Towards a Usage Control Policy Specification with Petri Nets

Basel Katt[1], Xinwen Zhang[2], and Michael Hafner[1]

[1] University of Innsbruck, Austria
[2] Samsung Information Systems America, San Jose, CA, USA
{basel.katt,m.hafner}@uibk.ac.at, xinwen.z@samsung.com

Abstract. Access control aims at restricting access to resources instantly. However, in collaborative computing environments with shared resources and distributed right management systems more advanced controlling mechanisms are required. For example, the control of the usage of a resource may need to be continuous, obligations is required, and concurrency is an important aspect when different users use a shared resource. To overcome these shortcomings of traditional access control, usage control has been proposed and investigated recently. In this paper we introduce a new usage control policy specification. Beyond existing approaches, the novelty of our policy is threefold: first, the ability to integrate the functional and security aspects of the system, thus lending support to control system behavior continuously. Second, post obligation is supported in a way that a violation of any rule during the current usage session, or after it ends, can affect the decisions of future usages. Finally, concurrency rules are embodied in the policy model, thus concurrent usages by different users to shared resources are controlled.

1 Introduction and Related Work

Usage Control has recently received attention in literature. It aims at meeting the new security requirements for distributed and collaborative environments. For example, collaborative computing applications, DRM systems and applications that process privacy sensitive data require that (1) control must be applied after a resource is released and must be applied continuously for its entire usage period, (2) obligatory tasks must be enforced, (3) temporal, cardinal, and/or periodic rules must be supported, and finally (4) concurrency must be considered when resources are shared. These new requirements stipulate the need for new policy languages and advanced enforcement mechanisms. Using this argument, we define *Usage Control* as *the ability to continuously control the usage of shared resources in a distributed environment. Continuously* means that the control must be ongoing and continuous for the whole period of a resource usage. *Control* means that permissions must be checked, obligatory tasks must be enforced, and environmental conditions must be considered. Furthermore, control can be applied instantly, temporarily, repeatedly, and/or periodically. *Shared* indicates that resources can be accessed by different users at the same time. Finally,

R. Meersman, T. Dillon, P. Herrero (Eds.): OTM 2009, Part II, LNCS 5871, pp. 905–912, 2009.

distributed environment allows the assumption that resources can be located on clients and/or servers.

The main approaches in the literature dealing with usage control are UCON model [7] and its specifications like in [9,3,6] and OSL [1]. UCON is a general model to capture the major security requirements of usage control. The policy specifications of UCON recognize the importance of a continuous and ongoing control, provide (pre- and on-)authorization and obligation rules, and supports (attribute) mutability. However, the current UCON model and its policy specifications lack post-obligation and concurrency support. Later, the authors in [2] considered the concurrent usages. They propose a static analysis method by creating the dependency graph between different controllers/users, however fail to define concurrency rules for users within one controller. In previous work [5] we proposed an extension of UCON specification to enhance its obligation expressiveness, however our approach lacked the ability to express periodic obligations, application behavior was not considered, and it was done in an ad-hoc manner that makes the resulting policy difficult to analyze. OSL, on the other hand, proposes an expressive policy language for usage control considering temporal, cardinal and permit rules. However, the continuity aspect of control beyond one action, the integration with the application behavior, mutability, and concurrency issues tend to be overlooked.

Based on our definition of usage control and the drawbacks we identify in the current approaches, we propose in this paper a usage control policy specification that embodies the following features. First, we emphasize the concept that controlling the usage of a resource must be continuous and ongoing during the entire usage period. In order to ensure a continuous and ongoing usage, the required behavior of the usage application must be identified. That means that the integration of the functional (behavioral rules) and non-functional (security rules) aspects is essential in the area of usage control. Based on this observation, we propose that a usage control policy can be interpreted as *the behavior of the reference monitor when an application uses a resource*, which embodies behavioral, security, and concurrency rules.

