Generating Monitors for Usage Control

Frederik Deckwerth
Frederik.Deckwerth@es.tu-darmstadt.de *

Real-Time Systems Lab,
Technische Universität Darmstadt,
D-64283 Merckstraße 25, Darmstadt, Germany

Abstract: Protecting computational resources and digital information against unauthorized access is one of the fundamental security requirements in modern computer systems. Usage control addresses the control of computational resources after access has been granted. Despite its fundamental importance, no systematic methods exist to implement formal usage control specifications. This paper presents a model driven approach to solve this problem based on graph transformation. Using the precise semantics of graph transformation, access control models and policies can be formally analyzed in an early phase. The existing solutions on automated verification and efficient implementation of graph transformation show that this approach is suitable to address security concerns throughout the overall software development process.

1 Introduction and Motivation

Protecting computational resources and digital information against unauthorized access is one of the fundamental security requirements in modern computer systems. *Access control models* provide a formal foundation for expressing *access control rules* that specify which *subjects* (e.g., persons) are authorized to perform a specific *operation* (e.g., read or write) on a certain *object* (e.g., a computational resource). A set of access control rules intended to protect a specific system is called an *access control policy*. An *access decision* is the process of evaluating an access control policy to decide whether a given subject is authorized to perform a given operation on a given object.

Access decisions are typically made once for a request, and in case of a positive authorization decision, no ongoing control of the resource usage is performed. Nowadays, as digital information is exchanged in distributed computer systems, there is an increasing interest in controlling access beyond the initial access decision, which is addressed by the proposed usage control model (UCON) [PS04].

As in existing access control models, UCON utilizes attributes for access decisions that characterize subject and object properties. Addidionally, the main innovations of UCON are (i) mutable attributes and (ii) obligations. *Mutable attributes* can change their values as

*Supported by CASED (www.cased.de)

a result of access decisions or other not directly influenceable factors. As a consequence, authorizations can become invalid during an ongoing access, which requires the access decisions to be reevaluated every time an attribute changes. As an example consider policies that require the reduction of an account balance based on resource usage. *Obligations* specify mandatory tasks to be performed by a subject before (pre-obligation) or during (on-obligation) resource usage. While on-obligations have to be evaluated continuously during resource usage, pre-obligations require a "history function" to check if the mandatory activities where previously performed. For example, medical personnel must sign a privacy policy before reading patient records.

Most access control models (including UCON as defined in [PS04]) do not explicitly support access decisions based on structural relational dependencies between objects and subjects. Although such dependencies can be expressed by arbitrary valued attributes, making them explicit often facilitates the specification and implementation of access control policies. Consider, for example, a social network scenario where access decisions are mainly based on friendship relations. Expressing this as an attribute requires every user to maintain an attribute (e.g. a list) that stores the names (i.e. unique identifiers) of all friends, which has to be kept consistent. In this case, encoding "friendship" as a structural relation (e.g., as an association) can lead to simpler and more readable access control policies.

Mutable attributes, obligations and explicit structural constraints impose new requirements concerning the formalism for specifying and the techniques for implementing such systems. The formalism should capture the combination of temporal aspects introduced by the obligations as well as structural relation. The need for a continuous evaluation of attributes and structural constraints combined with temporal constraints requires non-trivial run-time monitoring techniques.

As a consequence, deploying access control subsystems in real-world applications requires a systematic approach to realize systems that behave as expected. From an end-user perspective, the following basic requirements can be identified:
Formal model of the access control policy. To faithfully capture and analyze access control policies, a theoretical framework is needed that (i) provides a set of high-level concepts to specify a policy and (ii) is amenable to formal analyzing and verifying desired properties.
Systematic implementation. In practical applications, the security of usage control system does not only depend on the correctness of the policy, but also on its correct implementation. To implement a policy, the high-level specification has to be translated into an executable implementation. The more systematic this translation, the easier it is to ensure that the properties proven for the policy model also hold for the implementation.
Seamless integration. As systems are typically not built from scratch, it must be possible to integrate access control modules into legacy systems. To this end, it must be possible to tailor the implementation to the needs of the system.

2 Related Work

Several formal languages [Mar07, PHB06] have been introduced to express UCON policies with temporal logic or process algebra. These approaches focus on the specification of the control flow (i.e., the sequence of actions required to gain access) and do not address complex structural constraints. On the other hand, there exist different implementations to enforce usage control [GNC10, WPH11]. Most of these approaches focus on a specific application domain such as service-oriented architectures or assume specific infrastructures. Other more general approaches offer implementations of access control decision modules [NPDG11, KZB+08], which can handle policies written in a specific language, but are difficult to integrate into legacy systems as they cannot be easily adopted. Although [KP12] investigates how to systematically translate high-level policy specifications to implementation specific policies, their implementation is based on a static policy decision module and requires a manual implementation of modules for managing operations and attributes. Complex structural constraints are also not supported.

