User-friendly and Tailored Policy Administration Points

Manuel Rudolph
Fraunhofer IESE, Fraunhofer-Platz 1, Kaiserslautern, Germany

1 RESEARCH PROBLEM

Today, more and more sensitive personal and business-related data is collected and processed. Security solutions aim to protect such data from misuse, but often come along with usability issues (Gutmann and Grigg, 2005).

Security policies are an established instrument for specifying security demands. Policy Administration Points (PAPs) offer a policy vocabulary, which the policy creator can use to express the desired security demand as a security policy. Such security policies and the security mechanisms to which they refer can become very complex. Mistakes in the specification of security policies lead to misbehavior in the security mechanisms, which can cause, for instance, unintended data leakage and may convey a false feeling of security to the policy creator. Hence, specification mistakes must be minimized by providing appropriate PAPs that guide the policy creator in the best possible way through the policy specification process.

A well-known example for a PAP used by non-professionals is the Facebook menue for privacy and security settings. A study from Liu et al. revealed that users in many cases expected a different effect from their specified privacy settings than it was in reality (Liu et al., 2011), which indicates a wrongly specified security policy.

Intuitive and user-friendly usability of a PAP is, in my opinion, a basic requirement when letting users specify security policies. However, most current PAPs are neither easy to use nor easy to understand for less experienced policy creators. Examples for PAPs that do not sufficiently consider usability issues range from the Windows Local Group Policy Editor for managing the security settings of Windows clients to the PERMIS Policy Editor (Chadwick and Otenko, 2003; University of Kent, 2011) for specifying role based access control policies.

One problem is that most PAPs that have been found during a preliminary state of the practice research were designed generically to be used by different policy creator types, in different domains, or even both. This generality causes an overly high expressiveness of PAPs, which inevitably increases the complexity of the specification process as well as of the PAP implementation. Higher complexity is likely to increase the error-proneness of policy specification. In addition, in an organization with several types of policy creators that have different permissions for specifying security policies, an overly expressive PAP may allow policy creators to specify security policies for which they do not have clearance.

To avoid these problems, two essential steps toward more usable PAPs are suggested. First, PAP designers should aspire toward reducing the complexity and expressiveness of the security policy vocabulary and the policy specification process by better tailoring them to the respective application domain and to the expected user community. Second, PAPs should be made more user-friendly, especially for less experienced policy creators.

Thus, the following essential research question is addressed:

How can we efficiently develop Policy Administration Points that are providing a user-friendly security policy specification process with a security policy vocabulary tailored to the application domain and the policy creator?

2 TERMINOLOGY

In this paper, we use the following terminology:

- *Application Domain Model:* Set of entities, activities, and assets in a system or an organization to which elements of a security policy refer and their relation in the context of the application domain.
- *Application Domain Meta Model:* Model defining the syntax and semantics of an application domain model.
- *Security Policy Language:* Grammar to formulate security policies in a unified format used in security policy templates.
- *Security Policy Template:* Security policy pattern formulated in the security policy language that can be instantiated to a security policy.

