Enforcement of Privacy Requirements

Padmanabhan Krishnan, Kostyantyn Vorobyov

Enforcement of Privacy Requirements

Padmanabhan Krishnan[1] and Kostyantyn Vorobyov[2]

[1] paddykrishnan@ieee.org
[2] Centre for Software Assurance
Bond University, Gold Coast, Queensland, 4229, Australia
kvorobyo@bond.edu.au

Abstract. Enterprises collect and use private information for various purposes. Access control can limit who can obtain such data. However, the purpose of their use is not clear. In this paper we focus on the purpose of data access and demonstrate that dynamic role-based access control (RBAC) mechanism is not sufficient for enforcement of privacy requirements. To achieve this we extend RBAC with monitoring capability and describe a formal approach to determining whether access control policies actually implement privacy requirements based on the behaviour of the system. We demonstrate the advantages of our approach using various examples and describe the prototype implementation of our technique.

Keywords: Privacy protection, Access control, Formal analysis.

1 Introduction

Organisations collect, store and share information with individuals and other organisations. They need to respect the privacy of the entities they interact with and comply with legislative requirements. Privacy policies are used to specify how organisations handle the data in their interactions. These policies can also be shown to be compliant with legislation.

Privacy, especially privacy enhancing technologies (PETs), is focused on protecting an individual's information. Thus, issues such as identity management, user consent, data anonymisation and retention have been the focus of PETs. These issues have implications for enterprises, as they collect data from individuals and use it for various purposes. While access control can limit who can obtain the information, it is not clear (especially to an individual) how an enterprise restricts the use of data. This affects both, individuals (who may be reluctant to transact with an enterprise) and enterprises (which may be inadvertently breaching various privacy guarantees).

While technologies, such as encryption, access control and authorisation can be used to implement a policy, it is important to capture the privacy requirements. The policy then has to be developed from the requirements and finally one can develop enforcement mechanisms.

Policy authoring and enforcement are challenging issues. As it is not possible to anticipate all possible uses of data, it is difficult for designers to indicate the

appropriate policy. Thus, policies are often changed when inappropriate usage (i.e., a breach) is detected. Consequently, it is easier to specify what happens when a breach occurs [1]. The difficulty of writing privacy policies is increased as privacy does not have a standard meaning. Each person is likely to have a different interpretation, which could also depend on the application domain [2]. Additionally, privacy is context dependent and would depend on the user and also the queries handled by the system [3]. Finally, it is important for policies not to impede normal behaviour [4].

The purpose of data access [5, 6] has attracted attention, especially as there are often conflicting issues between organisations and individuals. For instance, in health systems the importance of surveillance indicates that not all personal information may be private. In general, the purpose for which data is used is important in privacy. Users give permission to enterprises for specific tasks (and they assume that the data will not be used for other tasks). For example, Facebook's privacy policy states that they can use the information they receive for any services they provide including making suggestions of new connections. This is a very broad policy, as anything can be viewed as a service. Amazon allows users to opt out of receiving promotional offers. However, it is not clear if the user's information is not used in creating such offers. Amazon also states that they will not share personally identifiable information to third party providers. But what is personally identifiable is not clearly stated.

Personally identifiable information could include name, date-of-birth, address and national identity number. The chances of identifying an individual from a collection depends on the data. For example, a commonly occurring name or a specific date-of-birth in a census data is unlikely to identify an individual. However, by combining various data types personal information can be identified. Thus, it is important to control the collection of accesses rather than only a single data access.

The main contribution of this paper is a formal approach to determining if access control policies actually implement privacy requirements given a behaviour. We show how a dynamic access control mechanism is not sufficient to enforce privacy requirements. We need to extend the access control mechanism with some monitoring capability.

A prototype implementation that supports this approach is also described. The usefulness of this approach is demonstrated via various examples. In Section 2 we present the formal details. In Section 3 we give some simple examples that illustrate our approach and in Section 4 we give a description of a proof-of-concept implementation of our technique. Section 5 presents a review of the related work. Finally, we draw some conclusions and describe future directions in Section 6.

2 Framework

A specific system in our framework consists of an automaton that represents the behaviour (such as gathering and using the gathered information) and an

automaton that represents a controller (including access control). For the behaviour automaton we do not separate individuals into separate automata. We have a single automaton where the label on the transition has the action as well as the individual (and role) who performs the action. The controller can observe actions exhibited by the behaviour automaton, but can also prevent certain actions. This is achieved using dynamic role based access control (RBAC) [7] and a simple semantics of purpose.

