Usable Security Policy Specification

Manuel Rudolph, Denis Feth

Security Engineering Department, Fraunhofer Institute for Experimental Software Engineering IESE

Abstract

Security policies determine which security requirements have to be met in a domain and how they are implemented organizationally and/or technically. However, their specification at run-time poses a challenge for policy authors (e.g., IT administrators or end users), especially if they are inexperienced in this task. Thus, specification interfaces have to guide the policy author during the specification process. However, matching appropriate specification processes to the policy authors' individual needs is challenging due to a high variability in the authors' skill levels and security perceptions. In this paper, we identify existing specification approaches, derive generic specification paradigms and show the feasibility of one of them in an industrial case study.

1 Introduction

The specification of security policies is typically performed by different stakeholders such as security experts, IT administrators and even end users. Security policies can define run-time requirements, for example, for access control mechanisms ("Only Bob may access this file") or usage control mechanisms ("Bob is allowed to access this file once, but must not redistribute it"). However, different policy authors—experts in the specification of machine-enforceable policies or non-expert users trying to specify policies for the first time—face *different* challenges during policy specification. Challenges can be, for example, unclear terminology, misunderstanding of policy effects or ambiguities in the specification process. Thus, policy authors need *different* levels of introduction, guidance, interaction and expressiveness. Security policy editors, so-called Policy Administration Points (PAP), should optimally support policy authors during the specification process. However, current PAPs do not adapt to the requirements of different stakeholders (Bauer 2009). Even well-known security policy specification interfaces, such as Facebook privacy settings, are not completely usable by non-experts (Liu 2011).

To tackle this problem, we built a framework for the model-based generation of PAPs, which in turn suit the individual needs of different policy author types and situations. Our model-based approach allows fast adaptation of PAPs to different application domains. The general

Veröffentlicht durch die Gesellschaft für Informatik e.V. 2016 in
B. Weyers, A. Dittmar (Hrsg.):
Mensch und Computer 2016 – Workshopbeiträge, 4. - 7. September 2016, Aachen.

http://dx.doi.org/10.18420/muc2016-ws03-0004

idea of tailoring policy specification to end users and to the application domain has been published in (Rudolph 2015). The PAP framework comprises four main aspects:

- **Platform:** A PAP should be available for different platforms—e.g., for Android, Windows or as a web application. Multi-platform support allows the user to select the most appropriate specification device in her current situation.
- **Policy Vocabulary:** A PAP should be tailored to a specific domain, and it should use the corresponding vocabulary of the domain. In our framework, the policy vocabulary is a generic model containing security policy templates. These templates are based on the security demands of the domain's stakeholders, and they can be instantiated to concrete security policies. The vocabulary model reduces specification complexity and allows the specification of security policies using domain specific terminology.
- **Target Language:** Security policies can target different security solutions that support different policy languages. Policies specified by authors are typically not machine-enforceable. Therefore, they need to be transformed into enforceable equivalents. Thus, the PAP framework supports the transformation of policies into different policy languages.
- **Specification Paradigm:** Different user groups have different needs with respect to the specification interface and process. The interaction concepts of the policy specification process and the user interface are defined by so-called specification paradigms (e.g., wizard, template). These paradigms are intended to provide the appropriate degree of freedom for the author by considering his skill level and needs.

In this paper, we focus on the definition and evaluation of specification. We derive and generalize some specification paradigms from policy specification interfaces found in the literature and in practice. In addition, we report on an industrial case study where we examined one specific specification paradigm.

2 Related Work

Various PAPs aiming to provide user-friendly policy specification using different specification paradigms have been suggested in research. One common approach is the translation of (semi-structured) natural language to machine-enforceable policies. This approach is used in tools like SPARCLE (Reeder 2007), KAoS (Uszok 2004) and the "natural language to PERMIS[1] policy" parser (Inglesant 2008). PERMIS[1] also offers a "Policy Editor" and a "Policy Wizard". The "Policy Editor" provides a multi-tabbed specification interface with a variety of functions. The "Policy Wizard" guides the user through the specification process, asking supportive questions. Both approaches use a generic, domain-independent terminology. Thus, the policy vocabulary might not be understandable to non-experts.

[1] http://sec.cs.kent.ac.uk/permis/

Besides generic PAPs, many domain-specific tools exist. For instance, the Local Group Policy Editor[2] of Windows systems mainly targets system administrators and offers a variety of settings (e.g., firewall settings, password policies, startup/shutdown scripts) for Windows environments. Facebook[3] allows its users to specify their privacy settings in a very fine-grained manner. Even if studies revealed that users partially expected different behavior from the specified security policies (Liu 2011), they are at least empowered to specify them at all.