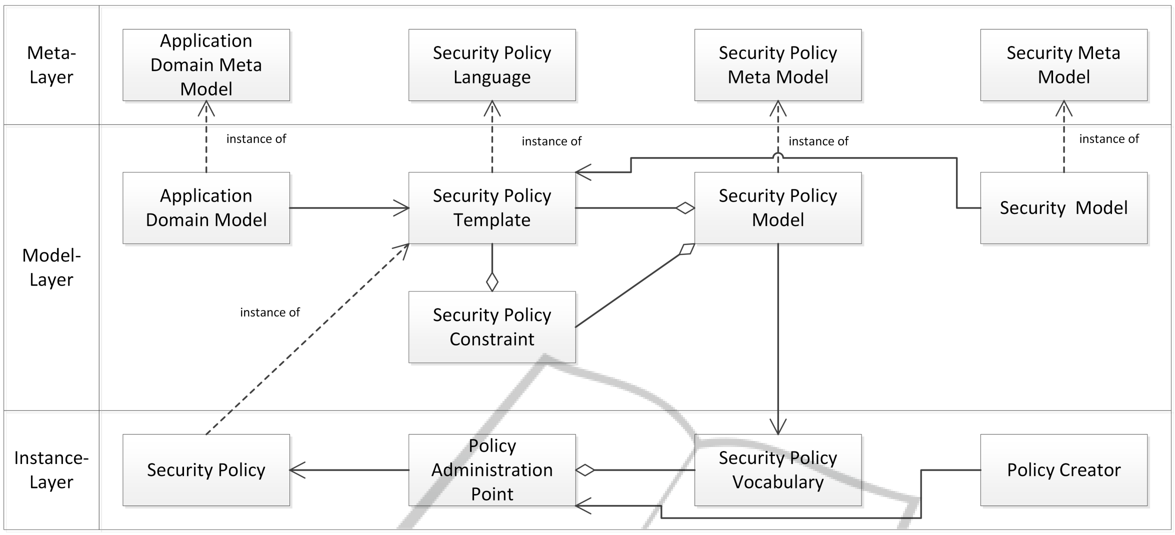

Figure 1: Terminology.

- *Security Policy Constraint:* Constraint that applies to security policy templates describing a restriction to a security policy template or a dependency between two or more security policy templates within a security policy model.
- *Security Policy Model:* Application domain-specific model that links security policy templates and security policy constraints as well as additional properties to the security policy templates.
- *Security Policy Meta Model:* Model defining the syntax and semantics of a security policy model.
- *Security Policy Vocabulary:* Subset of the security policy model specifically tailored to the needs of the policy creator and the expected policy usage context.
- *Policy Administration Point:* Human-Computer-Interface that consumes a security policy vocabulary to provide a set of security policy templates that can be instantiated as security policies for the specification of security demands.
- *Policy Creator:* Entity that has a security demand and specifies it as a security policy using a Policy Administration Point.
- *Security Policy:* Instance of a security policy template that describes the intended behavior of the policy enforcement components in a system in a given situation.
  - *Specification-level Policies:* Description of what must and must not happen throughout the execution of a system or within an organization.
  - *Implementation-level Policy:* Refinement of specification-level policies stating how the specification-level policies will actually be enforced.
- *Security Model:* Collection of application domain independent, recurring security mechanisms.
- *Security Meta Model:* Model defining the syntax and semantics of a security model.

The relation between the different elements is shown in Figure 1. Recall that the application domain model describes the entities that perform activities on assets in a given application domain. Some of the activities on assets are malicious and declared as threats. The threats from the application domain model are used in security policy templates to describe a situation which must be prevented or at least mitigated. In addition, security policy templates define reactions or countermeasures to specific threats that actually can prevent or mitigate these threats. Security policy templates are formulated in a natural language-based security policy language.

3 OUTLINE OF OBJECTIVES

The research question can be broken down into three main objectives. First, a security policy model for organizing security demands in the form of security policies or security policy templates must be elaborated that can generate tailored security policy vocabularies. Second, an elicitation method for gathering security demands from an application domain and filling the security policy model must be devised. Third, a usability-enhanced PAP that can consume a security policy vocabulary must be developed.

3.1 Security Policy Model

Currently, there is a lack of management solutions for security policies. In practice, security policies are written in natural language in a document and must then be deployed either in an organizational or in a technical manner. For each special case, a new set of security policies must be adapted from existing documents or developed from scratch. Defining or diversifying security policies at runtime is even harder, if supported at all. Furthermore, it is complicated and time-consuming to model and disclose dependencies between security policies.

To facilitate security management, an appropriate model for security policies helps. The security policy model must be able to store security policies and security policy templates in a unified form, to link security policies to each other in order to define dependencies, and to integrate restrictions that apply to security policies in a specific application domain.

If different individuals in an organization are empowered to specify security policies, new challenges arise. An individual may be allowed to specify security policies only from a limited security policy vocabulary. Accordingly, not all valid security policies might be allowable due to restricted policy entitlement of individual policy creators. A central organization-wide security policy model can be enriched with information about permissions, restrictions, and other properties with respect to security policies.

The reusability of security policy model instances should be investigated because instantiating a security policy model from scratch for every application domain would be a very effort-consuming process. To improve reuse of security policy knowledge, security policies and security policy templates that are used in different application domains should be generalized and stored as a security model that acts as a knowledge base.

