International Journal of the Physical Sciences Vol. 5 (6), pp. 865-875, June, 2010
Available online at http://www.academicjournals.org/IJPS
ISSN 1992 - 1950 © 2010 Academic Journals

Full Length Research Paper

Information flow analysis of UCON

Mohammad Nauman[1], Tamleek Ali[2], Muhammad Khurram Khan[3]*, Khaled Alghathbar[3,4]

[1] Department of Computer Science, University of Peshawar, Pakistan
[2] Institute of Management Sciences, Peshawar, Pakistan
[3] Center of Excellence in Information Assurance (CoEIA), King Saud University, Kingdom of Saudi Arabia
[4] Information Systems Department, College of Computer and Information Sciences, King Saud University, Kingdom of Saudi Arabia

Accepted 18 May, 2010

The UCON model extends traditional access control models through continuity of access decision and mutability of subject and object attributes. Due to these two features, the flow of information in UCON becomes considerably different from traditional access control models. A thorough analysis of this information flow is beneficial in any scenario where UCON is used. In this paper, we analyze information flow in UCON. In particular, we identify the rules for information flow, and determine how these rules can be applied to particular policy types of UCON. We specify information flow in core UCON models using temporal logic of actions and provide an algorithm for the automation of dynamic information flow analysis in UCON.

Key words: UCON, traditional access control models, temporal logic of actions.

INTRODUCTION

UCON (Park and Sandhu, 2002) is a highly expressive model covering continuous usage control of resources by subjects. It introduces two novel features in the form of continuity of access decisions -- which allows coupling of access decisions with usage of a resource -- and mutability of attributes -- which allows the system to change attributes of subjects or objects before, during and after usage. These two novelties of UCON make it different from the traditional access control models (Sandhu, 1993; Sandhu, 1996). They also lead to considerable differences in the way information can flow between objects in UCON.

Role-based access control is a popular model of access control. It has been seen that information flow in an RBAC policy can become complicated and can make it difficult to understand how information may flow due to a particular RBAC design (Osborn, 2002). In UCON, this issue becomes considerably amplified due to the extensions added by the two novelties of UCON.

In this paper, we analyze the information flow in a UCON system. We identify how the two novel features of UCON affect information flow. We define the rules for analysis of information flow in UCON and identify how these rules can be applied for the analysis of information flow in different UCON core models. The rest of the paper is organized as follows:

In Section 2, we briefly describe the UCON model. Information flow analysis in a traditional RBAC model is described in Section 2.3. Information flow in UCON is thoroughly analyzed in Section 3. We specify the rules of information flow in Section 3.1 and specify the information flow in different core models of UCON in Section 3.2. In Section 3.3, we provide two algorithms for the analysis of information flow in a UCON system. Implementation is outlined in Section 4. Finally, we conclude our contribution in Section 5.

BACKGROUND

Access control

Discretionary Access Control (DAC) (Lampson, 1974) and Mandatory Access Control (MAC) (Bell and LaPadula, 1973) mechanisms are some of the most widely known access control models but are severely limited in either their usability or expressive power. Role-based Access Control (RBAC) proposed by Sandhu (1996) is the precursor of many fine-grained and expres-

*Corresponding author. E-mail: mkhurram@ksu.edu.sa.

sive access control models. The simplicity, standardization and support of many organizational needs make RBAC a suitable candidate for many real-world organizations. In the basic RBAC model, users are assigned to roles and each role is assigned permissions based on its job requirement. Users receive permissions of all the roles to which they belong. Despite this simplicity, one drawback of RBAC is that it requires a central organization-wide authority to define the user-to-role and permission-to-role assignments.

Another similar access control model is the Attribute-based Access Control} (ABAC) (Johnston et al., 1998; Wang et al., 2004) model that defines policies based on subject and object attributes. ABAC is more robust than RBAC since attributes can be spread across distributed authorities and do not require a central management authority. Several extensions (Joshi et al., 2005) of both RBAC and ABAC have been proposed in the literature but a detailed discussion of these remains outside the scope of this paper. Building on the strengths of RBAC and ABAC and the requirements of today's distributed environments, a new paradigm of usage control emerged. Below, we provide details of this new paradigm of distributed control over access of information.

UCON

In traditional access control models, decision to allow or deny access to an object or resource is made only once -- when access is requested. Once a resource is released to a requestor, there is no way of controlling the usage of the object. Usage CONtrol (UCON) is an extremely flexible model proposed by Park and Sandhu (2002) for controlling access to a resource after it is released to a client. Usage control differs from access control in that it is concerned with continuous usage of an object, mutability in subject or object attributes as a result of this usage and the changes in access decisions resulting from this mutability of attributes. Thus, mutability of attributes and continuity of access decision are the two features distinguishing it from traditional access control models. Note that while changes to subject and object attributes are possible in traditional access control scenarios, they are not part of any access control model.