Before we describe the formal details, especially of the access control part, we present a motivating example. Assume that Alice releases some personal information. This information can be used for internal purposes, but cannot be used for marketing purposes. Assume Bob can access this information and can decide how to use it. But Bob has to be prevented from using it for marketing purposes. One way is to force Bob to assume different roles for each use. This, however, could increase the number of roles. Also, then there is no difference between purpose (which is a semantic concept) and roles (which is an enforcement concept). Furthermore, the access control mechanism will have to permit and then withdraw the role being assumed. Therefore, it is better to tag certain actions with predicates that represent purposes. The access control entity now either permits or disallows actions. This can be viewed as a mixture of access control and workflow transition enabling.

Thus the control automaton's alphabet will consist of normal actions, access control actions (i.e., permitting and withdrawing roles) and purpose related actions.

We define a composition operator that combines the behaviour automaton with the controller. The composition is based on synchronisation on common actions. However, the access control automaton cannot prevent behaviour purely via the synchronisation requirements. Hence, the composition operator allows actions to occur when the access control automaton cannot exhibit an action.

2.1 Formal Details

The main focus of the formalism is to describe the interactions between behaviour, access control and privacy policies.

We assume a set of atomic access control actions indicated by $\mathcal{A}$. These actions correspond to operations on data elements. We also assume a set of individuals $\mathcal{I}$ and a set of roles $\mathcal{R}$.

The set of behavioural actions (say Λ) that are performed by individuals assuming a particular role is defined by the set $\mathcal{A} \times \mathcal{I} \times \mathcal{R}$. A typical element of this set is indicated by α or by $\langle a, i, r\rangle$ where a is an action (element of $\mathcal{A}$), i an individual (element of $\mathcal{I}$) and r is a role (element of $\mathcal{R}$). We define projection functions act, $indiv$ and $role$ which identify the action, the individual and the role respectively. That is, $act(\langle a, i, r\rangle) = a$, $indiv(\langle a, i, r\rangle) = i$ and $role(\langle a, i, r\rangle) = r$.

The dynamic access control system uses the set of behavovioral actions of the form $\langle a, i, r\rangle$, $\langle i, +r\rangle$ or $\langle i, -r\rangle$. The access control uses actions belonging to Λ to observe the evolution of behaviour. The access control process can keep track of the behaviour and change the permission accordingly. For instance, if

an individual has accessed a data item, the access control process can withdraw access to other data items so that the privacy requirements are met.

The action $\langle i, +r \rangle$ indicates that user i can assume role r while the action $\langle i, -r \rangle$ indicates that user i can no longer assume role r. This will be extended with actions related to the semantics of purposes.

To capture the semantics of purposes, we assume P to be a set of atomic predicates (where a typical element is denoted by p). That is, each element of P represents a specific purpose. We use subsets of P to mark behaviours as a particular behaviour could correspond to many purposes.

We use finite state automata to describe the possible behaviours and access control actions. A behaviour automaton (denoted by $\mathcal{A}_B$) is of the form $(Q_B, \Lambda, \longrightarrow_B, q_{B_0})$, while an access control (denoted by $\mathcal{A}_C$) automaton is of the form $(Q_C, \Lambda_C, \longrightarrow_C, q_{C_0})$.

Here Q_B and Q_C are the sets of states, Λ and Λ_C the alphabets (or transition labels), $\longrightarrow_B$ and $\longrightarrow_C$ the transition relations and q_{B_0} and q_{C_0} the initial states of the respective automata. We do not have any notion of accepting states as behaviours are *valid*. For the sake of simplicity, we will assume that the automata are deterministic and hence the transition relations are functions.

Given a behaviour automaton, a purpose map is a function from the transition function to a subset of P. We let $\mathcal{M}_P = \{f \mid f :\longrightarrow_B\rightarrow \mathcal{P}(P)\}$ be the set of all possible purpose maps. Functions in $\mathcal{M}_P$ mark each transition in the behavioural automaton with a set of purposes.

Formally the labels of the control automaton are drawn from the set (which was denoted by Λ_C) $\Lambda \cup (\mathcal{I} \times \{+, -\} \times \mathcal{R}) \cup \mathcal{P}(P)$. That is, it can observe actions of the behaviour automaton, can change role permissions and allow or deny purpose related action. We will use β to indicate a typical element of this set.

