A Catalog of Security Requirements Patterns for the Domain of Cloud Computing Systems

Kristian Beckers
paluno
Duisburg, Germany
{firstname.lastname}@uni-due.de

Isabelle Côté, Ludger Goeke
ITESYS GmbH
Dortmund, Germany
{firstname.lastname}@itesys.de

ABSTRACT

Security and privacy concerns are essential in cloud computing scenarios, because cloud customers and end customers have to trust the cloud provider with their critical business data and even their IT infrastructure. In projects these are often addressed late in the software development life-cycle, because these are difficult to elicit in cloud scenarios, due to the large amount of stakeholders and technologies involved. We contribute a catalog of security and privacy requirement patterns that support software engineers in eliciting these requirements. As requirements patterns provide artifacts for re-using requirements. This paper shows how these requirements can be classified according to cloud security and privacy goals. Furthermore, we provide a structured method on how to elicit the right requirements for a given scenario. We mined these requirements patterns from existing security analysis of public organizations such as ENISA and the Cloud Security Alliance, from our practical experience in the cloud domain, and from our previous research in cloud security. We validate our requirements patterns in co-operation with industrial partners of the ClouDAT project.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*elicitation methods, methodologies*

General Terms

Security, Privacy, Documentation, Patterns

Keywords

Requirements patterns, requirements elicitation, security requirements, security standards, cloud computing, ISO 27001

1. INTRODUCTION

Software requirements that appear in documents (short: SRD) can be classified into categories, namely functional, quality [11], and non-functional requirements:

- **Functional requirements** describe the functionality of a system that satisfy the needs of its stakeholders. Based on these requirements we can derive the activities that describe how requirements fulfill the needs of its stakeholders.
- **Security requirements** are typically confidentiality, integrity or availability requirements. They refer to a particular piece of information, the *asset*, that should be protected, and it indicates the *counter-stakeholder* against whom the requirement is directed. A *stakeholder* is an individual, a group, or an organization that has an interest in the system under construction.
- **Other Non-Functional Requirements** are requirements that fulfill needs of stakeholders, which usually cannot be verified, for example, the usage of the system has to be intuitive.

In our recent research, we analyzed a number of SRD from small and medium enterprises (SRE) with regard to the consideration of security. We observed that a big percentage of SREs had problems eliciting and describing their security needs. The documents contained only very vague statements towards security, e.g., the system must not have any security issues. While functional requirements, in contrast, were given in more detail. This is the case because the SRE were able to describe what their required functionalities were. Moreover, the needed security requirements were often very similar. Hence, we propose re-usable textual requirements patterns. We previously analyzed the activities required to describe the context of cloud computing systems and to elicit security and privacy requirements. We used analysis patterns of clouds to support these activities [1]. We illustrated how to describe security textually from this information [2], which should be in compliance with the security standard ISO 27001 [13].

The contribution of this paper is a pattern catalog for security requirements with regard to cloud computing systems. We base our catalog on the work of Withall [19] from Microsoft, who provides guidelines and examples for formulating software requirements. Other researchers have proposed catalogs for textual requirements. Palomares et al. [17] present a catalog for functional software requirements for content management systems. Liu et al. [14] present requirements patterns for networked software systems. Our work differs from these approaches, because we focus on providing reusable requirements specifically for security in the context of the cloud computing domain.

The SREs also had problems regarding the category "other non-functional requirements". However, this category is not within the scope of this paper. Therefore, it will not be considered further.

This paper is structured as follows. We present the ClouDAT framework in Sect. 2 of which our pattern catalog is an essential part. We introduce our method for mining security requirements

SAC'14 March 18-22, 2014, Seoul, Korea.

patterns found in our catalog in Sect. 3 and show an example application in Sect. 4. We discuss our initial trials with SREs in Sect. 5 and conclude in Sect. 6.