The essential research question thus is: *How must a security policy model look like that is capable of managing and linking security policy templates and corresponding security policy constraints?*

3.2 Elicitation of Security Demands from an Application Domain

There are a lot of security guidelines that provide countless potential security policies as recommendations for organizations and their information systems. But it remains a challenge to select the appropriate policies for a concrete case. Furthermore, standard security policies do not reflect the specific characteristics of an organization or its information systems. Thus, security policies must be adapted to their application context.

Having only a standardized set of security policies that always applies leaves little scope for case-by-case security decisions. In some cases, users or administrators must be enabled to specify security policies that deviate from the standard policies. Therefore, they use PAPs. A generic PAP to specify any kind of security policy might be too powerful or too generic—and therefore too complicated—for the average policy creator.

Depending on the application domain in which the PAP will be used for specifying security policies, different requirements exist with respect to IT security. To reduce the complexity of a PAP and to tailor it to a given domain, a domain-specific security policy vocabulary is necessary. This requires a domain-specific instantiation of the security policy model. The instantiation of the model demands a domain-specific elicitation of security policies. Because the security policies that the policy creator want to specify may differ from case to case, case-specific security policy templates must be provided. Therefore, the general schema of relevant security policies needs to be analyzed. In doing so, varying elements of the policy can be identified and declared as variables. Variables can be either single parameters or policy elements, such as additional actions or conditions. For each policy template, security policy constraints, such as valid parameter values or dependencies on other policy templates, need to be elicited.

So far, no established method is known to me for eliciting the information necessary to instantiate a security policy model with security policy templates and security policy constraints. Devising such a method is proposed in this thesis.

The essential research question is: *How can we systematically elicit security policy templates including security policy constraints from an application domain?*

3.3 User-friendly Policy Administration Points

A PAP enables a policy creator to specify a security policy. Depending on the application domain, the PAP must support the specification of varying sets of security policies. It would be a huge effort to develop PAPs for each application domain from scratch. Therefore, generic customizable PAPs should be developed that can be configured to adapt to the given application domain. Basically, we strive for a PAP that is adapted to the application domain simply by

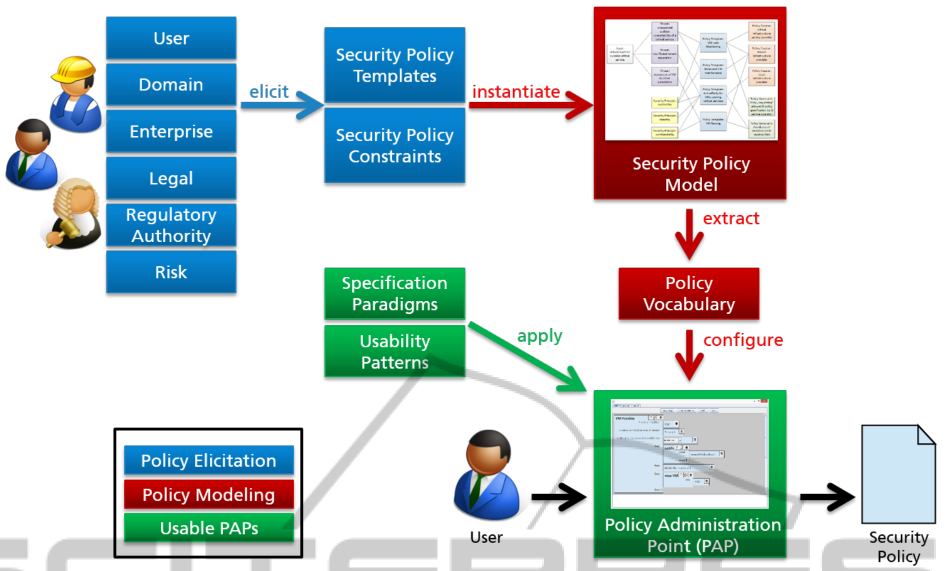

Figure 2: Schematic overview of the framework.

importing a suitable security policy vocabulary.

Different types of policy creators shall be enabled to specify security policies. These policy creators have different background knowledge about security, technology, or about the specific domain. Therefore, each policy creator type needs a different level of flexibility, usability, and guidance during the policy specification process, and the expressiveness of the PAP must be adapted accordingly.