To define the semantics of how the access control process influences or controls the exhibited behaviour, we need to keep track of the roles that can be assumed by the individuals. That is, we need to track the potential role assignments that are currently permitted. This set of possibilities is denoted by the set $\mathcal{S}$ which is the set of all functions from individuals to a subset of roles (i.e., $\mathcal{I} \rightarrow \mathcal{P}(\mathcal{R})$). We use ρ to represent a typical element of $\mathcal{S}$.

Given a specific role assignment, we define if a behavovioral action α is permitted only if the individual performing the action can assume the required rule. The formal definition is given below.

Definition 1. *The predicate permit(ρ, α) is true if and only if role$(\alpha) \in \rho($indiv$(\alpha))$ is true.*

We define the ready set of the access control automaton in a given state as the set of actions it can potentially exhibit at the state. The standard definition is given below.

Definition 2. *For a state q_c belonging to Q_c, we define ready(q_c) as follows.*

ready$(q_c) = \{\beta \mid$ exists q'_c such that $q_c \xrightarrow{\beta}_C q'_c\}$

In order to ensure that the access control system can indeed control the behaviour, we introduce a notion of stability. Essentially we want a system to evolve only after the access control process has finished making all the access control changes.

Definition 3. *An access control automaton is stable in state q_c, written as $stable(q_c)$, if and only if $ready(q_c) \subseteq \Lambda$.*

An access control automaton is stable when it can only observe behaviour actions and cannot exhibit any action that can change the role assignment.

This implies that a state that has both observable and access control transitions is not stable.

To define the semantics of the joint behaviour of the behavioural and access control automata, we define the set of possible states of the overall computation.

Definition 4. *The set of possible system states is the $\mathcal{S} \times Q_B \times Q_C$.*

A particular state of the computation is represented by a triple denoted by $\lceil \rho, q_B, q_C \rceil$ where $\rho \in \mathcal{S}$, $q_B \in Q_B$ and $q_c \in Q_C$

The transition relation of the automaton obtained by composing the behaviour and access control automata indicated by $\|$ is defined as follows. For this, we assume a specific purpose map m.

Definition 5. *$\mathcal{A}_B \parallel \mathcal{A}_C = (\mathcal{S} \times Q_B \times Q_C, \Lambda_C, (f_\emptyset, q_0^B, q_0^C), \longrightarrow)$ where $\longrightarrow$ is defined by the following rules.*

1. $\lceil \rho, q_b, q_c \rceil \xrightarrow{\alpha} \lceil \rho, q_b', q_c' \rceil$ *if* $q_b \xrightarrow{\alpha} q_b'$, $q_c \xrightarrow{\alpha} q_c'$ *provided* $stable(q_c)$, $permit(\rho, \alpha)$ *and* $m(q_b \xrightarrow{\alpha} q_b') = \emptyset$.

2. $\lceil \rho, q_b, q_c \rceil \xrightarrow{\alpha} \lceil \rho, q_b', q_c \rceil$ *if* $q_b \xrightarrow{\alpha} q_b'$ *provided* $stable(q_c)$, $\alpha \notin ready(q_c)$, $permit(\rho, \alpha)$ *and* $m(q_b \xrightarrow{\alpha} q_b') = \emptyset$

3. $\lceil \rho, q_b, q_c \rceil \xrightarrow{\epsilon} \lceil \rho', q_b, q_c' \rceil$ *if* $q_c \xrightarrow{i,+r} q_c'$ *where* $\rho'(j) = \rho(j)$ *if* $i \neq j$ *and* $\rho'(i) = \rho(i) \cup \{r\}$ *otherwise.*

4. $\lceil \rho, q_b, q_c \rceil \xrightarrow{\epsilon} \lceil \rho', q_b, q_c' \rceil$ *if* $q_c \xrightarrow{i,-r} q_c'$ *where* $\rho'(j) = \rho(j)$ *if* $i \neq j$ *and* $\rho'(i) = \rho(i) \setminus \{r\}$ *otherwise.*

5. $\lceil \rho, q_b, q_c \rceil \xrightarrow{\alpha} \lceil \rho, q_b', q_c' \rceil$ *if* $q_b \xrightarrow{\alpha} q_b'$, $q_c \xrightarrow{ps} q_c'$ *provided* $stable(q_c)$ *and* $m(q_b \xrightarrow{\alpha} q_b') \subset ps$.