2. THE CLOUDAT FRAMEWORK

The ClouDAT framework is a result of the ClouDAT project. The project, in turn, is a collaboration of the SREs ITESYS GmbH and LinogistiX as well as the universities University of Duisburg-Essen (UDE) in cooperation with the research institutes paluno the Ruhr institute for software technology and the Fraunhofer Institute of Software and Systems Engineering (ISST). This project is funded by the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems as well as the Ministry of Innovation, Science, Research and Technology of the German State of North Rhine-Westphalia and EFRE. The SREs involved in this projects make sure that the obtained results, frameworks, tools, and methods meet the needs displayed by most SREs.

The developed framework is going to be published as open-source. This allows interested parties to try out and use the framework free of charge.

The frameworks goal is to provide a means for a SRE to create a cloud-specific information security management system (ISMS) compliant to the ISO 27001 [13] standard. An ISMS is a process that ensures the security of an organization or parts thereof. Currently, the framework includes:

- A structural meta-model of a cloud system and a corresponding context-pattern and templates to elicit all relevant information of a cloud scenario [4]. The context-pattern is part of a larger pattern language that provides the means to also support different types of system like service-oriented architecture (SOA) (see [3] for more details).
- A simple method that describes how to conduct a security analysis and to create a cloud-specific ISMS [1].
- Tool-support for eliciting and analyzing the required information and also to create the required documentation for an ISO 27001 certification [2].
- A catalog for security requirements patterns.
- A mapping of security requirements patterns to mandatory controls in Annex A of the ISO 27001 standard.

3. A CATALOG FOR CLOUD SECURITY REQUIREMENTS PATTERNS

In this paper, we present a catalog of security requirements patterns for the domain of cloud computing. We build our patterns from state-of-the-art threat and risk catalogs, which we mapped to security requirements. In particular, we considered the work of the Cloud Security Alliance (CSA) [7], Gartner [12], the European Network and Information Security Agency (ENISA) [10], the German Federal Office for Information Security (BSI) [8, 6], Eurocloud [9], Bitkom [5], and Fraunhofer SIT [18], as well as the book from Mather et al. [15], and the works of Mell and Grance of the U.S. National Institute of Standards and Technology [16].

The current version of the catalog contains a total of 78 security requirements. This set of security requirements does not claim to be complete. Each application of the catalog can extend the set of requirements, if necessary.

We constructed the structured ClouDAT method for mining security requirements patterns. This ClouDAT method contains the following steps.

1. *Analysis*: We analyzed all the listed documents for risks, threats, security and privacy related texts. We marked the identified parts of the texts and stated to which cloud stakeholders or technical elements they refer to. We also mined the identified assets in the cloud scenarios. With asset being anything a stakeholder places value upon.
2. *Categorization*: We classify the identified texts into threats, risks, and security requirements. Threats refer to e.g. attackers that may exploit vulnerabilities in the cloud system. Risks focus on unwanted incidents that can cause a threat to occur. A security requirement states the confidentiality, integrity, and availability needs for the protection of an asset.
3. *Formulation*: We transform each threat and risk into security requirements. This is achieved by stating the needs for protection of assets being referred to. In this step, we check if security requirements appear multiple times. We formulate a new textual security requirement pattern for each requirement that appears at least two times.
4. *Alignment*: We check each of the selected requirement patterns if it refers to a cloud element or a stakeholder in the text. We replace these texts with placeholders of a certain type. Such a type refers to an element of our structural meta-model of a cloud system, e.g. an IaaS service.
5. *Mapping*: We map the security requirements to security controls of the ISO 27001 standard. We used structured reasoning and past experiences to supplement these mappings.
6. *Catalog Construction*: We categorize the resulting security requirements into protection needs. An example for such a need is authenticity. Thus, we obtain authenticity requirements patterns. We adopted the means of pattern representation proposed by Withall [19] to our approach.

4. APPLYING CLOUD SECURITY REQUIREMENTS PATTERNS

Withall [19] provides guidelines and examples for formulating software requirements based upon project experience. The author explains the need for documenting requirements including assumptions, glossary, document history and references. Withall's work aims at writing textual requirements, which also consider domain knowledge in the form of assumptions to these requirements. Our work differs from Withall's, because we do not provide patterns for requirements. Our work focuses more on the elicitation of domain knowledge.