3 Sketch of the Solution

I propose a model-driven approach based on graph transformation, which is a rule-based description for the manipulation of graph structures. Based on the precise semantics of graph transformation, access control models and policies can be formally analyzed in an early phase. The existing solutions on automated verification and on translation of graph transformation specifications into efficient implementations show that this approach is suitable to address security concerns throughout the overall software development process. Nevertheless, there are still open challenges regarding UCON. Existing approaches based on temporal logic or process algebra are suitable for expressing temporal constraints but lack support for expressing structural constraints. The situation for a graph transformation based approach is exactly the opposite: structural constraints can be easily expressed and verified, but temporal constraints are difficult to handle as graph transformation does not offer an explicit mechanism for the control flow. As none of the approaches is necessarily superior, I propose to investigate how to combine the strengths of both approaches. This leads to several challenges regarding the formalism as well as its efficient implementation.

3.1 Expressing Access Control Policies with Graph Transformation

The general viability of graph transformation to express and analyze access control models and policies is illustrated in [KMPP02]. The approach demonstrates that role based access control (RBAC) [SCFY96] with separation of duty constraints, which prohibit assigning two particular roles to the same subject, can be expressed by graph transformation. The basic idea is to model the system state by a graph called a *system model* and state changes by applying graph transformation rules to the system model. A graph transformation rule

consists of three graphs called the left-hand side (LHS), the right-hand side (RHS), and the negative application condition (NAC). The LHS together with a NAC defines the application condition of a rule. A rule is applied to the system model by (i) finding a match (i.e., a structural identical part) of the pattern described by the LHS, (ii) checking the negative application condition (which prohibits the presence of certain structures) and (iii) replacing the match in the system model by the RHS. Separation of duty constraints are mapped to *graph constraints* which is a graph that specifies a forbidden pattern. A system model is said to be in an allowed state if it does not contain a forbidden pattern. By proving that none of the rules can transform an allowed state into a forbidden one, it can be guaranteed that the system never evolves into a forbidden state, provided that the initial system model was an allowed state. Such a static analysis can be performed in an automated fashion. Moreover, the rules can be adopted automatically by adding NACs that prevent rules to be applied for exactly those situations that would lead to a forbidden system model state.

This powerful constructive approach is well understood for structural patterns [HW95]. As UCON strongly relies on non-structural constraints (e.g., constraints on integers or strings) it has to be investigated how the approach can be extended to also handle non-structural constraints. From a theoretical point of view, non-structural constraints can be treated as invariants and proven inductively. However, in general applications, such invariants are hard to check automatically. Based on the rule based character and explicit treatment of NACs, graph transformation might be feasible for such an automated analysis. Inspired by the treatment of non-structural constraints [AVS12], I consider an approach that can verify structural as well as non-structural constraints as feasible.

As previously discussed, obligations require temporal constraints to express before/after relations. The integration of temporal logic with graph transformation has been studied in [GHK00, RS06] from a theoretical point of view. However, regarding the implementation, it is still an open topic how to efficiently realize run-time monitors that can evaluate constraints combining structural and temporal expressions.

3.2 Implementation

Several graph transformation engines have been proposed to execute a formal graph transformation specification or to check constraints in a graph. All engines have to solve a common problem: the efficient querying of complex graph-based model structures, which can be expressed as an NP-complete graph pattern matching problem. In general, the solutions can be categorized into batch and incremental approaches. While batch algorithms start the search for a pattern for each request from scratch, incremental approaches explicitly store all matches and incrementally maintain these matches when the system model is updated. Using this approach, it is possible to obtain all matches of a pattern in constant time, which makes it ideally suited for monitoring graph structures. However, this is achieved at the expense of extra memory for maintaining the matches and additional time in the update phase.

During my previous work, I was involved in developing heuristics for improving the run-

time of an batch pattern matching process [VDWS12]. These techniques may be also applied to incremental approaches to reduce the time for updates and memory consumption.

Monitoring temporal constraints requires a "history function" to check if an activity was performed in the past. As access control systems usually run for long periods, logging all activities is not feasible. Several algorithms [BKM10, MJG+12] to efficiently maintain such "history functions" exists, however, a combination of these approaches with incremental pattern matching is still an open topic.