The second and closely connected with the first feature is the *concurrency* feature. When dealing with *stateful* usage control policy and shared resources, concurrency becomes a relevant issue. In real-world distributed and collaborative environments, resources are shared. This fact requires placing constraints on how concurrent usages of resources are regulated. Finally, the above discussion of the term *control* in the context of usage control furnishes a solid grounding for the necessity of an *expressive* policy language. A comprehensive requirements study of usage control is different application scenarios conducted in [8] showed, among others, that: first, decision factors of a usage are not confined to which credentials or attributes users posses, however it enfolds which tasks and actions must be fulfilled by users. For example, students registering for an online course are obliged to pay certain amount of fee within six months after being authorized using their valid student IDs. Second, it is required to express different types of authorization and obligation rules, such as instant, temporal, cardinal, and

periodic rules. The mentioned registration requirement, for example, comprises an instant authorization constraint and temporal obligation rule. This observation lends support to our further classification of authorization and obligation policies into *instant, temporal, cardinal* and *periodic* policies.

Contributions: In this paper we propose a policy specification for usage control based on Colored Petri Nets (CP-net). Our policy supports (1) the continuous control by defining behavioral rules in the form of petri nets (2) an expressive set of authorization and obligation rules, in specific post obligations, and (3) concurrency control for shared resources. CP-nets embody the Petri Nets' powerful modeling capabilities for concurrent system behaviors and the powerful specification support of the *ML* functional language. Furthermore, its mathematical foundation and the tool support enable and ease analysis and verification of the usage control policy.

Outline: In next section we give an overview of Coloured Petri Nets. Section 3 presents the main elements of our usage control policy, whose specification is discussed in section 4. Finally, we conclude this paper and discuss the future work in Section 5.

2 Overview of Colored Petri Nets (CP-Net)

A Colored Petri Net (CP-net) [4] can be defined as a tuple *CPN = (Σ,P,T,A,N,C,G,E,I)*, where Σ is a set of *color sets*, *P, T* and *A* are sets of *places/states, transitions* and *arcs*, respectively. *Arc expressions* are defined by the function *E*. *N* is a *node* function that determines the source and destination of an arc. *C* is a *color* function that associates a color set *C(p)* or a type with each place *p*. *G* is a guard function that maps each transition *t* to a boolean expression *G(t)*. For a transition to be enabled, a *binding* of the variables that appear in the arc expressions must be found. For this "binding element", which is a pair of a transition and a binding *(t,b)*, the guard function must evaluate to true. *I* is an *initialization* function that maps each place to a multiset *I(p)*. S_{MS} denotes the set of multi-sets over *MS*. A *marking M* is a function defined on a place *p* such that $M(p) \in C(p)_{MS}$, thus multiple tokens in the place can carry the same value. $M(p)$ denotes the marking of a place p in the marking M. If a binding element (t, b) is *enabled* in a marking M_1, denoted $M_1[(t, b)\rangle$, then (t, b) may *occur* in M_1 yielding some marking M_2. This is written as $M_1[(t, b)\rangle M_2$. Accordingly, a *finite occurrence sequence* is a sequence consisting of markings M_i and binding elements (t_i, b_i) denoted $M_1[(t_1, b_1)\rangle M_2 ... M_{n-1}[(t_{n-1}, b_{n-1})\rangle M_n$ and satisfying $M_i[(t_i, b_i)\rangle M_{i+1}$ for $1 \leq i < n$.