One way to regulate expressiveness is to change the specification paradigm. Policy creators can have varying degrees of freedom depending on the specification paradigm offered by the PAP. A specification paradigm with a low expressiveness is, for instance, a list of predefined security policies the policy creator can select from. In relation, a wizard for combining and instantiating parameterized policy templates is a specification paradigm with a higher expressiveness.

The expressiveness of the PAP is also regulated by the imported security policy vocabulary. Note that the security policy vocabulary is only a subset of the policy model. Vocabulary extraction is regulated by filtering mechanisms. Filter criteria can be, for example, the permission model inside the security policy model or parameters of the security policy templates, such as the linked security principle. Furthermore, the PAPs can be enriched with guidance functionality to tailor them to the individual policy creator type.

The essential research question thus is: *How can usable Policy Administration Points for a specific type of policy creators be developed that self-adapt to a security policy vocabulary?*

4 EXPECTED OUTCOME

The expected outcome is a method that tailors PAPs and the policy vocabulary to the requirements of an application domain and the policy creators (see Figure 2). The method consists of the following steps:

- elicitation of security policy templates and constraints from various stakeholders within the application domain
- instantiation of the security policy model with the elicited information
- extraction of a tailored policy vocabulary
- configuration of usability-enhanced PAPs using the policy vocabulary

The proposed method extends the state of the art by supporting the systematic identification of proper sources for security policies within an application domain and their methodical assessment for eliciting the domain-specific policies. In addition, a security policy model will be suggested that improves policy management by allowing to express dependencies between different policies as well as constraints for individual policies. Based on this policy model and the systematic elicitation of domain-specific policies, PAPs for end users are better tailored to the specific security demands of the various policy creator types in an application domain and are made more usable. The following measurable benefits are expected when using the proposed method:

- Due to higher usability and user specific tailoring, policy specification in PAPs becomes less errorprone, which results in fewer wrongly specified security policies.

- The process of policy specification becomes more intuitive, easier to use, and less time-consuming even with limited background knowledge and minimal or no prior user training, which increases the acceptance of a policy enforcement mechanism such as data usage control both by the user and the enterprise.
- The development effort for providing tailored PAPs is reduced.

5 STAGE OF THE RESEARCH

The research is in its beginning, but first results have been achieved and partially published.

A preliminery state of the art research on the three main research questions and a state of the practice research on existing policy specification tools have been conducted.

A first version of the policy elicitation methodology has been proposed, tested with partners within a European project, and published (Rudolph et al., 2014). Further improvements and tests will follow.

Two prototypes of usable and adaptive Policy Administration Points are under development. Each PAP implements a different policy specification paradigm.

Building first adaptive PAP prototypes very early will allow to derive requirements from bottom-up for the policy vocabulary and the security policy model. The development of the elicitation methodology will in addition provide requirements from the top-down perspective.

A first draft version of the syntax and the semantics of the underlying policy vocabulary have been developed. Manually written policy vocabularies can be consumed by the PAP prototypes, which then self-adapt to the given vocabulary. One of those PAPs is currently being integrated in an industrial setting. In the future, the policy vocabulary will be generated from a security policy model, which is still under development.

6 METHODOLOGY

This section describes the plans for achieving the three main objectives presented before. The three objectives will not be tackled in strict order but partially in parallel in order to use insights gathered from one objective in another one.

6.1 Security Policy Model

For elaborating the security policy model, the following steps are planned:

1. The results of the preliminary state of the art research as well as the gathered insights from the draft versions of the policy elicitation methodology and the adaptive PAP prototypes will be analyzed to define requirements for the security policy model.
2. Before the security policy model can be developed, security policies must be better understood. The elements and characteristics of security policies must be analyzed. Therefore, exemplary security policies from different application domains are collected and compared. Generic elements of those policies are identified. Rules that define valid combinations of the policy elements are determined. From these results, a security policy language is derived. For testing the security policy language, security policies from different domains are formulated using the policy language.
3. An object-oriented security policy model is developed that is capable of storing security policy templates. One possible template for the policy "Each MP3 file may only be opened 3 times" could be "<filetype> may only be opened <n> times". Security policy templates are formulated in the security policy language and can therefore be split in their elemental parts. Elements of the policy templates may be combined into new policy templates. The model should also be capable of storing security policies as instances of security policy templates and linking them to the corresponding template. Additional parameters may be attached to the security policy templates. Examples are, for instance, a permission model that defines allowable policy creators, or flags that differentiate between technically and organizationally enforceable as well as between specification-level and implementation-level security policies. This differentiation becomes relevant for mapping a specification-level policy to corresponding refined implementation-level policies. In principle, PAPs can specify specification-level policies without any concrete technical enforcement mechanism as well as precise implementation-level policies. The former can be either refined to implementation-level policies or released as an organizational rule to be enforced in an arbitrary non-technical way. Dependencies between policy templates can be modeled as object relations, for example, to map specification-level policies to implementation-level policies. The categorization