4 Plan of Action

I plan to implement a framework based on graph transformation for (i) modeling usage control policies (ii) formally analyzing usage control policies and (iii) generating efficient monitors for usage control, which requires appropriate extensions of the pattern matching engine that is currently under development at our group [VAS12]. More specifically:

Modeling usage control policies. In a first step it has to be investigated how the usage control model can be formalized using graph transformation. This includes how to exactly identify the concepts that are difficult to express by graph transformation. Based on these results, an appropriate extension to graph transformation can be developed.

Analyze and formal verify access control policies. It has to be investigated to what extent constraints that combine structural and non-structural restrictions can be verified statically. Moreover, it has to be investigated whether this is also possible for temporal constraints. Constraints that cannot be statically verified have to be checked at runtime by monitoring techniques.

Generate efficient monitors for usage control. In the first step, the pattern matching engine currently developed at our group has to be extended to support incremental pattern matching. In a next step, it has to be investigated how temporal aspects can be integrated.

Finally, the viability of the approach has to be evaluated based on a non-trivial case study. Additionally, I plan to develop a benchmark to provide a basis for a performance comparison with other approaches.

References

[AVS12] A. Anjorin, G. Varró, and A. Schürr. Complex Attribute Manipulation in TGGs with Constraint-Based Programming Techniques. *Electronic Commun. of the EASST*, 49, 2012.

[BKM10] D. Basin, F. Klaedtke, and S. Müller. Monitoring security policies with metric first-order temporal logic. In *Proc. of*, SACMAT '10, pages 23–34. ACM, 2010.

[GHK00] F. Gadducci, R. Heckel, and M. Koch. A Fully Abstract Model for Graph-Interpreted Temporal Logic. In Hartmut Ehrig, Gregor Engels, Hans-Jrg Kreowski, and Grze-

gorz Rozenberg, editors, *Proc. of TAGT '00*, volume 1764 of *LNCS*, pages 310–322. Springer, 2000.

[GNC10] G. Gheorghe, S. Neuhaus, and B. Crispo. xESB: An Enterprise Service Bus for Access and Usage Control Policy Enforcement. In Masakatsu Nishigaki, Audun Jsang, Yuko Murayama, and Stephen Marsh, editors, *Trust Management IV*, volume 321 of *IFIP*, pages 63–78. Springer, 2010.

[HW95] R. Heckel and A. Wagner. Ensuring consistency of conditional graph grammars-a constructive approach. *ENTCS*, 2:118–126, 1995.

[KMPP02] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A graph-based formalism for RBAC. *ACM Trans. Inf. Syst. Secur.*, 5(3):332–365, August 2002.

[KP12] P. Kumari and A. Pretschner. Deriving implementation-level policies for usage control enforcement. In Proc. of CODASPY '12, pages 83–94. ACM, 2012.

[KZB+08] B. Katt, X. Zhang, R. Breu, M. Hafner, and J. Seifert. A general obligation model and continuity: enhanced policy enforcement engine for usage control. In Proc. of SACMAT '08, pages 123–132. ACM, 2008.

[Mar07] F. Martinelli. A model for usage control in GRID systems. In *SecureComm '07*, page 520, 2007.

[MJG+12] PatrickOâNeil Meredith, Dongyun Jin, Dennis Griffith, Feng Chen, and Grigore Rou. An overview of the MOP runtime verification framework. *International Journal on Software Tools for Technology Transfer*, 14:249–289, 2012.

[NPDG11] R. Neisse, A. Pretschner, and V. Di Giacomo. A Trustworthy Usage Control Enforcement Framework. In *Proc. ARES '11*, pages 230 –235, aug. 2011.

[PHB06] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, 2006.

[PS04] J. Park and R.S. Sandhu. The UCONABC usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

[RS06] T. Rötschke and A. Schürr. Temporal Graph Queries to Support Software Evolution. In A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, editors, *ICGT '06*, volume 4178 of *LNCS*, pages 291–305. Springer, 2006.

[SCFY96] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, 1996.

[VAS12] G. Varrø', A. Anjorin, and A. Schürr. Unification of Compiled and Interpreter-Based Pattern Matching Techniques. In Antonio Vallecillo, Juha-Pekka Tolvanen, Ekkart Kindler, Harald Strrle, and Dimitris Kolovos, editors, *ECMFA '12*, volume 7349 of *LNCS*, pages 368–383. Springer, 2012.

[VDWS12] G. Varró, F. Deckwerth, M. Wieber, and A. Schürr. An Algorithm for Generating Model-Sensitive Search Plans for EMF Models. In Z. Hu and J. Lara, editors, *Proc. of ICMT*, volume 7307 of *LNCS*, pages 224–239. Springer, 2012.

[WPH11] A. Wahl, S. Pfister, and B. Hollunder. Complex Event Processing for Usage Control in Service Oriented Infrastructures. In *Proc. SERVICE COMPUTATION '11*, pages 92–97, 2011.