Model-Based Policy Derivation for Usage Control Enforcement

Prachi Kumari

kumari@cs.tum.edu
Technische Universität München, Germany
Supervisor: Prof. Dr. Alexander Pretschner, TUM

Abstract. Usage control is concerned with how data is used after access to it has been granted. In existing usage control enforcements, policies are assumed to exist; the derivation of implementation-level policies from end user specification-level policies has not been looked into. The behaviour users expect from a usage controlled system, may therefore differ from the actual behaviour. This research fills this gap. The thesis is that it is possible to adaptively derive implementation-level policies from specification-level policies in the context of usage control with limited input from system administrators (and no input from end users for the derivation of policies) in distributed systems. Policy derivation uses a model-based refinement of domain-specific abstractions like data and action in terms of technical constructs like events and states of systems. A methodological guidance to achieve this translation in a semi-automated way is also discussed.

1 Introduction

Usage control [1, 2] is an extension of access control that puts conditions on the future usage of data. At the level of end users, such requirements are expressed in abstract terms, for example, "picture may not be printed" or "document may not be distributed". Several enforcements of usage control exist for various policy languages [3–7] and at different layers of abstraction in the system [8–11]. These solutions focus on the enforcement of the policies and do not look into specification, translation, conflicts and other policy-related issues. In other words, they assume the policies to somehow exist and enable the enforcement of them. Such solutions have a drawback that system implementations of usage control policies might not always adequately reflect end user requirements. This is due to several reasons, one of which is the problem of mapping concepts in the end user's domain to technical events and artifacts. For instance, semantics of basic operators such as "copy" or "delete", which are fundamental for specifying policies, tend to vary according to context. For this reason they might be mapped to incomplete or incorrect sets of system events in enforcement. This might allow events that should have been inhibited and might wrongly block those that should have been allowed. I fill this gap by addressing the translation of specification-level policies into implementation-level policies in the context of the usage control

model introduced in [6] and later extended in [12]. *Enforcement of policies and related guarantees are not in the scope of this work.*

The problem. Usage control policies are specified as temporal and cardinality constraints on user actions on data. In order to translate specification-level policies, we need, firstly, to define the meaning of actions on data in technical terms. Secondly, because specification-level policies tend to be formulated in terms of the *future* usage of data, we need to transform the constraints to their *past* forms [6]; otherwise, the system would need to be able to look into the future. Thirdly, a well-defined methodology is required to automate the translation.

The solution. To address the first challenge, I propose a domain meta-model that captures the relationships among different artifacts in any domain, at different levels of technical details (**step 1**). This meta-model is then combined with an existing usage control model to formalize policy derivation (**step 2**). The second challenge, the translation of constraints, is essentially the problem of deriving past-time rules. As there is no generic way to formulate this derivation, this part of policy translation is handled via fixed rules. To go from policies to event-condition-action rules (that are finally deployed), predefined templates are used. The methodological aspects of policy derivation are addressed in **step 3** which runs parallel to step 1 and 2. Till step 3 however, I do not take into account the fact that the domain structure changes over time. Though unrealistic, this assumption is reasonable in order to simplify the problem for initial results. **Step 4**, which is also a part of the methodological aspect of the problem, handles the dynamic domain structure and describes the adaptive policy translation.

The rest of this paper is structured as follows: §2 argues about the relevance of this work with respect to related work; §3 discusses the general research methodology and briefly describes the results published so far. §4 discusses the current status and the approach to complete the thesis and, §5 concludes.

2 Related Work & Relevance

The goal of this work is to achieve a **model-based derivation of policies** in the context of **usage control**. In general, this aims to fill the gap between user and system requirements which is a fundamental problem in the area of Requirements Engineering. There are several approaches in the literature based on goals, models, clasifications, ontologies etc. that attempt to fill this gap by distinguishing between the user-level abstractions and the corresponding technical constructs at system levels. Some researchers have also addressed policy derivation targeting issues like conflict prevention, where the focus has been on the refinement of constraints rather than resources, objects or events [13].

Some of the prominent work on **policy derivation** have used resource hierarchies [14], commitment/obligations analysis [15], goal decomposition [16] and data classification [17]. In the context of **action refinement**, in [18], [19] and [20], ontology-based refinement techniques are described for semi-automated translation of access control policies. This is similar to my action refinement model because of the hierarchical structure of the resources considered. How-

ever, their policies are refined from the abstract level (users, resources and applications) to the logical level (user ids, resource addresses and computational commands like read/write); further technical representations of policy elements in concrete systems are not considered. [21] also addresses action decomposition for policy refinement: subjects perform operations on targets (services and devices) which are specified at a higher level. Using a system model and a set of refinement rules, actions are decomposed and one higher level policy is refined into multiple policies. In the context of giving **meanings to abstract actions**, although secure deletion is covered in [22], there is no further discussion on the different interpretations of deletion. In contrast to these works, I give a formal definition of policy refinement along with a generic way of modeling semantics of actions like "copy" and "delete". My approach is to classify the abstract and the concrete elements according to the technical details into different models and perform transformations on the elements of these models.