The requirement patterns mentioned in [19] are ordered by the domains

- fundamental,
- information,
- data entity,
- user function,
- performance,
- flexibility,
- access control, and
- commercial.

For more details on these domains, please refer to [19].

We extended the above-mentioned domains by cloud computing-relevant domains we have identified throughout the project work. These domains are

- confidentiality,
- integrity,
- availability,
- authenticity,
- security management,
- non-repudiation,
- privacy,
- compliance, and
- transparency.

Structure of requirement patterns

We discuss the structure of security requirement patterns on the basis of the example security requirement pattern given in Table 1. This pattern specifies the requirements to the integrity of the communication regarding cloud computing.

Generally, a requirement pattern must have a name and an unique identifier. Our example (see Table 1) considers a security requirement pattern named *Integrity of cloud communication*. It is identified through the unique index *I6*.

In [19], the structure of requirement patterns is defined by the sections *Basic Details*, *Applicability*, *Discussion*, *Content*, *Template(s)*, *Example(s)*, *Extra requirements*, *Considerations for development* and *Considerations for testing*.

These sections are discussed in the following. However, we omit the sections *Extra requirements*, *Considerations for development* and *Considerations for testing*, as we do not require them for our work.

Basic Details

The basic details provide basic information about a requirement pattern.

One basic detail is the pattern manifestation. It represents meta-information of the requirement pattern such as version number, date of last change, or used specification language. If different variants or versions of a requirement pattern exist, more than one pattern manifestation are given. Because there exists only one version of our security requirement pattern example (see Table 1) it only has one pattern manifestation. It has the version number *1.2* and the last changes were made on *07.05.2013*. The specification language of our example is *English*.

Another item of the basic details describes the domain to which a requirement pattern belongs to. In our example the security requirement pattern belongs to the domain *Integrity*.

Furthermore, related requirement patterns are stated. These are requirement patterns that are applied in the same context as the considered requirement pattern. If necessary, the kinds of the appropriate relationships can be described. Our example security requirement pattern is related to several of our other security requirement patterns, namely to *Physical protection/ authentication of people (Aut13)*, *Destruction of cryptographic keys (SM10)*, *Securing integrity and confidentiality of cryptographic keys (SM11)*, and *Sufficient quality of cryptographic keys (SM12)*. The relations are depicted in Figure 1. Example 1 shows the full description of the security requirement pattern *SM12*. It states the requirement for the generation of cryptographic keys that are relevant in the context of the integrity of cloud communication.

The anticipated frequency represents the number of usages of the concerned requirement pattern in a typical requirement specification. We expect for our example requirement that it will be used twice in average.

A requirement pattern can be classified by a list of pattern classifications. The items of this list are represented by a name-value-tuple represented as *Name: Value* [19]. The different tuples are separated by semicolons (cf. [19]). Our security requirement pattern example can be assigned with *Functional* and *security*. Therefore, the tuples for our example look as follows: Functional: Yes, Security: Yes.

> Mechanisms for the generation of cryptographic keys shall ensure sufficient quality of the keys in «*all cloud elements*».

Example 1: The related security requirement pattern *SM12*

Applicability

Usually, the application of a requirement pattern can succeed only in on clear situation (cf. [19]). This situation is described in the applicability section. If necessary, this section can also contain descriptions of situations where the requirement pattern should not be applied (cf. [19]). Our security requirement pattern example (see Table 1) is used, when the integrity of cloud communications has to be preserved.

Discussion

The discussion section provides context information that is helpful for the specification of the requirement. It can support the creation of requirements by describing a creation process, list topics that shall be considered or caution against pitfalls (cf. [19]). The discussion in our example (see Table 1) mentions that it can be instantiated for the following two categories of cloud communication:

Category 1 Cloud communications between direct stakeholders and cloud services.

Category 2 Cloud communications between cloud services.

