systems and are expressed in a modeling language M2, e.g. UML as mentioned in the previous section, which is conform to a so called metamodel. Metamodels themselves are also expressed in a metamodeling language which is conform to a metametamodel (M3).
These architecture levels can be found in the *model transformation pattern* by Jouault et al. [4] which can be seen in Figure 3.

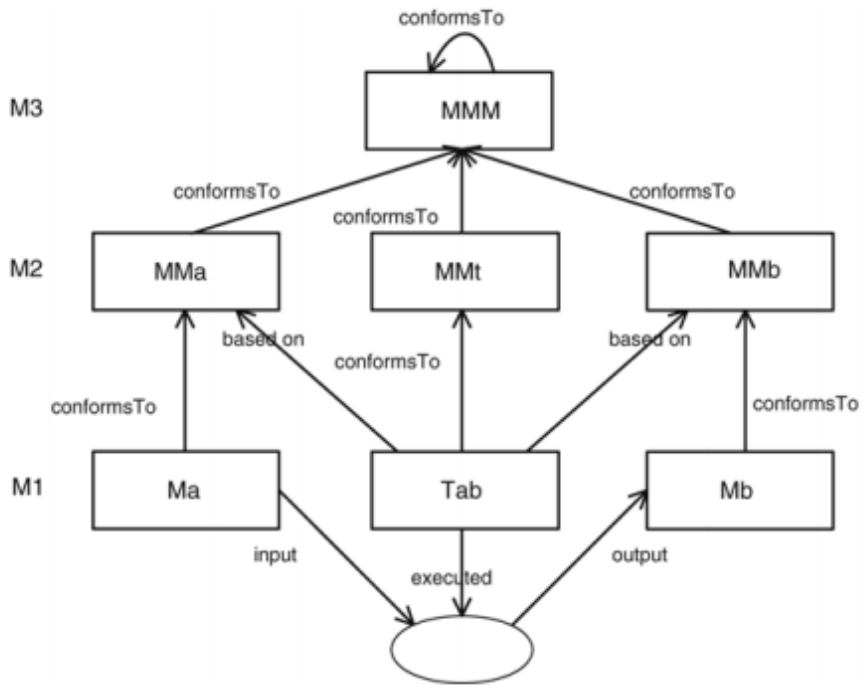

Figure 3: Model transformation according to Jouault et al.

Here a source model Ma is being transformed into a target model Mb using a transformation language. Both models and the transformation language are conform to their respective metamodel which is the traditional understanding of a model transformation but is unnecessarily complex for the security concept transformation.
Throughout the introduction and the background chapter the derivation of security attributes based on structural properties of a system of interest was

mentioned. Given a model M this derivation can be seen as alteration of M and therefore as a *model transformation*. The resulting model M' is different to M, both however, are conform to the same metamodel MM. The transformation is being shown in Figure 5.

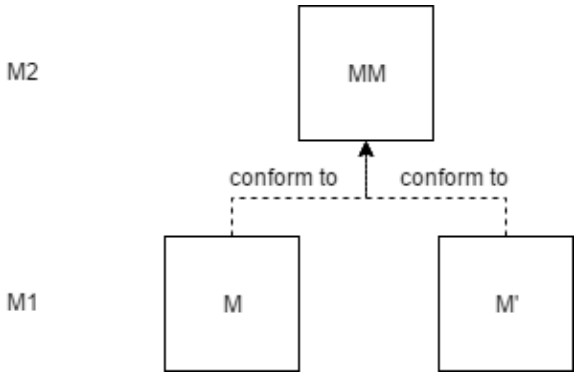

Figure 4: Model transformation

The definition of a transformation T, or better a *transformation rule set*, that alters a model M is the main goal of this thesis.
Prior to the actual rule set definition one final concept has to be introduced. A user-selected *Granularity level* serves as a second input in the model transformation.

2.2.2 Granularity Levels

Information systems are often complex because of the number of interconnections and interdependencies between components and therefore it might be difficult to assess potential impacts and risks of a system [2].
One logical goal would be to decrease the overall complexity to enable a better risk assessment. *System abstraction* tries to achieve this by reducing the level of details [2]. Branagan et al. differentiate between two possibilities, *Whole-part decomposition* which decomposes a system into smaller subsystems and *Distinct development perspectives* which focuses on certain parts of a system depending on the current development perspective.
In this thesis the focus will be on the *Whole-part decomposition* even though an adaption of development perspectives is certainly possible. The main difficulty would be the definition of such perspectives in the security context because they would be highly dependent on the respective security analyst. The perspectives would not be unambiguous.
Therefore the change of the level of detail, i.e. the granularity level, will be achieved by uniting or decomposing components of a system of interest. By

decomposing a larger system into smaller subsystems one could focus on only specific security attributes and dismiss others.
A user will select a certain granularity level, i.e. a certain set of components, as an input to the transformation function T as mentioned in 2.2.1. Together with the defined rule set a valid model M' will be generated which to be considered valid has to be conform to its metamodel MM.