Usage control is a superset of a multitude of models covering access control, digital rights management and privacy. Usage control has also been shown to be able to represent the Chinese Wall Policy (Brewer and Nash, 1989) and Mandatory Access Control (Sandhu, 2002) mechanisms (Brewer and Nash, 1989).

Zhang et al. (2005) have presented a formal specification of UCON in which a UCON policy consists primarily of:

(1) a set of system states corresponding to a subject (s), object (o) and right (r),
(2) predicates involving system attributes only (called conditions), those involving object and subject attributes (authorizations) and directives to a subject for performing some action during or before access (obligations),
(3) updates to subject or object attributes before access (preupdate), during access (onupdate) or after completing access (postupdate).

Policy statements in UCON are categorized as authorizations, obligations and conditions. Authorizations refer to predicates which are based on subject or object attributes only. Obligations are directives to a subject to perform additional actions before or during an access. Predicates exclusively based on environment attributes such as system time, device type etc., are categorized as conditions.

In UCON, a usage session is identified by a 3-tuple, (subject, object, right). Associated with each usage session (s, o, r) is a set of system states. Depending on the policy type, these can include initial (starting state), requesting (s has tried access to o), accessing (s is exercising right r on o), denied (s was denied access to o), revoked (the system revoked access from s) and end (access was ended by s).
The transitions between these states can be seen in Figure 1 (Park and Sandhu, 2002).

Associated with each state is a state transition action and possibly attribute update actions. State accessing is reached when the transition action permitaccess is performed by the system. Moreover, because accessing is the state in which some right is being exercised by the subject, changes to attributes are coupled with usage thus attribute updates of type onupdate are associated with it. Similarly, when access is ended by the subject, endaccess transition takes place; the usage session moves from state accessing to end and any attributes which need to be updated after access finishes are postupdate. Note that only subject or object attributes mutability is allowed in UCON.

A UCON policy type is defined in terms of its predicate types and decision and update timings. Decision timings are an important aspect of the UCON model. The two decision timings are pre-decision to allow access to a resource is made before granting access and on continued decision to allow access to a resource is coupled with the usage of the resource.

Predicates types are authorizations (A), obligations, (B) and conditions (C). Attribute update timings can be 0 (no updates in the policy statement), 1 (preupdates only), 2 (onupdates only), 3 (postupdates only) or a combination of 1, 2 and 3. For example, a policy in which decision is coupled with usage of the resource, predicates involve authorizations only and attributes are updated before and after access will be of type onA_{13}. The dynamics of state transition actions are specified formally by Zhang et al. (2005) using an extended Temporal Logic of Actions (Lamport, 1994). Temporal operators are used for the

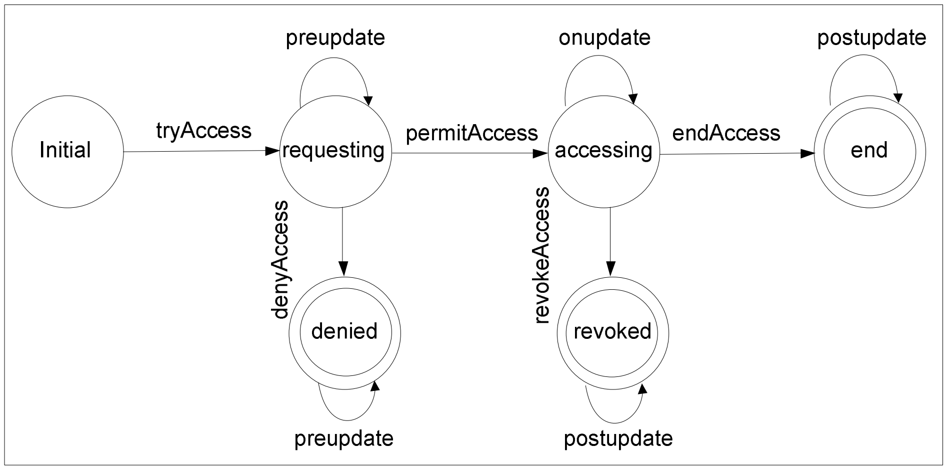

Figure 1. UCON model states.

purpose of defining actions of a UCON system. The temporal operators most relevant to our discussion are the following:

(1) Once (⧫): Returns true if the operand has been true in at least one past state.
(2) Has-always-been (■): Returns true if the operand has been true in all past states.
(3) Eventually (◇): Returns true if the operand is true in at least one future state.
(4) Always (□): Returns true if the operand is true in all future states.