Content

This section lists items of information that a requirement of this type must convey or can optionally convey, respectively. An information item is a tuple of an item name and an informal text. Optional information items are marked by the keyword *optional* that is enclosed in parentheses. In our example (see Table 1), the content is described by one information tuple. It represents information regarding the participants in a cloud communication in the context of our security requirement pattern example.

Templates

Templates act as starting point for the specification. They provide text passages that are characteristic for a requirement of a certain type. These text passages can contain placeholders that can be filled with appropriate information regarding the relevant actual situation. Each template corresponds to an information item of the content. Here, a requirement pattern doesn't have to contain templates for all information items. In [19] the rule of thumb is mentioned that a requirement pattern shall only contain templates that are used in at least 20 percent of the specified requirement patterns. It is not mandatory that all templates are used for the specification of a requirement. The modifying or disregarding of the text passages during the instantiation of the requirement pattern is also allowed.

A template consists of a *summary* and a *definition*. The definition represents the specification of a part of the requirement. A definition is identified by the corresponding summary. It typically consists of a text regarding the pattern name and a text that allows

the distinction from the other templates of the requirement pattern. The summary gives a short hint about the content of the description.

The content and description can contain the above mentioned placeholders. A placeholder is marked by doubled-angled brackets and an italic notation.

Templates can contain optional parts. These are text passages that can or cannot be used regarding the situation the requirement is specified for. Optional parts allow for the creation of alternative requirements for typical situations. As mentioned above, text passages could be disregarded, albeit they are no optional parts. Optional parts are marked by squared brackets.

Descriptions can contain lists of items. In this case, each item of a list is labeled by a sequence number. The omission of list items is represented by an an ellipsis (...).

Our security pattern example specifies two templates (see Table 1). Here, each template relates to a category regarding the individual participants in cloud communications, whereby the integrity of the communications has to be preserved. The templates contain the placeholders «direct stakeholder» and «cloud service» as well as just «cloud service», respectively. During the application of the ClouDAT framework a cloud computing scenario is specified. This cloud computing scenario bases on the structural meta-model of a cloud mentioned in Section 2. The placeholders in our security requirement patterns can be substituted by the appropriate information from the specified cloud scenario. Regarding to the templates in our example, information in terms of the representation of direct stakeholders and cloud services in a corresponding cloud computing scenario can be inserted into the templates. For reasons of space the structural meta-model of a cloud and the specification of the cloud computing scenario cannot be described in detail in this paper. However, the interested reader may refer to [2] to get more information on the mentioned meta-model.

Table 1: Security requirement pattern *integrity of cloud communication*

Security Requirement Pattern: I6 Integrity of cloud communication		
Section	**Meaning**	
1. Basic details	• Version number: 1.2 • Date of last change: 07.05.2013 • Specification language: English • Domain: Integrity • Related patterns: Aut13, SM10, SM11, SM12 • Anticipated frequency: in average 2 • Pattern classifications: Functional: Yes; Security: Yes	
2. Applicability	Use the integrity of cloud communication security requirement pattern to specify for which communication connection the integrity must be preserved.	
3. Discussion	An *integrity of cloud communication security requirement* specifies the two participants for whose communication the integrity has to be preserved. These security requirements are subdivided in the following categories: • **Category 1: Direct Stakeholder to cloud service** to specify the need of integrity for the communication between direct stakeholders and cloud services. • **Category 2: Cloud services among each other** to specify the need of integrity for the communication between cloud services.	
4. Content	An *integrity of cloud communication security requirement* has to contain the two participants for whom the integrity of the communication has to be preserved. A cloud communication security requirement should contain: • **Cloud communication participants** The two participants in a cloud computing communication have to be – a direct stakeholder and a cloud service **or** – two cloud services.	
5. Template(s)	**Summary**	**Definition**
	Integrity of the communication between «direct stakeholder» and «cloud service»	The integrity of the communication between «direct stakeholder» and «cloud service» shall be preserved.
	Integrity of the communication between «cloud service» and «cloud service»	The integrity of the communication between «cloud service» and «cloud service» shall be preserved.
6. Example(s)	**Summary**	**Definition**
	Integrity of the communication between administrators and IaaS	The integrity of the communication between administrators and IaaS shall be preserved.
	Integrity of the communication between end customers and SaaS	The integrity of the communication between end customers and SaaS shall be preserved.
	Integrity of the communication between IaaS and PaaS	The integrity of the communication between IaaS and PaaS shall be preserved.