3 Related Work

4 Approach

This section presents the approach addressing the previously mentioned goal of a model transformation based on the user-selected granularity level of a system of interest. Firstly, the security concept metamodel will be thoroughly described, each element of the metamodel will be put in the security context and advantages and disadvantages of such an interpretation will be highlighted.
The second part will deal with the actual transformation rules. Model transformations will be mathematically defined and the transformation rules for each element of the model will follow. Aside from the solution possible edge cases will be presented and evaluated.

4.1 Security Concept

The following metamodel is based on the security elements mentioned in Section 2.1.3. It shows the interconnections between elements and adds restrictions. This metamodel serves as a base enabling the creation of model instances capturing the relations between components/assets of a specific SUS. It also provides a security context and the option for the user to select a granularity level.

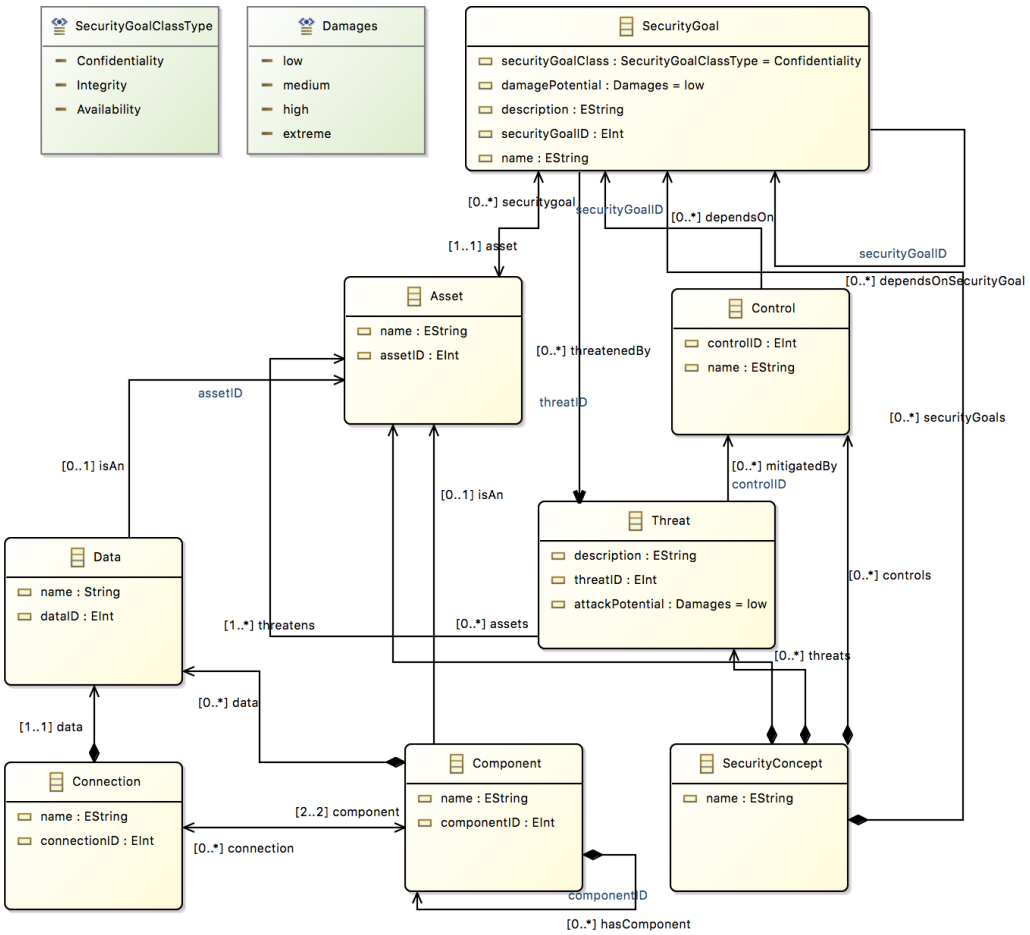

Figure 5: Security Concept Metamodel

In the figure above all the core elements *Security Goal*, *Asset*, *Control* and *Threat* are pictured. All of these elements are part of a *SecurityConcept*, which in this model will be simply identified by a name.
SGs have a *Security Goal Class* attribute which describes the purpose of each SG. The *Damage Potential* attribute indicates at the importance of a Goal, i.e. how important it is to secure a certain asset. The higher the damage potential the higher the impact if the security of an asset is breached.
One key aspect of this metamodel is the dependency between SGs. A SG is dependent on another SG if both belong to the same asset and have the same security goal class. These dependencies, amongst others, have to be considered during potential transformation steps (Section 4.6).
Each SG belongs to exactly one asset whereas an asset itself can have unlimited SGs. In this thesis both physical and virtual components can be considered an asset. Both *Data* and *Component* can be assets according to

the metamodel.
There will be two different types of data. For once *processed data*, i.e. data that is being processed or kept in storage by a specific component. On top of that *transmitted data* will be considered separately since the transmission channel itself can be seen as an asset. The resulting interconnections are shown in the following figure:

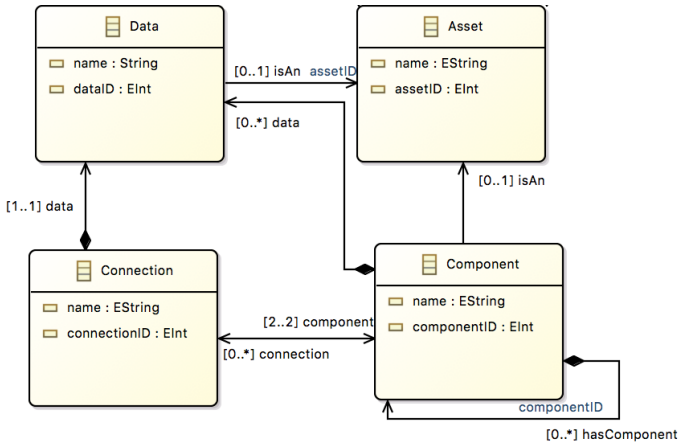

Figure 6: Two different interpretations of data