3 Usage Control Policy Elements

Similar to UCON specifications [9,3], the main elements of our usage control policy are: *subjects, objects, contexts* and *actions*. *Subjects* (S) are active principals that can perform actions on resources. We assume that every subject possesses

a unique identifier. Second, *Objects* (O) represent the resources that must to be protected and they are also distinguished by a unique identifier. Third, a system may contain a set of contexts or environmental entities (C). The system clock or system locator are examples of context entities. Furthermore, rules determining authorization and obligation decisions, in our policy, are based on attributes of subjects, objects, and contexts. Therefore, all entities in the system are specified by a set of attributes. subject role, the IP address of the user's machine, or the creator of an object is examples of attributes of subject, context and object, respectively. We define for each of these elements a new type (colset). Thus we have the following basic types: $\Sigma = \{SUB, OBJ, CON\}$, representing subjects, objects, contexts, respectively. Accordingly, we define three basic places that contains the available subjects, objects, and contexts, we call them *Subjects, Objects* and *Contexts* places.

Finally the usage control policy encompasses a set of actions (Act), which can be classified into the following categories: *subject actions* ($SA \subseteq Act$), *enforcement actions* ($EA \subseteq Act$), and *obliged actions* ($OA \subseteq Act$). Each action has a name and an optional set of parameters. We use action name to refer to an action, which implies also its parameters. Subject actions are actions that subjects execute on resources and they are represented as CP-net transitions. Enforcement actions are those that a reference monitor can execute and thus do not need fulfillment check. The reference monitor is responsible for executing these actions to enforce any made decision or perform additional tasks required to compensate violations of policy constrains. Updating subject/object attributes, logging a failed access, or checking a fulfillment of an obligation are examples of enforcement actions. These actions are specified as an ML functions of the net representing the usage control policy. Finally, obliged actions are actions that must be fulfilled by the subject. They are represented by their names/IDs and associated with fulfillment check actions.

4 Usage Control Policy Specification

Beside expressive authorization and obligation, including post-obligation, the main features of our usage control policy are the continuity of control by introducing behavioral rules and concurrency control. The integration among these types of rules results in the usage control policy.

4.1 Behavioral Rules

In order to ensure continuity of control, the functional behavior of the system must be considered and controlled. The determination of how applications, which use protected resources, must behave (safe behavior) is encoded in behaviroal rules. These rules are represented as a CP-net (called a Usage Pattern). The following conditions must be considered when defining a usage pattern, illustrated in Figure 1:

1. Transitions represent subject actions; and places, which are of subject type (SUB), indicate the state of each user using the resource.
2. Objects and Contexts places are connected to all transitions of the behavior using double headed arcs.
3. The usage pattern has one *Initial* place and one *End* place, which represents the states of the normal ending of the usage session by the user.
4. The *Subjects* place is connected to the *Initial* place by a start/join session transition *(str)* and the *End* place is connected to the *subjects* place by a close/leave session transition *(end)*.

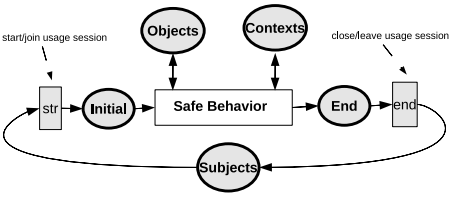

Fig. 1. The structure of a Usage Pattern

Definition 1. *A usage pattern is defined as a Coloured Petri Net that fulfills the above mentioned conditions. The set of all usage patterns is denoted by* $UP \subset CPN$.

4.2 Authorization and Obligation

An authorization rule defines the rights of a subject to use a resource and an obligation rule determines the tasks/actions that a subject must perform when a resource is used. Each one of these rules can be further classified. Authorization rules can be *instant, temporal, or cardinal* and obligation rules can be *instant, temporal, or periodic*. In the following we use the set theory to define the different types of authroization and obligation rules. Later, we show how these rules are encoded in the usage control policy.