The PAPs we analyzed provide different levels of guidance for the policy author and request different levels of expressiveness from him, as well. However, it remains a challenge to choose the most appropriate specification paradigm for a concrete policy author.

Apart from practical implementation work on existing PAPs, there is also research into user-friendly security software. Whitten (Whitten 2004) shows how the usability patterns "safe staging" and "metaphor tailoring" improve the usability of security software. Safe staging proposes a step-by-step activation of security functionality in order to avoid overwhelming the user. Unexperienced users start with limited functionality. Additional features are offered when specific conditions (e.g., time, skill test or explicit activation by the user) are fulfilled. Metaphor tailoring proposes real life analogies for security concepts to fit the mental model of the users.

Independently of our security setting, general usability concepts and improvement have been extensively studied. For example, usability pattern libraries provide hundreds of different patterns (e.g., Tidwell 2011). Vollat (Vollat 2012) summarizes the state of the art in the area of usability patterns, and patterns are analyzed and rated with respect to their applicability to improving the usability of PAPs and the policy specification process in general.

3 Policy Specification Paradigms

We want to provide policy authors with a policy specification interface tailored to the personal skill level, to the policy author's preferences and to the application domain. This demands several interaction concepts as well as different levels of freedom for the security policy specification. These interaction concepts are called *security policy specification paradigms* and have to incorporate the following aspects:

- In order to reduce the complexity and to focus the specification according to the individual skill level, different *degrees of freedom* can be offered to the author.
- The *visual representation* of the security policies can differ. Policies can be presented textually or graphically in different ways. Examples are textual templates and so-called building blocks (predefined policy blocks that can be combined like lego bricks). Policy

[2] http://windows.microsoft.com/en-us/windows7/group-policy-management-for-it-pros

[3] http://www.facebook.com

templates and building blocks may contain variable parts for customizations, for example, names, amounts, email addresses or object identifiers.

- Different *interaction concepts* between the user and the system can be provided. Examples include assisted forms or wizards guiding through the specification process.

Figure 1: Different Security Policy Specification Paradigms

A large spectrum of interaction concepts is used in practice or has been suggested by research. We categorized them into six generic *specification paradigms*. These paradigms are shown in Figure 1, ordered according to the degrees of freedom they provide and the mandatory knowledge of the policy author. We assume that higher degrees of freedom require more expertise in order to produce correct security policies.

Below, we briefly describe the highlighted variants from Figure 1. We omit the first and last paradigm, as they do not provide any variation point for the PAP design. In the first paradigm, we have to provide full freedom and cannot support the author by restricting expressiveness of the policies; in the last paradigm, the PAP is just used for informing the user.

Composition of Predefined Policy Blocks: With this paradigm, policy authors can assemble predefined semantic building blocks to a concrete security policy according to a predefined policy "grammar". This paradigm can be beneficial if many small policy building blocks exist, which may introduce very complex policy templates in a domain.

Policy Template Instantiation: This paradigm uses policy templates that can be instantiated by the author. The customization parts can contain free text variables, predefined values or a selection of optional template parts. This paradigm can be beneficial if there are many similar policies that need to be adapted for each specific use case.

Policy Specification Wizard: Similar to the Policy Template Instantiation, the user follows a fixed specification process. In each specification step, the user is informed about meaning and consequences of specific security decisions. The wizard paradigm allows the concatenation of several security policy specifications in a predefined order. This paradigm is beneficial, as it does not overwhelm the author with too much information in a single step.

Selection from List of Predefined Policies: A predefined list of immutable security policies is presented to the policy author (e.g., in a natural or symbolic language). She can select policies according to his security needs. This paradigm is beneficial if there is only a very limited set of security policies or if the author is not supposed to have much freedom in specification.

High flexibility is needed to adjust PAPs for different users and situations. For example, in a typical corporate setting, IT administrators (security experts) could use a "Composition of Predefined Policy Blocks" PAP to set the basic security policies, whereas project managers (domain experts) would rather use a "Policy Template Instantiation" PAP to adapt security policies to the specific needs of a project. Finally, project members with little security and domain experience should be constrained to a PAP with the "Predefined Set of Security Policies" paradigm just for information purposes.

4 Initial Case Study

In general, usability in our approach can only be evaluated using concrete, generated PAP instances. Thus, as an initial step, together with a large German IT company (called "the company" in the following) we evaluated the usability of a generated "Policy Template Instantiation" PAP for Android. We wanted to learn whether such a PAP could be beneficial for the company and how different user groups would accept the chosen paradigm.