As already mentioned data can be viewed as being processed and transmitted. Therefore an element *Connection* was added. A connection is the transmission channel between two components. It must have an associated data. The processed or stored data however can be directly associated with a component. In both cases data can be viewed as an asset. There is no possibility to assign a connection as an asset, the reason being that the transmission medium itself, i.e. the cable, wire, is rarely an object of interest but more so the data which is being transmitted.
Lastly the selection of *Granularity Levels* by users should be enabled. Instead of having two different input models, one security concept model and one model depicting the system structure, one can reproduce the structural properties by adding a reference to the component element.

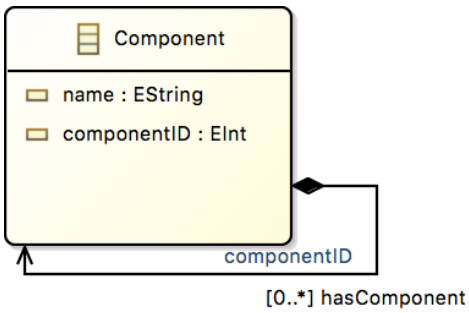

Figure 7: Structural information

Having this extra reference one can create infinitely deep structural dependencies within the model. Therefore the actual transformation will only require one model instead of two separate ones.

4.2 Metamodel elements and their security properties

To put the different elements of the metamodel into a security context one has to clearly define the Security Goal Classes for possible assets. Interpretations of the classes are not unambiguous and it is necessary to discuss how security attributes propagate through different abstraction layers and what kind of impact interrelated components on different layers have on each other.

4.2.1 Components

Before introducing the transformation rules one has to look at the different kinds of components that can be potentially found in a model instance. Even though the model element remains the same (*Component*) a distinction which is made here is necessary because the interpretation of Security Goal Classes differs depending on the component type.

Physical Component

Physical components are components that can be accessed physically, e.g. computers, servers, switches etc. Since those can be accessed physically and therefore be manipulated physically one has to define the Security Goal Classes accordingly.

1. **Confidentiality** - Will be *undefined* for physical components and will only hold for data stored/processed on those components

2. **Integrity** - Ensured when the component has not been altered by an adversary in any way; can be broken by an adversary having physical access

3. **Availability** - Ensured when the component is able to carry out its designated task; can be broken by an adversary having physical access

Virtual Component

Virtual components cannot be directly accessed physically. Examples would be virtual machines, virtual switches etc. The classes are similar to the physical counterpart except from the breach of the respective class.