of policy templates and their mapping to security properties is also supported. In addition, descriptions that explain the effect on the target, depending on whether the policy is or is not deployed, can be attached to policies. It must be researched which additional information is necessary in the security policy model.

4. The security policy templates can be instantiated by policy creators to express their security demands. Depending on the application domain or on the policy creator, there may exist constraints regarding security policies allowable for deployment. For example, not all security policy templates or not all parameter values within a security policy template may be allowed for the instantiation of a security policy template. Thus, a predefined selection of parameter values or parameter thresholds should be determined. Furthermore, dependencies between policies can be constrained. For instance, repelling or attracting policies may exist in the application domain. Examples for repelling policies could be contradictory policies such as ”Business documents must not be opened outside the enterprise premises“ and ”All documents can be opened at home after entering a password“.
5. It is planned to evaluate the suitability of the security policy language and the security policy model by instantiating security policy models with security policies from two different application domains. As, so far, no method for eliciting security policies from the application domain has been elaborated, the author assumes that a complete list of security policy templates and corresponding security policy constraints is available. The elicitation method proposed in Section 6.2 will, after being elaborated, be used to provide such a list. It will be tested in a case study whether all security policy templates and security policy constraints can be inserted into the model and all necessary relations and dependencies can be modeled.
6. A comparison of the two security policy models shall reveal opportunities to reuse security knowledge for another application domain. If this is possible, a generic security model acting as a knowledge base could be elaborated. The model will comprise common security characteristics of assets. For example, regularly used security principles such as confidentiality, integrity, or availability can be described in the model. Those fundamental security characteristics can probably be stored as generic security policy templates. Thus, such a model can reveal security policy templates that should be considered during the policy elicitation in an application domain to establish a consistent level of security.

6.2 Elicitation of Security Demands from an Application Domain

It is planned to devise the security demand elicitation method in the following way:

1. Before security policy templates and security policy constraints can be elicited in an application domain, the domain itself must be better understood. It must be clarified what kind of relevant elements exist in the application domain and what kind of relations exist among them. Elements could be entities that perform actions, actions that can be performed, and assets deserving protection on which actions can be performed. Together, an adverse action performed by an unauthorized entity on a valuable asset describes a threat. In addition, security-related properties of entities and assets should be considered. A domain model is proposed that describes the elements and their relations in the domain. It will be researched whether elements of the domain model can be mapped to or related to elements in the security policy model. Such mappings and relations can support the elicitation of security policies. The application domain model must be instantiated for each application domain in cooperation with domain experts before starting the security policy elicitation.
2. To be able to gather security policies, proper information sources must be identified. Potential sources can be different stakeholders in the application domain, such as system operators, security officers, requirements engineers, or end users. Entities from the application domain model can be stakeholders, too. Existing documentation regarding security guidelines, enterprise regulations, and legal obligations can be additional sources for relevant security policy templates. A list of the generally most promising information sources should be elaborated as a starting point for each policy elicitation. A method for identifying additional sources will be proposed.
3. Security policies are systematically retrieved from the identified stakeholders. Requirements engineering techniques are used for the elicitation of security policies from stakeholders and other information sources. To this end, the most appropriate requirements engineering techniques in the state of the art must be identified and probably adapted for eliciting security policies. The domain model instance and the security model will

help to focus on the relevant assets in the respective application domain. It must be researched what kind of security policy level (specification-level policies, implementation-level policies, or both) can and should be collected during the elicitation phase. In addition, it must be determined which part of the security policy templates can by elicited from which source. Stakeholders can demand specific reactions on threats, and technical experts must validate whether those demands are technically enforceable. Security policies are elicited in natural language and split into their basic elements, and their structure is specified in the security policy language. Parameterizable parts of the policy are identified and security policy templates are created. These templates and the corresponding security policy instantiations are stored in the security policy model.