4.1 Setup

Our evaluation was split into two phases: exploration and discussion. In the exploration phase, participants were asked to test the Android PAP and to fill out a questionnaire in parallel. The questionnaire contained five questions (listed below) and an AttrakDiff (Hassenzahl 2003) word pair sheet. The task of the participants during the exploration was to answer the questionnaire with the following mindset: "Imagine that you need to specify security policies for a new project as a project leader in the company":

- The specification of security policies is a challenge in the company (1—low to 5—high)
- Name the three most positive and the three most negative aspects of the policy editor.
- Is there any possible application for such a policy editor? If yes, which?
- What would be the benefits of introducing such a policy editor?
- Which additional features does the policy editor need to provide in order to be acceptable?

In the second phase, a discussion round with all participants was conducted, in which feedback from the participants was collected. First, the participants talked about their experiences during the exploration phase. After that, we explicitly asked for positive and negative experiences, potential extension points (e.g., other platforms), and scenarios, where such a PAP would be beneficial. As our PAP framework also supports transformations to machine-enforceable policies, we also asked for (security) systems that could be configured using such policies. The

questions were quite similar to the questions already asked in the questionnaire. However, we wanted to let all participants hear the answers from the other participants to allow comments and extensions.

4.2 Execution

Before the evaluation, we elicited 13 security policy templates for the company in an expert workshop (the elicitation process and its results will be published separately). We built an Android PAP (cf. Figure 2) that uses the *Policy Template Instantiation* paradigm. On the left side of the policy instantiation screen, the user can choose between the 13 policy templates. On the right side, the instantiation of the template is performed. In Figure 2, a password policy is instantiated. The correct transformation of the instantiated natural language security policies into machine-enforceable equivalents was not part of this case study.

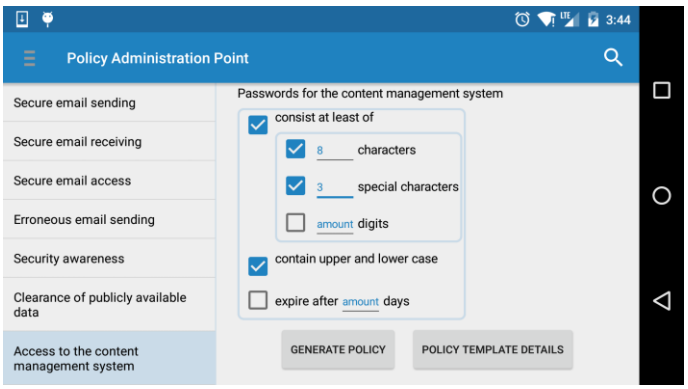

Figure 2: Android PAP app

In a second workshop, we conducted the PAP evaluation. Three participants joined. They represented the security, domain and end-user perspectives, which we consider the main stakeholder groups for security policy specification. The evaluation started with a short explanation of the evaluation and its target. We explained the functionality of the PAP in a slideshow with screenshots and presented the questionnaire. Next, the participants tested the PAP and answered the questionnaire for 25 minutes. Finally, we had a 25 minutes discussion. We neither tracked the concrete user interactions, nor did we store the specified security policies.

4.3 Results and Discussion

4.3.1 Questionnaire and Results of the Discussion

Specifying security policies at the company is considered a rather challenging task. The average rating was 3.7 out of 5 points. Thus, better guidance (e.g., by using a PAP tailored to the end user) could be beneficial for the company.

Regarding the tool itself, easy usage, clarity and a structured, unified specification process were named as benefits of the Android PAP. Regarding the selected paradigm, the restricted variety of the templates, the unified and domain-specific diction of the policies, and the structuring of the policies were positively mentioned.

Besides the positive feedback, some drawbacks of the Android PAP were reported. The set of 13 templates was perceived as confusing, although the Android PAP provides search and filter mechanisms. The templates and policies themselves were linguistically speaking not yet "human". On the one hand, the templates could be rephrased. On the other hand, a policy wizard with more guidance (e.g., detailed explanations of individual customization parts in the policy templates) could better support the policy authors in understanding the security policies.

The participants stated that, in general, a policy editor such as the Android PAP could be used to specify “data privacy statements”, “declaration of consent” or “modification of existing rules on changes in laws”. Using such an Android PAP, the company could benefit from the standardized and centralized procedure of specifying security policies. Currently, this is done in an unstructured way using checklists, and the security policies are documented in a text file. In addition, it could be beneficial that each data owner is empowered to specify security policies that reflect the personal protection needs on her own. This advantage is especially seen for non-expert users.