The first two rules specify permitted behaviour. This requires the action to be permitted in the current state. Furthermore, the access control automaton must be in a stable state and the transition has no specific purpose. Note, that if an action has the right permissions, it cannot be prevented by the access control automaton. That is, there is no need for the behaviour automaton to synchronise on all common actions. The third and fourth rules describe the access control

automaton changing the current permissions. Thus, the behaviour automaton does not change its local state. The last rule enforces the required semantics of purpose. If the transition of the behaviour automaton has a purpose related marking, it is permitted only if the controller allows all the purposes present in the marking.

We write $\lceil \rho, q_b, q_c \rceil \overset{\alpha}{\Longrightarrow} \lceil \rho', q_b', q_c' \rceil$ if there is a sequence of transitons $\lceil \rho, q_b, q_c \rceil \xrightarrow{\epsilon} \lceil \rho^1, q_b^1, q_c^1 \rceil \xrightarrow{\epsilon} \cdots \lceil \rho^k, q_b^k, q_c^k \rceil \xrightarrow{\alpha} \lceil \rho^{k+1}, q_b^{k+1}, q_c^{k+1} \rceil \xrightarrow{\epsilon} \cdots \lceil \rho', q_b', q_c' \rceil$. That is, there is a sequence of "internal" moves (indicated by the ϵ transitions) around a transition exhibiting α. We write this as the triple $(\sigma, \alpha, \sigma')$ where σ is $\lceil \rho, q_b, q_c \rceil$ and σ' is $\lceil \rho', q_b', q_c' \rceil$.

Before we present a few simple examples, we make some observations about the structure of the access control automaton.

At any state if the automaton has a transition of the form $\langle a, i, r \rangle$ and $\langle j, +r \rangle$ or $\langle j, -r \rangle$, the transition with the label $\langle a, i, r \rangle$ can be removed without affecting the overall semantics. This is because of the definition of stability; the transition with the label $\langle a, i, r \rangle$ will never be taken. Similarly, if there is no transition of the form $\langle i, +r \rangle$ from the initial state of the access control automaton, the joint behaviour will not exhibit any action.

For the behaviour automaton, any state that has transitions of the form $\langle a, i, r \rangle$ and $\langle b, i, r \rangle$ can exhibit neither or both actions unless there is a purpose that distinguishes the two transitions.

2.2 Privacy Requirements

We use linear time temporal logic (LTL) to encode the requirements, including privacy, on the behaviour of the composed system. We define two types of atomic predicates. The first is $occurs(\langle a, i, r \rangle)$ which is true at a given state if there is a transition with the label $\langle a, i, r \rangle$. We also define abbreviations where we can leave one of the fields blank. This implicitly implies universal quantification. For instance, $occurs(\langle a, -, r \rangle)$ is an abbreviation for $\forall i \in \mathcal{I} : occurs(\langle a, i, r \rangle)$.

The second is $occurs(p)$ where p is a purpose and is true if there is a transition marked with a set that contains p. As usual we take runs of the composed automata to define satisfaction. More precisely, $(\sigma_0, \alpha_0, \sigma_1), (\sigma_1, \alpha_1, \sigma_2), \cdots, i \models occurs(\alpha)$ iff $\alpha_i = \alpha$. Similarly, $(\sigma_0, \alpha_0, \sigma_1), (\sigma_1, \alpha_1, \sigma_2), \cdots, i \models occurs(p)$ iff one of the transitions in $(\sigma_i, \alpha_i, \sigma_{i+1})$ has a marking that contains p.

To express LTL properties we use standard logical and LTL operators, such as $\vee$, $\wedge$, $\neg$, $\rightarrow$(for implies), $\mathcal{U}$, $\Box$ and $\Diamond$.

3 Examples

In this section we present some simple examples that illustrate our approach.

The first example is of a user, say Alice, who writes a blog and also applies for the job. The interviewer (which is a role) is allowed to read the job application and once they have read the application they can not read the blog.

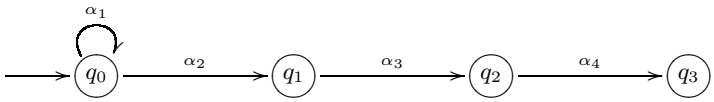
To model the behaviour we use the following abbreviations:

$\alpha_1 = \langle blogWrite, Alice, user\rangle$,
$\alpha_2 = \langle apply, Alice, user\rangle$,
$\alpha_3 = \langle readApplication, Bob, interviewer\rangle$,
$\alpha_4 = \langle readBlog, Bob, interviewer\rangle$,

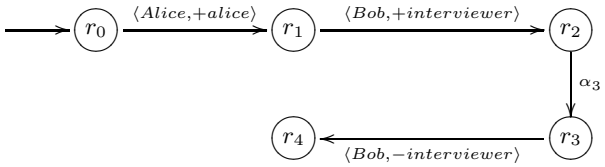
This requirement can be written as

$occurs(\langle readApplication, -, interviewer\rangle) \rightarrow \Box\neg(occurs(\langle readBlog, -, interviewer\rangle)$.