In the context of **usage control**, the only known work on policy derivation is described in [23]. A major difference with my work however is the absence of a generic and formal definition of the refinements. Other existing work on usage control, as mentioned in §2, focus on the enforcement part, derivation of policies from specification to implementation levels has not been addressed.

Contribution. Firstly, this work provides a formal definition of policy derivation and a generic way to model high-level actions like copy and delete for security policies, specifically, usage control policies. Secondly, along with adding to the expressiveness of the policy language, this work enables the derivation of security policies from specification to enforcement, in an automated way with limited user intervention. I am not aware of any other work that achieves automated policy derivation and deployment for usage control. Hence the contribution of this thesis with respect to related work.

3 Research Description & Results

Use Case. I explain the major steps of this work through an example from the social network domain: Alice wants to protect the pictures she posts in her profile so that her friends who get access to them should not be able to make local copies of those pictures. Alice can specify the policy "never copy photos" using one of the policy templates described in **step 3** below. Because of the domain model and the mappings already provided by an administrator (**step 1**), this policy is translated to many technical policies at various levels of abstraction (**step 2**) and, sets of implementation-specific executable rules are generated, deployed and enforced in the machines of Alice's friends who access her photos. If her friends update/change the systems installed on their machines, the adaptive policy translation accommodates these changes (**step 4**).

Step 1: The Domain Meta-Model. As user actions and data in usage control policies vary according to the domain context, I address the problem at the domain-level. In the meta-model, I distinguish between data and user actions

(the platform-independent model), their corresponding technical representations (the platform-specific model) and, the implementations of those technical representations (the implementation-specific model). This is analogous to the MDA viewpoints [24] with a minor difference in the naming of the different levels. The meanings of all the model elements starting from action and data at the top level are provided by mapping them to their next lower technical representations. A detailed description of the meta-model, a high-level translation methodology and the initial evaluation by example has been published in [25].

Step 2: Policy Derivation. For derivation of policies, the domain meta-model of step 1 is combined with the extended usage control model of [12]. This is useful in formalizing the semantics of the meta-model mappings. Additionally, it also enables another type of action refinement: in terms of the states of the system as states are already part of the extended usage control model. A few new operators are also added to the policy language in order to model the semantics of actions like "copy". The combined model and the formalization of policy derivation that includes action refinement and rule-based future to past translation of conditions has been published in [26].

Step 3: Policy Derivation Methodology. In this step I address the methodological aspect that consists of the definitions of various models and the specification, derivation, instantiation and deployment of policies. The specification of policies is simplified via templates defined by an administrator which are later configured by the end user. The policy derivation is semi-automated. One of the reasons that the complete process cannot be automated is that one specification-level usage control policy can be enforced in different ways and a decision making on enforcement strategy is required. For example, "never copy photos" can be enforced by altogether inhibiting a copy action or replacing the original photo by a predefined one or, allowing the copy with logs. I plan to partially automate this decision making with templates where majority of cases are covered by the predefined templates and only "exceptions" are handled by the administrators. The complete policy derivation methodology is discussed in [25].

Step 4: Adaptive Policy Derivation. To provide a realistic solution to the problem at hand, we must get rid of the assumption of static domain structures. However, if the systems evolve over time, we need a way to keep track of all the changes that take place. Additionally, we must handle cases of pending obligations and modification of action refinements in the domain. For this, I am working on a methodological guidance to dynamically accommodate changes in the domain structure. Intuitively, this includes considering all cases when the domain model needs to be updated, provide interfaces for systems to register, publish their technical artifacts and eventually de-register.

Evaluation. The complete research work is planned in two phases: the derivation of policies for static domains; and the extension of the framework to enable the adaptive policy derivation. The results of each phase are planned to be evaluated at the end of the phase by instantiating the translation of policies for existing usage control enforcements for two use cases: one, a web-based social

network example with Firefox, X11 and OpenBSD; two, a semi-closed enterprise setup with Thunderbird and Windows 7 OS. The results are to be evaluated with respect to (i) requirements and (ii) related work answering if *this work can be classified as filling a gap or as improving an existing process or both.* Also, insights on generic semantics of actions are to be discussed.

4 Current Status & Future Work

By the time of submission of this paper, most of the work described in steps 1-3 has already been done for the translation of policies in static domains. This also includes the investigation of typical specification-level policies in a web-based social network and the design of templates to be used by end-users to specify usage control policies. The respective publications at each step are mentioned in §3. Currently, I am implementing static policy translation for a semi-closed enterprise setup with Thunderbird and Windows 7. I am also working on the adaptive semantics of actions and writing the dissertation. As next steps, I will extend the implementations of policy translation for adaptive cases. I plan to complete the work described in §3 by Autumn 2014.