4. Besides the actual security policies, we may need to determine additional security policy constraints. Recall that collected security policies are parameterized and used as templates; but not all feasible parameter values may be allowed in the given domain. Thus, a predefined selection of parameter values or parameter thresholds should be determined. Additionally, information about attracting and repelling policies should be determined. A list of relevant security policy constraint types is elaborated and a method for gathering those is developed. Security policy constraints are stored in the security policy model and linked to the corresponding security policy templates.

6.3 User-friendly Policy Administration Points

It is planned to develop PAP prototypes consuming tailored policy vocabularies in order to evaluate their usability and effectivity:

1. The security policy vocabulary that the policy creator can use in the PAP must be extracted from the security policy model. The security policy vocabulary is a subset of the security policy model and contains all information that the policy creator needs when specifying a security policy. It must be researched which information is necessary for the PAP to enable policy creators to specify their individual security policies. The necessary information depends on the specification paradigm of the PAP. Examples for specification paradigms are the selection of predefined security policies, the instantiation of security templates with a predefined set of parameters, or the combination of different policy elements to a security policy. The security policy vocabulary for a specific PAP must be extracted from the security policy model and imported into the PAP. Therefore, a data format for storing a security policy vocabulary must be developed. It must be capable of storing security policy templates and security policy constraints as well as their relations. Additional information about security policies may be attached. The security policy model is extended by an export filter that extracts only necessary information into the security policy vocabulary.
2. State-of-the-practice usability concepts are collected and rated regarding their applicability to PAPs and the policy specification process. In addition, existing policy specification tools are analyzed regarding their usability.
3. Different generic customizable PAP prototypes are developed. These PAPs are designed for different policy creator types and differ in their specification paradigms. Such a PAP can import a security policy vocabulary and self-adapt to it. A policy creator using the adapted PAP can specify all security policies that are valid according to the security policy templates and security policy constraints defined in the selected security policy vocabulary. Besides tailoring the PAP to the characteristics of different policy creator types, the collected usability concepts are considered during the development of the PAP to provide the policy creator with a more user-friendly security policy specification process.
4. Finally, the tailored PAPs and the extracted security policy vocabularies are evaluated. The evaluations of these two targets must be separated because an insufficient quality of the security policy vocabulary would most probably negatively affect the evaluation of the PAP using this vocabulary. The evaluation of the proposed method will be carried out in application domains different from the one used for developing the security policy model, the PAPs, and the elicitation method. This shall demonstrate that the applicability of the proposed method is not restricted to a specific set of application domains. The quality attributes to be evaluated are effectiveness, efficiency, understandability, and user satisfaction, which—if increased—collectively lead to a higher usability of the PAPs. Satisfaction can be measured by the subjective feedback of the test persons, efficiency by the time needed to specify security policies, effectiveness by the fraction of correctly specified security policies, and understandability by the fraction of correctly specified security policies in

relation to the test persons' estimations of correctly specified security policies. The effect of different complexities of security policies on the specification process must be considered during the evaluation. A first controlled experiment will evaluate the security policy vocabulary. PAPs using the already evaluated security policy vocabulary will be evaluated in a second controlled experiment. In summary, three parameters influence the evaluation and must be taken into account: the complexity of the security policies, the application domain, and the PAPs chosen. Thus, several exercises to describe security demands that can be specified as security policies using the proposed security policy vocabulary will be assigned to the probands in the controlled experiments. The different exercises lead to security policies of different complexity. In the first controlled experiment, security policies must be formulated in natural language on paper, using a printed and human-readable version of the security policy vocabulary (i.e., policy templates and policy constraints). In the second controlled experiment, two different PAPs are instantiated with the already evaluated security policy vocabulary and are used by the probands to specify the same security policies as in the preceding exercises. To prevent a learning effect, a proband from the first experiment may not take part in second one.

7 STATE OF THE ART

The tailoring of PAPs involves the three main sub problems policy elicitation, policy modeling and policy specification. Below, the related work covering the respective problem domains is briefly surveyed.