If the behaviour automaton had the following structure,



the following automaton can enforce the above requirement:



This is because after Bob has read Alice's application (α_3), he cannot assume the role of an interviewer. Note, that the stability requirements on the access control automaton means that α_4 is not enabled until the transition from r_3 to r_4 is executed. But once this transition is executed, α_4 cannot be exhibited as the *permit* predicate will evaluate to false.

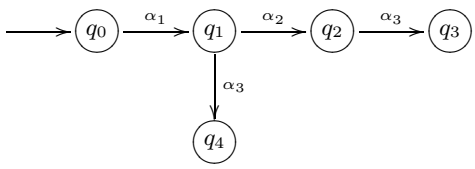
The second example is when Alice generates some data item and Bob can access it only after it is made anonymous. To model this we let:

α_1 be $\langle dataWrite, Alice, generator\rangle$,
α_2 be $\langle dataAnonimise, Alice, generator\rangle$,
α_3 be $\langle dataAccess, Bob, accessor\rangle$.

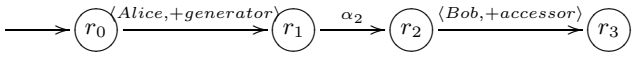
The privacy requirement is captured by the formula

$\neg\ occurs(\alpha_3)\ \mathcal{U}\ occurs(\alpha_2)$

The behaviour can be represented by



Then, the enforcement automaton is as follows.



In this case the access control automaton gives Bob the permission to assume the role of *accessor* only after observing α_2. Hence, the behaviour automaton cannot perform exhibit α_3 in state q_1. This is very similar to classical discrete event control systems where the controller can observe certain actions before enabling other actions.

Our final example is from [8].

Sometimes a patient needs to be transferred to another unit. This is normally permitted unless the patient opts out. Also a patient's treatment can be used for training purpose. This, however, requires explicit permission from the patient.

For the sake of simplicity we assume there is only one patient Pat who can assume the patient's role (pat) and one doctor Doc who can assume the doctor's role doc.

The abbreviation α_1 stands for $\langle optOut, Pat, pat \rangle$ which indicates that the patient wants to opt out of the transfer scheme; while the abbreviation α_2 denotes $\langle signPerm, Pat, pat \rangle$ which indicates that the patient is happy for the treatment information can be used for training.

The actions α_3: $\langle diagnosis, Doc, doc \rangle$ and α_4: $\langle treat, Doc, doc \rangle$ are part of the normal medical process.

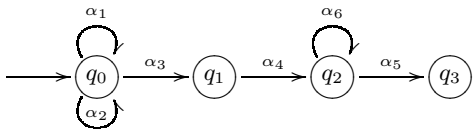
The action α_5: $\langle move, Doc, doc \rangle$ indicates the patient being transferred while the action α_6: $\langle useTrain, Doc, doc \rangle$ indicates the treatment being used in the training process.

We use the predicates $forTraining$ and $transfer$ as the set of purposes.

The privacy requirements are:

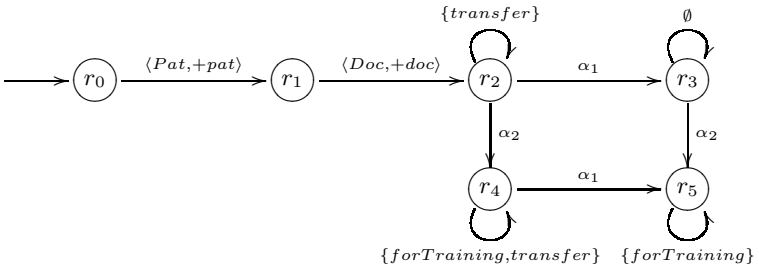
$\neg occurs(\alpha_6)\ \mathcal{U}\ occurs(forTraining) \vee \Box(\neg occurs(forTraining))$, and
$\Box(occurs(\alpha_1) \rightarrow \Box(\neg occurs(transfer)))$

Consider the following behaviour.



Let the transition $q_2 \xrightarrow{\alpha_6} q_2$ be marked with the purpose $forTraining$ and the transition $q_2 \xrightarrow{\alpha_5} q_3$ be marked with the purpose $transfer$.