Examples

A requirement pattern should contain at least one requirement as an example of an instantiation of the requirement pattern. In our integrity of cloud communication security requirement pattern the first two examples are instantiations of the first template. Their content belongs to Category 1 meaning they specify requirements to the cloud communication between direct stakeholders and a cloud service (see Discussion). The first example considers the communication between direct stakeholders, namely administrators and the cloud service IaaS. In the second example, the communication between end customers of the cloud and the cloud service SaaS is considered. The third example is an instantiation of the second template. Its content belongs to Category 2. According to this, it specifies requirements for the communication between cloud services. In this case, the communication between cloud services IaaS and PaaS is referenced.

In addition to creating a catalog of security requirement patterns, we mapped our security requirement patterns to appropriate security measures in form of the controls given in Annex A of ISO 27001 [13]. This mapping was conducted in the following four steps:

1. Determining particular assets, security goals and expressive keywords regarding individual security requirement patterns from our catalog. By means of this additional information the set of possibly relevant security requirement patterns regarding an ISO 27001 control can be narrowed down.

2. Screening the ISO 27001 Annex A and assign the relevant ISO 27001 controls to the individual security requirement patterns. This step contains the following sub-steps:

   (a) Identifying a set of security requirement patterns for every ISO 27001 control with the help of the above men-

tioned additional information.

(b) Close consideration of the security requirement patterns determined in (a).

(c) Capturing of relevant mappings.

3. Iterating over our security requirement patterns and executing full text searches in the ISO 27001 Annex A. Here, the search items are the according additional information (assets, security goals, keywords) from the appropriate security requirement pattern.

4. Mutual review amongst the project participants of the created mapping and discussion of open issues.

In Figure 1, we illustrate the mapping of ISO 27001 controls to our example security requirement pattern given in Table 1. Furthermore, we provide the full text of the mapped ISO 27001 control *A.10.6.1* in Example 2.

> Networks shall be adequately managed and controlled, in order to be protected from threats, and to maintain security for the systems and applications using the network, including information in transit.

Example 2: ISO 27001 control *A.10.6.1 Network controls* [13]

5. DISCUSSION AND ANALYSIS

We analyzed our security requirements pattern catalog against scenarios developed in the ClouDAT project and discussed this catalog with security consultants, who have already used our catalog partially in industrial projects. Resulting in a consolidated set of security requirements patterns.

During the discussion, we found out that in ISO 27001 projects the selected controls have to be documented and reasoned in the so-called *Statement of Applicability (SOA)*. Especially, the effort for this task is significantly reduced by the presented catalog and tool support. The security consultants also mentioned that this structured procedure

- helps to elicit security requirements in more detail,
- supports the identification of threats to assets,
- helps to not forget relevant security concerns, and
- supports the identification and classification of functional as well as security requirements.

We analyzed how many security requirements were selected and which were excluded, as well. Based on our experience in the field of cloud security, we reconstructed 73 percent of the initial SRD documents. The remaining requirements were very specific to the documents. In addition, using our catalog, we could elicit an average of 17 percent of additional security requirements for the SRD documents.

In discussions with evaluators of the ISO 27001 standards, the following concerns towards our pattern catalog were raised:

- Security requirements pattern might hinder security reasoning, because security analysts will randomly select patterns.
- The reading of the entire pattern catalog is time consuming.
- The structure of the requirements patterns has to be learned beforehand, and
- Our method does not support the do-check-act phases of the ISO 27001 standard.