Single Instant Access Right

Definition 2. *A single instant access right is defined as 6-tuple* $ir=(s,o,c,sa,[ea1,ea2]) \in IR$. *This right indicates that a subject* $s \in S$ *is allowed to perform an action* $sa \in SA$ *on an object* $o \in O$ *in the context defined by* $c \in C$, *where ea1 and ea2 are optional enforcement actions that must be executed by the reference monitor if the action is allowed or denied, respectively. Function* $ir.authCheck : Attr(S) \times Attr(O) \times Attr(C) \rightarrow Bool$ *indicates an enforcement action that takes as parameters subject, object and context attributes and returns a Boolean value indicating whether the subject is authorized or not. It defines the constraints or conditions that must be fulfilled to authorize the subject.*

Single Temporal Access Right

Definition 3. *A single temporal access right is defined as a triple tr=(ir,t,[ea]) $\in$ TR, where ir is an instant access right, $t \in N$ is a time interval, within which the right is granted, and $ea \in EA$ is an optional enforcement action that must be executed when the execution is revoked.*

Single Cardinal Access Right

Definition 4. *A single cardinal access right is defined as a triple cr=(ir, n, [ea]) $\in$ CR, where* ir *is an instant access right, $n \in N$ is the number of times subjects are allowed to perform the action specified in ir, and ea$\in$ EA is an optional enforcement action that must be executed after the execution is revoked.*

Single Instant Obligation

Definition 5. *A single instant obligation is defined as 6-tuple $iobl = (s, o, sa, oa, [ea1, ea2]) \in IO$, which indicates that a subject $s \in S$ is allowed to perform an action $sa \in SA$ on an object $o \in O$ if the obliged action $oa \in OA$ has been fulfilled by s, where ea1 and ea2 are optional enforcement actions that must be executed by the reference monitor if the action is allowed or denied, respectively. Function $iobl.oblCheck : S \times OA \rightarrow Bool$ indicates an enforcement action that takes an obliged action and a subject as input parameters and returns a boolean value indicating whether the subject has fulfilled the obliged action.*

Single Temporal Obligation

Definition 6. *A single temporal obligation is defined as 6-tuple $tobl = (s, o, sa, oa, ea, t, [ea1]) \in TO$, which indicates that a subject $s \in S$ is allowed to execute an action $sa \in SA$ on an object $o \in O$. However the obliged action $oa \in OA$ has to be fulfilled by s within the time interval $t \in N$, otherwise the enforcement compensation action $ea \in EA$ is executed. Finally, $ea1 \in EA$ is an optional enforcement action that must be executed when the action is allowed.*

Single Periodic Obligation

Definition 7. *A single periodic obligation is defined as 8-tuple $pobl = (s, o, sa, oa, ea, t, n, [ea1]) \in PO$, where s, o, sa, oa, ea represent the subject, object, subject action, obliged action, and enforcement compensation action, respectively. $n \in N$ is the number of times the obliged action must be fulfilled, and $t \in N$ defines the duration within which the obliged action must be fulfilled. Finally, ea1 is an optional enforcement action that must be executed when the action is allowed.*

4.3 Concurrecny

A concurrency rule expresses the way multiple users are allowed to use a resource at the same time, which can be categorized into *cardinal* and *type*. *Cardinal* indicates how many subjects are allowed to execute an action on a resource at the same time, while *type* indicates whether the concurrent execution is true or interleaved.

Definition 8. *A concurrency rule is defined as a triple* $conc = (sa, Par\_Set, n) \in CONC$, *where* $sa \in SA$ *is a subject action,* $Par\_Set \subseteq S$ *is a set of subjects that are allowed to execute the action sa in a true concurrent way, and* $n \in N$ *is the maximum number of subject that are allowed to execute the subject at the same time. We call* Par_Set *the true concurrent set.*

4.4 Usage Control Policy

To build the usage control policy from the single elements we define the UCPN (Usage Control Coloured Petri Nets).