Consider the following access control automaton.



The joint behaviour ensures that whenever the patient selects to opt out (indicated by the action α_1), the access control removes the option of the $transfer$ purpose (in this case the ability to exhibit α_5). When the patients gives permission (indicated by the action α_2), the action α_6 can be exhibited. Note, that all these actions are executed by the medical staff (indicated by role *doc*) and hence dynamic RBAC by itself cannot enforce such requirements.

4 Prototype Implementation

We now describe the proof-of-concept implementation of our approach.

The prototype implementation consists of two parts: the front-end (a graphical user interface (Figure 1)) and the back-end (a code generator).

The graphical interface allows a user to specify atomic elements of a system, which includes individuals, access control actions, roles, predicates and states. Using these elements one can construct two labelled transition systems (LTS) that describe behavioural and enforcement automata. Additionally, a user is provided with the interface for specification of properties in LTL.

LTS specified via the front-end are used to generate a specification in the SAL [9] language. The generated specification consists of two modules that represent behaviour and enforcement automata and SAL properties, which, depending on the user's intent, should or should not hold in a system with enforced privacy requirements.

We now explain how we generate SAL specifications. At the SAL level we remove the notion of users, roles and access control actions, replacing behavioural actions with predicates. Actions observed by enforcement automaton (i.e. $\langle a, i, r\rangle$, $a \in \mathcal{A}$, $i \in \mathcal{I}$, $r \in \mathcal{R}$) are mapped to boolean variables, which indicate whether a particular behavioural action was observed (*true*) or not (*false*). Similarly, the set of roles, permitted or forbidden for individuals, is mapped to the set of boolean variables, such that a user can assume a particular role, if the respective variable is set to *true* and can not otherwise. Finally, a label in the behavioural LTS may be associated with a set of predicates which represent purposes. Values of purposes are specified by the user.

Generated modules are composed into an asynchronous system synchronised as follows:

- A transition in an enforcement LTS with a label of the form $\langle i, \oplus, r \rangle$, $i \in \mathcal{A}$, $\oplus \in \{+, -\}$, $r \in \mathcal{R}$ is executed unconditionally and sets a boolean variable (say p_1) that allows or forbids an individual i to assume some role r to *true* or *false* respectively.
- A transition in a behaviour LTS with a label of the form $\langle a, i, r \rangle$, $a \in \mathcal{A}$, $i \in \mathcal{I}$, $r \in \mathcal{R}$ is executed if and only if permitted by p_1 (i.e., the value of p_1 is *true* – an individual i can assume role r) set by enforcement automaton. That is, a given user can perform an action assuming a particular role only if that is permitted by privacy requirements. This, in turn, sets to *true* a boolean variable (say a_1), which indicates that some action a, performed by the individual i, assuming the role r was observed via the enforcement LTS.
- A transition in an enforcement LTS with a label of the form $\langle a, i, r \rangle$, $a \in \mathcal{A}$, $i \in \mathcal{I}$, $r \in \mathcal{R}$, is executed if and only if permitted by a_1 (set by the behaviour LTS). This indicates that some action action a, performed by the individual i, assuming the role r was observed by the behaviour LTS.

```
TRANSITION [
    behaviour_state = q0 AND GeneratorAlice = true
        -->
    behaviour_state' = q1; a1' = true;
    []
    behaviour_state = q1 AND GeneratorAlice = true
        -->
    behaviour_state' = q2; a2' = true;
    []
    behaviour_state = q1 AND AccessorBob = true
        -->
    behaviour_state' = q4; a3 = true;
    []
    behaviour_state = q2 AND AccessorBob = true
        -->
    behaviour_state' = q3; a3 = true;
]
```

Listing 1. Example 2. Behaviour LTS

```
TRANSITION [
    enforcement_state  = r0
        -->
    enforcement_state' = r1;
    []
    enforcement_state  = r1
        -->
    enforcement_state' = r2; GeneratorAlice' = true;
    []
    enforcement_state  = r2 AND a2 = true
        -->
    enforcement_state' = r3;
    []
    enforcement_state  = r3
        -->
    enforcement_state' = r3; AccessorBob' = true;
]
```

Listing 2. Example 2. Enforcement LTS

Code listings 1 and 2 depict SAL representation of enforcement and behaviour LTS based on example 2. Variables `enforcement_state` and `behavior_state` represent enforcement and behavioral automata states, booleans `AccessorBob` and `GeneratorAlice` forbid or allow individuals to assume roles and boolean variables `a1`, `a2`, `a3` represent observed actions α_1, α_2 and α_3 respectively. States `r0 ... r3` of the enforcement transition system and `q0 ... q4` of the behaviour LTS refer to states $r_0, ..., r_3$ and $q_0, ..., q_4$ of the enforcement and behaviour automata in example 2.