6. CONCLUSIONS

In this work, we presented a catalog of security requirements patterns for the cloud computing domain. This catalog is part of the ClouDAT framework, which is a holistic method including tool support for evaluating cloud computing offers with regard to security. Our catalog is currently composed of 78 security requirements in 10 different protection categories. We showed a structured method for mining the patterns and validated our catalog via discussions with the industrial partners of the ClouDAT project. These industrial partners applied or are applying our method to their scenarios.

Our pattern catalog offers the following main contributions:

- Starting from a context description of a cloud scenario, our pattern catalog can be used to elicit security requirements with little effort.
- The instantiation is tool supported and uses the context description as an input. Hence, the instantiation of the patterns refers to particular parts of a cloud.
- Using a structured pattern catalog decreases the risk of underspecifying security requirements for a given cloud scenarios. In addition, the catalog can also be used to reason why particular patterns are not selected. This documentation can be used for audits to ensure that correct decisions were taken.
- We provide a mapping of our security requirements patterns to ISO 27001 controls, which helps the user in not only defining his/her needs, but also to specify how protection mechanisms should look like.
- We provide tool support for all the steps of our method.

As future work, we want to conduct an extensive study with industrial patterns to ensure the applicability of our catalog to a wider variety of cloud scenarios. We are also defining mappings to further security standards such as the BSI 100.X series of standards. Furthermore, we intent to have a closer look at the concerns raised by the evaluators and address them appropriately in our framework.

Acknowledgments

This research was partially supported by the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS, ICT-2009.1.4 Trustworthy ICT, Grant No. 256980) and the Ministry of Innovation, Science, Research and Technology of the German State of North Rhine-Westphalia and EFRE (Grant No. 300266902 and Grant No. 300267002).

7. REFERENCES

[1] K. Beckers, I. Côté, S. Faßbender, M. Heisel, and S. Hofbauer. A pattern-based method for establishing a cloud-specific information security management system. *Requirements Engineering*, pages 1–53, 2013.

[2] K. Beckers, I. Côté, L. Goeke, S. Güler, and M. Heisel. Structured pattern-based security requirements elicitation for clouds. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES) - 7th International Workshop on Secure Software Engineering (SecSE 2013)*, pages 465–474. IEEE Computer Society, 2013.

[3] K. Beckers, S. Faßbender, and M. Heisel. A meta-model approach to the fundamentals for a pattern language for context elicitation. In *Proceedings of the 18th European*