1. **Confidentiality** - Will be *undefined* for virtual components and will only hold for data stored/processed on those components

2. **Integrity** - Ensured when the component has not been altered by an adversary in any way; can be broken remotely by an adversary, i.e. without having physical access

3. **Availability** - Ensured when the component is able to carry out its designated task; can be broken remotely by an adversary, i.e. without having physical access

4.2.2 Data

Similar to the component element the data element can be interpreted in different ways. The distinctions between the different data types are very minor but noteworthy nonetheless. We consider three types of data, data which is being stored by a component, data which is being processed by a component and data which is being transmitted between components. The common terminology is *Data at Rest*, *Data in Motion* and *Data in Use* [5]. The used terms might be new, the distinction between different states of data however can be found in publications going as far back as 1982, e.g. by Dorothy E. Denning [10].

Data at Rest

Data at Rest applies to devices that hold data [5]. An example would be a database containing sensitive information. The database being the *Virtual Component* and the stored data being the *Data at Rest*. One could also define the server with the database as a *Physical Component*.

1. **Confidentiality** - Ensured when the data is protected from unauthorized disclosure at rest
2. **Integrity** - Ensured when the data is protected from unauthorized modification at rest
3. **Availability** - Ensured when the data is available to authorized parties when needed, i.e. authorized parties can access and use the data when needed

Data in Use

Data in Use applies to data that is being processed by a component, i.e. is being used by a service running on a component. An example would be data that is being processed by an API on a server. The API being the *Virtual Component* and the server being the *Physical Component*. The *Data in Use* would be the processed information by the API backend.

1. **Confidentiality** - Ensured when the data is protected from unauthorized disclosure during processing
2. **Integrity** - Ensured when the data is protected from unauthorized modification during processing
3. **Availability** - Ensured when the data is available to authorized parties when needed, i.e. the data is being processed by the service/the respective service is running when needed

Data in Motion

Data in Motion applies to all data transmitted between components. We do not specify any protocols here to keep the definition as broad as possible.

1. **Confidentiality** - Ensured when the data is protected from unauthorized disclosure during transmission
2. **Integrity** - Ensured when the data is protected from unauthorized modification during transmission
3. **Availability** - Ensured when the data is available to authorized parties when needed, i.e. is not lost or intercepted during transmission

The different component/data types shown here should only show the different interpretations of the elements of the security concept metamodel. The interpretation of the element itself does not have an influence on the chosen metamodel element, i.e. the element for both physical and virtual components will still be *Component*. The only difference can be found in data since transmitted data always belongs to a *Connection*. For both processed and stored data however the metamodel element *Data* will be used.

4.2.3 Interpretation of Interconnections between Elements

4.3 Model Transformation

4.4 Transformation Rule Set for Security Goals

4.5 Transformation Rule Set for Threats

4.6 Transformation Rule Set for Controls

References

[1] ISO/IEC 27001:2005 - information technology – security techniques – information security management systems – requirements. Technical report, 2005.

[2] Mark Branagan, Robert Dawson, and Dennis Longley. Security risk analysis for complex systems. pages 1–12, 2006.

[3] Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Co., New York, NY, USA, 1988.

[4] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl: A model transformation tool. *Science of Computer Programming*, 72(1):31 – 39, 2008.

[5] Prathaben Kanagasingham. Data loss prevention. *SANS Institute. Aug*, 2008.

[6] Michael Menzel, Christian Wolter, and Christoph Meinel. *Towards the Aggregation of Security Requirements in Cross-Organisational Service Compositions*, pages 297–308. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[7] Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pages 109–118, New York, NY, USA, 2004. ACM.

[8] OMG. OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1, June 2013.

[9] Klaus Pohl, Harald Hönninger, Reinhold Achatz, and Manfred Broy. *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer Publishing Company, Incorporated, 2012.

[10] Dorothy Elizabeth Robling Denning. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., 1982.

[11] E. Seidewitz. What models mean. *IEEE Software*, 20(5):26–32, Sept 2003.

[12] The Common Criteria Recognition Agreement Members. Common criteria for information technology security evaluation. http://www.commoncriteriaportal.org/, September 2012.

[13] Judith Thyssen, Daniel Ratiu, Wolfgang Schwitzer, Alexander Harhurin, Martin Feilkas, and Eike Thaden. A system for seamless abstraction layers for model-based development of embedded software. In *Software Engineering (Workshops)*, pages 137–148, 2010.

[14] J.R. Vacca. *Computer and Information Security Handbook*. Elsevier Science, 2012.