The elicitation of security policy templates is related to the elicitation of security requirements and risks. To this end, the whole spectrum of standard techniques for requirements engineering (Tenerowicz, 2008), such as structured interviews of the involved stakeholders or creativity workshops assessing security issues, and risk assessment methodologies (Smith et al., 2013) can be used in principle as well as (mis)use case modeling (Alexander, 2003; McGraw, 2006). Mead et al. propose Security Quality Requirements Engineering (SQUARE) as a systematic process for eliciting security requirements (Mead et al., 2005; U.S. Computer Emergency Response Team, 2007), but without describing specific elicitation techniques. However, most of the proposed elicitation methods are not tailored to and have not been evaluated in the specific context of security requirements or policy elicitation, respectively. The elicitation step proposed in this paper will use state-of-the-art techniques for the elicitation of security demands from various sources. It must be evaluated which techniques lead to the best results depending on the type of information source.

Valuable sources for general security demands can be security checklists, control question catalogs, and security ontologies that have been proposed in the literature (U.S. National Institute of Standards and Technology, 2014; U.S. National Institute of Standards and Technology, 2008; German Bundesamt für Sicherheit in der Informationstechnik, 2005; Common Criteria Maintenance Board, 2012; U.S. Department of Defense, 1985). The elicitation step will consider those sources.

Several policy enforcement frameworks have been described in the literature, some of which include security policy models or PAPs, respectively.

SPARCLE (Vaniea et al., 2008; Karat et al., 2009) is a security and privacy policy framework for enforcing access control policies. It transforms policies specified in constrained natural language into a formal language and later in a machine-readable representation. It analyzes the formal policies, identifies conflicts and dominance relations, determines policy coverage, and provides suggestions for conflict resolution. In (Vaniea et al., 2008), Vaniea et al. report on experimental evaluations of improving the usability of policy specifications. Compared to SPARCLE, my framework is not limited to access control policies. It uses, for example, adjustable security policy templates in natural language as the policy specification paradigm instead of constrained natural language as SPARCLE does. Furthermore, the usability patterns applied to SPARCLE's PAP are different from the ones planned to be applied to the proposed framework's PAPs, which are described in (Vollat, 2012).

KAoS (Institute for Human & Machine Cognition, 2013) is a set of platform-independent services that enable users to specify and enforce security, predictability, and controllability policies. KAoS defines a policy as an enforceable, well-specified constraint on the performance of machine-executable actions by a subject in a given situation. KAoS policies, expressed in Web Ontology Language (OWL2) (World Wide Web Consortium, 2012), can be specified as authorization policies and obligation policies, but other types of policies can be constructed from these two primitive types. KAoS provides a PAP for experts that is neither tailored to a specific application domain nor provides a user-friendly specification process for non-experts, as it is intended in my framework.

Another approach for the elicitation and representation of security constraints is Secure Tropos (Mouratidis and Giorgini, 2007). The Tropos methodology aims to model security concerns throughout the whole development process. The Secure Tropos paradigm is based on agent-oriented software engineering and centers around the concepts of actors, their goals, obligations, capabilities, security constraints, and dependencies. The notation used is based on UML, and constraints can be formally expressed (and verified) with the Object Constraint Language (OCL) (Object Management Group, 2014). The stakeholder and security requirements identification used in Secure Tropos is another option that could be used in my approach, but its applicability has not yet been tested.

In addition to the already mentioned policy frameworks, there are various other policy languages for the specification of machine-readable policies, for example the Ponder language (Damianou et al., 2001), SSPL (Al-Morsy and Faheem, 2009) or Rei (Kagal et al., 2003). For a brief overview of available policy languages, see (De Coi and Olmedilla, 2008; World Wide Web Consortium, 2012).

ACKNOWLEDGEMENTS

The research presented in this paper is supported by the German Ministry of Education and Research projects Software Campus (grant number 01IS12053) and Software-Cluster SINNODIUM (grant number 01IC12S01F) as well as by the European Commission in the context of the Research Framework Program Seven (FP7) project SECCRIT (Grant Agreement No. 312758). The sole responsibility for the content of this document lies with the author.