Note how enforcement automaton prevents privacy violation (i.e., a transition from q_1 to q_4). The transition is executed only if Bob can assume the role of accessor (i.e., `AccessorBob` is *true*), which is set to *true* only when action α_2 is observed (i.e., `a2` is *true*), which is only possible if data is made anonymous. That is, transition $q_1 \rightarrow q_4$ is eliminated by the enforcement LTS, which prevents a privacy violation.

Finally, the generated SAL specification can be checked with `sal-smc` (SAL symbolic model checker) using LTL properties specified by the user via front-end. For example, one can check whether privacy requirements are indeed enforced or whether the enforcement of privacy requirements does not impede system's behaviour, i.e., a particular state in the behaviour transition system is reached or a particular action is executed. For instance, in the above example, one can specify a property `G(not q4)` (state q_4, which constitutes a privacy violation, is never reached) to verify that generated system does prevent the violation of privacy.

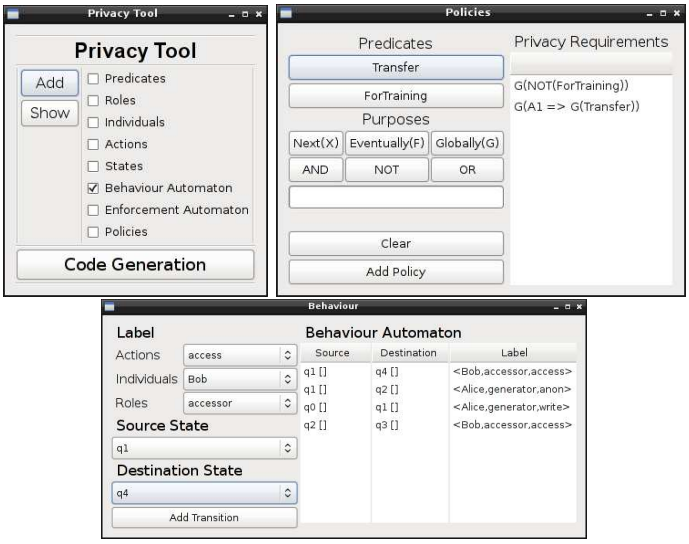


Fig. 1. User Interface of Prototype Implementation

5 Related Work

Our semantic framework is based on parallel composition of finite state automata. Our composition operator is derived from the classical controller [10] for discrete event systems and synchronisation on common actions [11].

The precise semantics of the different uses of purposes in the privacy policies are not clear. The data-purpose algebra [12] shows how data can be used at each stage in the computation. They use a set of atomic values to indicate purpose. These atomic values are associated with data items indicating if the data can be used for a specific purpose. The semantics in [13] is to support automatic auditing. It is also based on Markov decision processes. Conditional purpose using a hierarchical structure and compliance is presented in [5]. The meaning of purpose via an action is presented in [6]. Semantics of intention [14] provides another look at purpose. Johnson et al. [15] present the concepts of template author, policy authors and policy implementers. But it is more about managing privacy policies rather than semantics of the policies themselves.

RBAC [16] and its extensions [17] are very common forms of access control. They can be used to specify who has access to data and also what role they need to assume. One can verify if an implementation technique actually satisfies the policies specified in RBAC. The link between access control and workflow [18] is used to verify designs. The formalism is based on Petri nets. Our access control automaton describes a much simpler semantics. However, our requirements are also limited.

There is also need to model dynamic behaviour. Denotic logic [19] and modal logic [6] have been used to give a semantics to purpose. [3] also develop a notion of privacy where portions of data can be protected.

6 Conclusions

In this paper we have described a formal approach to determining whether access control policies implement privacy requirements given a system's behaviour. This is achieved by extending dynamic role-based access control mechanism with monitoring capability. We represent a specific system using two automata, such that first, behaviour automaton, represents behaviour (e.g. gathering and using the gathered data) and second, controller automaton, captures privacy requirements of the system (including access control). Enforcement of privacy requirements is achieved via a synchronised composition of the two, such that the controller grants access permissions, observes actions exhibited by the behaviour automaton and prevents actions which may violate privacy. In this paper we show how access control may fail to detect privacy violations and demonstrate the applicability of our approach using various examples. We have implemented our approach in a prototype tool, which provides a simple interface for specification of the system's behaviour and privacy requirements and can automatically generate a specification in the SAL language. One can then model check the generated specification against an arbitrary set LTL properties using SAL symbolic model checker.