Desirable extension points and improvements for the PAP from the viewpoint of participants are adapters to existing systems. That is, policies should be automatically transformed into machine-enforceable policies that can be interpreted by, for example, the Windows Group Policy Editor. This feature was explicitly excluded in our case study as our experiment focused on the usability of the user interface and the specification process. In addition, the capability to add existing forms or other documents to the templates would increase the acceptance of the PAP. According to the participants, a clear process for the specification and maintenance of security policy templates at the company would be necessary in practice. Finally, the preferred a better guidance through the specification process. Thus, a policy wizard might be the better paradigm for employees of the company.

4.3.2 AttrakDiff Results

According to the AttrakDiff result, the product's user interface was rated as “fairly practice-oriented” regarding the pragmatic dimension; the PAP was rated between neutral and task-oriented. This means that on average, users can achieve their tasks with the tool, but there is room for improvement. From the hedonic point of view, users seemed to be stimulated by the tool, but only on an average level. Thus, there is room for improving the hedonic quality as well. With respect to the hedonic quality “identity”, the product’s mean value is located slightly above average level. Thus, our PAP meets ordinary standards, but a higher value would bind the user more strongly to the PAP. Regarding the hedonic quality “stimulation”, the mean value is located slightly above the average level, as well. Thus, our PAP also meets ordinary standards, and improvements would motivate, enthrall and stimulate users more strongly. The attractiveness was rated moderate. The AttrakDiff test revealed that the PAP is accepted as a user-friendly and attractive tool, but there are still improvement possibilities regarding usability and attractiveness.

5 Conclusion and Future Work

We identified multiple policy specification interfaces in research and practice and derived several specification paradigms from existing work. These paradigms will be part of a framework that can generate PAPs tailored to the application domain and to the author's needs. We tested one of these specification paradigms, the *Policy Template Instantiation* paradigm, in an industrial case study. This study revealed that such a PAP would be beneficial for the company, but still has improvement potential. In the future, we want to evaluate the feasibility of different paradigms in a larger scaled controlled experiment and elaborate a better matching of specification paradigms to policy author types depending on their skills and preferences. In addition, we need to tackle the challenge of choosing the most appropriate specification paradigm for a concrete policy author.

Acknowledgments

The research presented in this paper is supported by the German Ministry of Education and Research projects IUNO (grant number 16KIS0328) and Software Campus (grant number 01IS12053). The sole responsibility for the content of this document lies with the authors.

References

Bauer, L., Cranor, L. F., Reeder, R. W., Reiter, M. K. & Vaniea, K. (2009). *Real life challenges in access-control management*. In Olsen, D. R., Arthur, R. B., Hinckley, K., Morris, M. R., Hudson, S. & Greenberg, S. (Publ.): *the SIGCHI Conference*. p. 899.

Hassenzahl, M., Burmester, M. & Koller, F. (2003). *AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität*. In Szwillus, G. & Ziegler, J. (Publ.): Wiesbaden: Vieweg+Teubner Verlag, pp. 187–196.

Inglesant, P., Sasse, M. A., Chadwick, D. & Shi, L. L. (2008). *Expressions of expertness: the virtuous circle of natural language for access control policy specification.* Proceedings of the 4th symposium on usable privacy and security. New York, NY, USA: ACM, pp. 77–88.

Liu, Y., Gummadi, K. P., Krishnamurthy, B. & Mislove, A. (2011). *Analyzing Facebook privacy settings: User expectations vs. reality*. Proceedings of the 2011 ACM SIGCOMM conference. pp. 61–70.

Reeder, R., Karat, C.-M., Karat, J. & Brodie, C. (2007). *Usability Challenges in Security and Privacy Policy-Authoring Interfaces*. In Baranauskas, C., Palanque, P., Abascal, J. & Barbosa, S. (Publ.): Human-Computer Interaction – INTERACT 2007. Springer Berlin Heidelberg, pp. 141–155.

Rudolph, M. (2015). *User-friendly and Tailored Policy Administration Points*. 1st International Conference on Information Systems Security and Privacy (ICISSP), Doctoral Symposium.

Tidwell, J. (2011). *Designing Interfaces*. Sebastopol, CA: O'Reilly.

Uszok, A., Bradshaw, J. M., Johnson, M., Jeffers, R., Tate, A., Dalton, J. & Aitken, S. (2004). *KAoS policy management for semantic web services*. IEEE Intelligent Systems. 19(4), pp. 32–41.

Vollat, C. (2012). *Graphical User Interface Development for Usable Policy Administration Points (PAPs)*. Bachelor Thesis. Kaiserslautern, Germany: TU Kaiserslautern.

Whitten, A. (2004). *Making Security Usable*. PhD Thesis. Pittsburgh, PA: Carnegie Mellon University.