REFERENCES

Al-Morsy, M. and Faheem, H. (2009). A new standard security policy language. *Potentials, IEEE*, 28(2):19–26.

Alexander, I. (2003). Misuse cases: use cases with hostile intent. *Software, IEEE*, 20(1):58–66.

Chadwick, D. W. and Otenko, A. (2003). The permis x. 509 role based privilege management infrastructure. *Future Generation Computer Systems*, 19(2):277–289.

Common Criteria Maintenance Board (2012). Common Criteria for Information Technology Security Evaluation, CCv3.1 Revision 4 (CCMB-2012-09-001, -002, -003). http://www.commoncriteriaportal.org/cc/.

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The ponder policy specification language. In Sloman, M., Lupu, E., and Lobo, J., editors, *Policies for Distributed Systems and Networks*, volume 1995 of *Lecture Notes in Computer Science*, pages 18–38. Springer Berlin Heidelberg.

De Coi, J. L. and Olmedilla, D. (2008). A review of trust management, security and privacy policy languages. In *SECRYPT*, pages 483–490. Citeseer.

German Bundesamt für Sicherheit in der Informationstechnik (2005). BSI: IT-Grundschutz Catalogues. https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz.html (a more recent version from 2013 in German is available at https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html).

Gutmann, P. and Grigg, I. (2005). Security usability. *Security Privacy, IEEE*, 3(4):56–58.

Institute for Human & Machine Cognition (2013). KAoS Policy Services Framework: User Guide. http://ontology.ihmc.us/KAoS/KAoSUsersGuide.pdf.

Kagal, L., Finin, T., and Joshi, A. (2003). A policy language for a pervasive computing environment. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 63–74.

Karat, J., Karat, C.-M., Bertino, E., Li, N., Ni, Q., Brodie, C., Lobo, J., Calo, S. B., Cranor, L. F., Kumaraguru, P., and Reeder, R. W. (2009). Policy framework for security and privacy management. *IBM J. Res. Dev.*, 53(2):242–255.

Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. (2011). Analyzing facebook privacy settings: User expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70.

McGraw, G. (2006). *Software security: building security in*, volume 1. Addison-Wesley Professional.

Mead, N. R., Hough, E., and Jr., T. R. S. (2005). Security Quality Requirements Engineering (SQUARE) Methodology. Technical Report CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Mouratidis, H. and Giorgini, P. (2007). Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(02):285–309.

Object Management Group (2014). Object Constraint Language (OCL). http://www.omg.org/spec/OCL/.

Rudolph, M., Schwarz, R., and Jung, C. (2014). Security policy specification templates for critical infrastructure services in the cloud. (in press).

Smith, P., Busby, J., Langer, L., Schöller, M., and Shirazi, N. (2013). SECCRIT Deliverable D3.1 Methodology for Risk Assessment and Management. https://seccrit.eu/publications/publicreports.

Tenerowicz, C. L. (2008). Elicitation Techniques. https://confluence.cornell.edu/display/BAF/Elicitation+Techniques.

University of Kent (2011). Permis. http://sec.cs.kent.ac.uk/permis/index.shtml.

U.S. Computer Emergency Response Team (2007). SQUARE - Requirements Engineering for Improved System Security.

U.S. Department of Defense (1985). DoD 5200.28-STD: Trusted Computer System Evaluation Criteria. http://csrc.nist.gov/publications/history/dod85.pdf.

U.S. National Institute of Standards and Technology (2008). NIST Special Publication 800-55, Revision 1: Performance Measurement Guide for Information Security. http://csrc.nist.gov/publications/PubsSPs.html.

U.S. National Institute of Standards and Technology (2014). NIST Special Publication 800-53, Revision 4: Security and Privacy Controls for Federal Information Systems and Organizations. http://csrc.nist.gov/publications/PubsSPs.html.

Vaniea, K., Karat, C.-M., Gross, J. B., Karat, J., and Brodie, C. (2008). Evaluating assistance of natural language policy authoring. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS '08, pages 65–73, New York, NY, USA. ACM.

Vollat, C. (2012). Graphical user interface development for usable policy administration points (paps). Master's thesis, University of Kaiserslautern.

World Wide Web Consortium (2012). Web Ontology Language (OWL). http://www.w3.org/2001/sw/wiki/OWL.