Acknowledgements

The first author was affiliated with Bond University where most of this work was done. He is currently affiliated with Oracle Labs. The second author was supported by a VC grant from Bond University.

References

1. Ayres, L.T., Curtin, C.M., Ng, T.A.: Standardizing breach incident reporting: Introduction of a key for hierarchical classification. In: Proceedings of The 5th IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, IEEE Computer Society (May 2010) 79–83
2. Cate, F.H.: The limits of notice and choice. IEEE Security & Privacy **8**(2) (2010) 59–62
3. Farnan, N.L., Lee, A.J., Chrysanthis, P.K., Yu, T.: Don't reveal my intension: Protecting user privacy using declarative preferences during distributed query processing. In: Proceedings of the 16th European Symposium on Research in Computer Security. Volume 6879 of Lecture Notes in Computer Science., Springer (September 2011) 628–647
4. Antón, A.I., Earp, J.B., Carter, R.A.: Precluding incongruous behavior by aligning software requirements with security and privacy policies. Information & Software Technology **45**(14) (2003) 967–977
5. Kabir, M.E., Wang, H., Bertino, E.: A conditional purpose-based access control model with dynamic roles. Expert Systems with Applications **38**(3) (2011) 1482–1489
6. Jafari, M., Fong, P.W.L., Safavi-Naini, R., Barker, K., Sheppard, N.P.: Towards defining semantic foundations for purpose-based privacy policies. In: Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, ACM (February 2011) 213–224
7. Ferraiolo, D.F., Kuhn, D.R.: Role-based access controls. In: Proceedings of the National Computer Security Conference, U.S. Department of Commerce, Gaithersburg, Md. 20899 USA, NSA/NIST, Elsevier Advanced Technology Publications (October 1992) 554 – 563
8. LeBlanc, M.: Physiotherapists' privacy requirements in Ontario. Technical report, College of Physiotherapists of Ontario (2004)
9. de Moura, L., Owre, S., Rueß, H., Rushby, J., Shankar, N., Sorea, M., Tiwari, A.: SAL 2. In Alur, R., Peled, D., eds.: Computer-Aided Verification, CAV 2004. Volume 3114 of Lecture Notes in Computer Science., Boston, MA, Springer-Verlag (July 2004) 496–500
10. Wonham, W., Ramadge, P.: Modular supervisory control of discrete-event systems. Mathematics of Control, Signals and Systems **1** (1988) 13–30
11. Zielonka, W.: Notes on finite asynchronous automata. Theoretical Informatics and Applications **21**(2) (1987) 99–135
12. Hanson, C., Berners-Lee, T., Kagal, L., Sussman, G.J., Weitzner, D.J.: Data-purpose algebra: Modeling data usage policies. In: Proceedings of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society (June 2007) 173–177
13. Tschantz, M.C., Datta, A., Wing, J.M.: On the semantics of purpose requirements in privacy policies. The Computing Research Repository **abs/1102.4326** (2011)

14. Kagal, L., Pato, J.: Preserving privacy based on semantic policy tools. IEEE Security & Privacy **8**(4) (2010) 25–30
15. Johnson, M., Karat, J., Karat, C., Grueneberg, K.: Optimizing a policy authoring framework for security and privacy policies. In: Proceedings of the 6th Symposium on Usable Privacy and Security. Volume 485 of ACM International Conference Proceeding Series., ACM (July 2010)
16. Jha, S., Li, N., Tripunitara, M.V., Wang, Q., Winsborough, W.H.: Towards formal verification of role-based access control policies. IEEE Transactions on Dependable and Secure Computing **5**(4) (2008) 242–255
17. Fong, P.W.L., Siahaan, I.: Relationship-based access control policies and their policy languages. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, ACM (June 2011) 51–60
18. Barletta, M., Calvi, A., Ranise, S., Viganò, L., Zanetti, L.: Workflow and access control reloaded: a declarative specification framework for the automated analysis of web services. Scalable Computing: Practice and Experience **12**(1) (2011)
19. Piolle, G., Demazeau, Y.: Representing privacy regulations with deontico-temporal operators. Web Intelligence and Agent Systems **9**(3) (2011) 209–226