Definition 9. *A Usage Control CPN is defined as a tuple UCPN = (UP,R,EA,EP), where UP is the finite set of usage patterns, R is a rule label function that maps each transition to a set of security/concurrency rules, EA is a finite set of "end_arcs", and* $EP \in P$ *is the (abnormal) end place.*

First, the rule function is defined as $R : T \rightarrow P(Ru)$, where $Ru = \{IR \cup TR \cup CR \cup IO \cup TO \cup PO \cup CONC\}$ is the set of all security and concurrency rules, and $P(Ru)$ denotes the powerset of Ru. Furthermore, This function binds (1) subject, object and context elements of different security/concurrency rules into subject, object, and context variables in the arc expressions of the arcs connected to the transition *tr*; and (2) all enforcement/obliged actions and other elements defined in the rules into corresponding CP-net related ML functions/variables. Second, each usage control policy contains one *EP*, with a coloset/type *SUB*, which includes subject tokens for users whose usage was ended by the reference monitor abnormally due to a security/concurrency rule violation. This place is connected to the *Subjects* place through an end transition, similar to the *End* place of the usage pattern. Finally, *end_arcs* are a special kind of headed arcs, denoted as a dotted line, that connect a transition with the end place *EP*, illustrated in Figure 2. Informally, the semantics of this arc indicates that

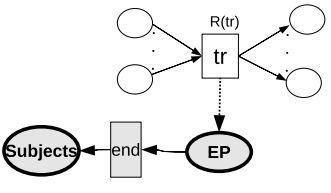

Fig. 2. An example of a UCPN policy

during the occurrence of *tr*, any violation of the associated rules defined in *R(tr)* results in moving the bound subject token, exclusively, through this arc to the *EP* place. All other output arcs of this transition will be ignored. This indicates that the violation of any rule causes the usage of the violated subject to be halted. Furthermore, additional (compenstation) enforcement actions can be defined for the end transition(s), in a way that it affects the future usages of this subject.

Based on the above discussion, we can finally define a *Usage Session* as a finite occurrence sequence $\sigma = M_0[Y_1...Y_n\rangle M_n$, where (1) the M_0 and M_n for all places except *Subjects, Objects* and *Contexts* is {}; (2) the first step Y_1 involves the occurrence of the *str* transition; and the last step Y_n involves the occurrence of an *end* transition connected to *PE* or *End* places. Any enforcement actions defined at end transitions allows enforcing post-rules (rules that must be enforced after the usage session is closed).

5 Conclusion and Future Work

In this paper we introduce a new usage control specification. We argue that functional and security aspects should be integrated in the context of usage control. Thus, beside authorization and obligation, including post-obligation, rules, we support defining behavioral and concurrecny control. The full enforcement semantics of the defined UCPN and its analysis is planned for future work. By doing so, we aim at closing the gap between a (usage control) security policy and the enforcement mechanisms of its reference monitor.

References

1. Hilty, M., Pretschner, A., Basin, D., Schaefer, C., Walter, T.: A policy language for distributed usage control. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 531–546. Springer, Heidelberg (2007)
2. Janicke, H., Cau, A., Siewe, F., Zedan, H.: Concurrent enforcement of usage control policies. In: IEEE Policy 2008 (2008)
3. Janicke, H., Cau, A., Zedan, H.: A note on the formalization of ucon. In: ACM SACMAT 2007 (2007)
4. Jensen, K.: Coloured Petri Nets, vol. 1. Springer, Heidelberg (1992)
5. Katt, B., Zhang, X., Breu, R., Hafner, M., Seifert, J.-P.: A general obligation model and continuity enhanced policy enforcement engine for usage control. In: ACM SACMAT 2008 (2008)
6. Martinelli, F., Mori, P.: A Model for Usage Control in GRID systems. In: ICST SecureComm 2007 (2007)
7. Park, J., Sandhu, R.: The ucon_abc usage control model. ACM TISSEC 7(1), 128–174 (2004)
8. Pretschner, A., Hilty, M., Schütz, F., Schaefer, C., Walter, T.: Usage control enforcement: Present and future. IEEE Security and Privacy 6(4), 44–53 (2008)
9. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal model and policy specification of usage control. ACM TISSEC 8(4), 351–387 (2005)