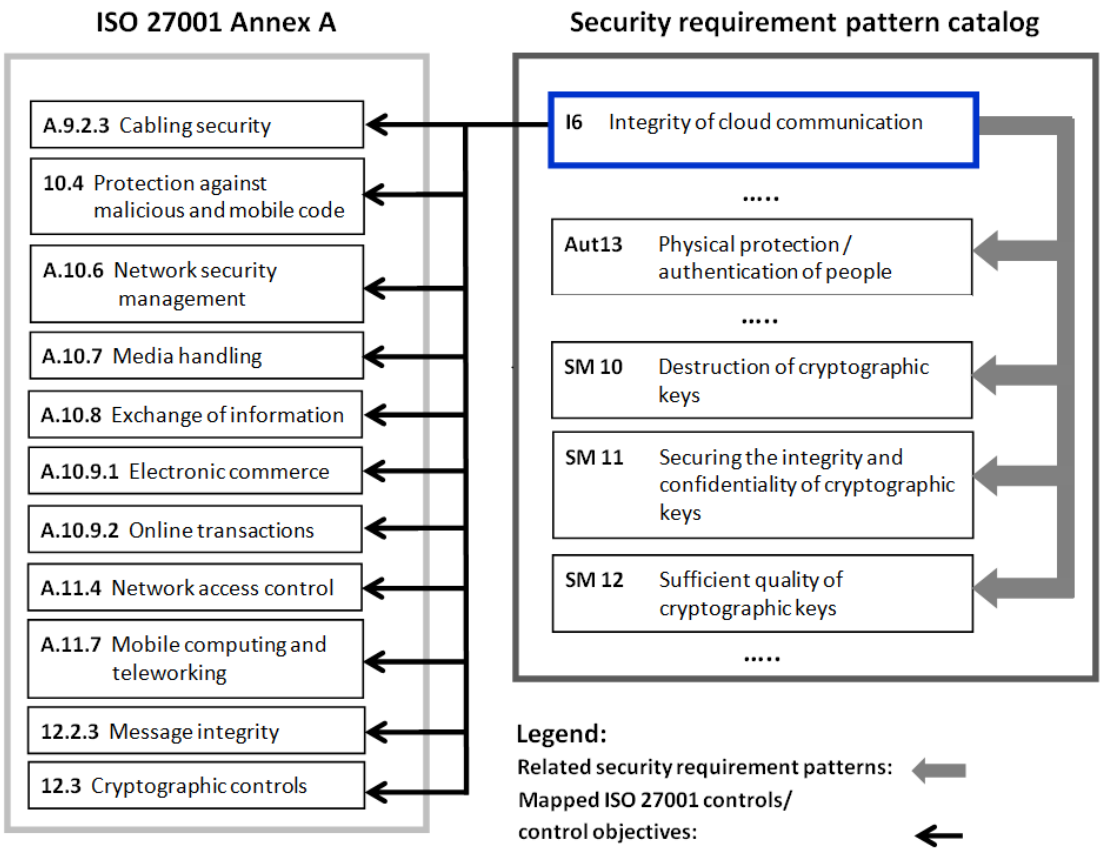

Figure 1: ISO 27001-mapping for *integrity of cloud communication* security requirement pattern

Conference on Pattern Languages of Programs (Europlop), pages –. ACM, 2013. Accepted for Publication.

[4] K. Beckers, J.-C. Küster, S. Faßbender, and H. Schmidt. Pattern-based support for context establishment and asset identification of the ISO 27000 in the field of cloud computing. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, pages 327–333. IEEE Computer Society, 2011.

[5] BITKOM. Cloud-computing - evolution in der technik, revolution im business, 2009.

[6] BSI. It-grundschutzkataloge, 2010. `http://www.bsi.bund.de`.

[7] Cloud Security Alliance (CSA). Top threats to cloud computing, March 2010.

[8] A. D. Essoh. Cloud computing und sicherheit – geht denn das?, 2010. `https://www.bsi.bund.de/cae/servlet/contentblob/808266/publicationFile/46724/07_essoh_bsi.pdf`.

[9] Eurocloud. Eurocloud prüfkatalog, 2010.

[10] European Network and Information Security Agency (ENISA). Cloud computing - benefits, risks and recommendations for information security, 2009.

[11] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt. A comparison of security requirements engineering methods. *Requirements Engineering*, 15(1):7–40, 2010.

[12] J. Heiser and M. Nicolett. Assessing the security risks of cloud computing, June 2008.

[13] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Information security management systems - Requirements, 2005.

[14] W. Liu, K.-Q. He, K. Zhang, and J. Wang. Combining domain-driven approach with requirement assets for networked software requirements elicitation. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, ICSC '08, pages 354–361, Washington, DC, USA, 2008. IEEE Computer Society.

[15] T. Mather, S. Kumaraswamy, and S. Latif. *Cloud Security and Privacy*. O'Reilly, 2009.

[16] P. Mell and T. Grance. *Effectively and Securely Using the Cloud-Computing Paradigm*. NIST, 2009. presentation at NIST.

[17] C. Palomares, C. Quer, X. Franch, S. Renault, and C. Guerlain. A catalogue of functional software requirement patterns for the domain of content management systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1260–1265. ACM, 2013.

[18] W. Streitberger and A. Ruppel. Cloud-computing sicherheit - schutzziele.taxonomie. marktübersicht, fraunhofer institute for secure information technology (sit). Technical report, Fraunhofer Institute for Secure Information Technology (SIT), 2009.

[19] S. Withall. *Software Requirement Patterns*. MICROSOFT PRESS, 2007.