5 Conclusion

Through this research, I want to fill the gap between the end user's understanding of usage control policies and the actual enforcement of them in real systems. For this, I provide a way to define the meanings of abstract constructs in end-user policies. It is hard to establish a notion of correctness between the meanings of low-level and high-level policies because the semantics of high-level propositions is not explicitly defined but rather exists in the user's mind. Hence the correctness of the policy derivation is bound by that of the domain model on which the derivation is based. The contribution comprises both the technical and the methodological aspects of deriving implementation-level policies from specification-level policies. It might be contended that the latter are too complex for end users to understand and specify them at all. In that case, data protection officers or any other trusted authority might specify these policies for particular domains. Besides users, service providers would also benefit from this research because their contracts/terms of use could be audited for compliance.

References

1. A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, 2006.
2. J. Park and R. Sandhu. The UCON ABC usage control model. *ACM Trans. on Information and System Security*, 7(1):128–174, 2004.
3. R. Iannella (ed.). Open Digital Rights Language v1.1, 2008. `http://odrl.net/1.1/ODRL-11.pdf`.

4. Multimedia framework (MPEG-21) – Part 5: Rights Expression Language, 2004. ISO/IEC standard 21000-5:2004.
5. X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu. A logical specification for usage control. In *Proc. SACMAT*, pages 1–10, 2004.
6. M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In *Proc. ESORICS*, pages 531–546, 2008.
7. N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In *Proc. POLICY 1995*, pages 18–39.
8. M. Harvan and A. Pretschner. State-based Usage Control Enforcement with Data Flow Tracking using System Call Interposition. In *Proc. 3rd Intl. Conf. on Network and System Security*, pages 373–380, 2009.
9. A. Pretschner, M. Buechler, M. Harvan, C. Schaefer, and T. Walter. Usage control enforcement with data flow tracking for x11. In *Proc. STM 2009*, pages 124–137.
10. L. Desmet, W. Joosen, F. Massacci, K. Naliuka, P. Philippaerts, F. Piessens, and D. Vanoverberghe. The S3MS.NET Run Time Monitor: Tool Demonstration. *ENTCS*, 253(5):153–159, 2009.
11. P. Kumari, A. Pretschner, J. Peschla, and J. Kuhn. Distributed data usage control for web applications: a social network implementation. CODASPY '11.
12. A. Pretschner, E. Lovat, and M. Buechler. Representation-independent data usage control. In *Proc. 6th Intl. Workshop on Data Privacy Management*, 2011.
13. Steven Davy, Brendan Jennings, and John Strassner. Policy conflict prevention via model-driven policy refinement. In *Proc DSOM 2006*, pages 209–220.
14. L. Su, D. Chadwick, A. Basden, and J. Cunningham. Automated decomposition of access control policies. In *Proc. POLICY 2005*, pages 6–8.
15. J. Young. Commitment analysis to operationalize software requirements from privacy policies. *Requirements Engineering*, 16:33–46, 2011.
16. A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. A goal-based approach to policy refinement. In *Proc. POLICY 2004*, pages 229–239.
17. Y.B. Udupi, A. Sahai, and S. Singhal. A classification-based approach to policy refinement. In *Proc. 10th IFIP/IEEE IM*, 2007.
18. J. Beatty and J. Hulgan. Experiences with a requirements object model. Lecture Notes in Comput. Sci., pages 104–117. Springer Berlin / Heidelberg, 2009.
19. A. Guerrero, V.A. Villagrá, J.E. López de Vergara, A. Sánchez-Macián, and J. Berrocal. Ontology-based policy refinement using swrl rules for management information definitions in owl. In *DSOM*, pages 227–232, 2006.
20. B. Aziz, A.E. Arenas, and M. Wilson. Model-based refinement of security policies in collaborative virtual organisations. ESSoS, pages 1–14, 2011.
21. R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman. Decomposition techniques for policy refinement. In *Proc CNSM '10*, pages 72 –79, 2010.
22. Joel Reardon, David Basin, and Srdjan Capkun. Sok: Secure data deletion. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 301–315, Washington, DC, USA, 2013. IEEE Computer Society.
23. R. Neisse and J. Doerr. Model-based specification and refinement of usage control policies. In *Proc. PST'2013*, pages 169–176.
24. J. Miller and J. Mukerji. Mda guide version 1.0.1. Technical Report omg/03-06-01, Object Management Group (OMG), June 2003.
25. Prachi Kumari and Alexander Pretschner. Deriving implementation-level policies for usage control enforcement. CODASPY '12, pages 83–94. ACM, 2012.
26. Prachi Kumari and Alexander Pretschner. Model-based usage control policy derivation. In *Proc. ESSOS 2013*, pages 58–74. 2013.