All UCON actions are defined in terms of UCON core models and temporal operators. For example, the permitAccess action in preA$_1$ model is described as:

$$permitAccess \rightarrow ⧫(tryAccess(s,o,r) \wedge (p_1 \wedge \cdots \wedge p_i))$$

$$permitAccess \rightarrow ⧫preupdate(attribute)$$

Consider for example, the following policy (Zhang et al., 2006) associated with a protected file that dictates that a user can only read the file if she is in the reading group of that file. Moreover, after she finishes reading the book, her expense is updated to reflect the rent of the reading session:

$$permitAccess(s,o,read) \rightarrow ⧫(tryAccess(s,o,read)$$
$$\wedge (o.readingGroup \in s.readingGroup))$$

$$endAccess(s,o,read) \rightarrow \Diamond(postupdate(s.expense))$$

$$postupdate(s.expense) : s.expense + o.readingCost$$

In this paper, we use this formalization of UCON core models to specify the possible information flows in a UCON system. Moreover, we build on a previously specified formalization of information flows in RBAC models (Osborn, 2002). Below, we briefly discuss this formalization of information flow in RBAC models.

Information flow analysis of RBAC

Possible information flow in an RBAC system has been analyzed by Osborn (2002). This analysis provides a mapping from a role-graph to an information flow graph. The result is that, given an RBAC policy, user assignments and sessions, the mapping can generate a graph of possible information flows between objects. This analysis is especially relevant to our work because it has defined the underlying rules for information flow in role based access control. These rules of information flow are:

(1) If a role r has privileges (o_1, read) and (o_2, write) and there is at least one user assigned to this role, then information can possibly flow from o_1 to o_2.
(2) If there are two roles r_1 and r_2 with privileges (o_1, read) and (o_2, write) respectively and there exists a user u, such that u is assigned both roles r_1 and r_2, then information can flow between o_1 and o_2.
(3) It should be noted that while this approach of information flow analysis of RBAC can create a "can-flow" graph, it still relies on run-time information regarding user-role assignments and sessions to completely capture all information flows.

Building on this background work, we now identify the differences between RBAC and UCON information flows, define the rules of the flows and specify the information flows in different UCON core models.

Information flow in UCON

Access decision continuity and attribute mutability lead to differences in information flow analysis from RBAC. Below, we identify the differences between RBAC and UCON system which affect information flow in the UCON model. These are:

(1) In RBAC, decision to grant access is made one time only and after that, the object is considered to be usable by the subject in all future states. In UCON, the continuity of access decision leads to a more detailed level of usage specification and thus can lead to different information flows. Revocation and end of access is part of the model and can be used to analyze information flow at a more fine-grained level.
(2) RBAC relies on roles and information flow thus relies on roles and indirectly on subjects. In UCON, there are no explicit roles and analysis of information flow due to subjects can be analyzed directly. This too leads to a fine-grained information flow analysis.
(3) In RBAC, attribute updates are possible but they are not part of the model itself and thus cannot be addressed in the information flow analysis. In UCON, these are a part of the model and thus need to be considered during information flow analysis. Also note that updates are performed by the UCON system and thus can always be performed as specified by the policy. These lead to conditions in which, while a subject may not be able to write information to an object, information may nonetheless flow from the subject's attributes to the object's attributes.
(4) Information flow in traditional access control models can be analyzed statically. Rights are pre-assigned to roles and information flow implications of these permissions do not require run-time attributes of the subjects or objects. In UCON, however, rights are assigned dynamically at the time of request. Information flow analysis in UCON, therefore is more complex due to this dynamic nature of UCON rights assignment.

In a UCON system, there are two sources of information flow -- subjects and the system itself. Corresponding to these two sources, we have identified two rules of information flow in UCON.

Rules

In defining the rules of information flow, we make the following four assumptions:

(1) The only two rights which can cause information to flow are read and write. All other rights can be mapped to either read or write (Osborn, 2002).
Information flow is transitive.
(2) The set of subjects are a subset of the set of objects.
(3) For simplicity, we assume that only two usage sessions are active. If there are more usage sessions, their information flow can be analyzed using the transitivity property of information flow.

We identify the following rules of information flow in UCON. The first rule corresponds to the information flow caused by the subjects of the system. The second rule captures the information flow which results due to the updation of attributes by the system itself.

Rule 1

If there exists a state such that (s, o_1, read) and (s, o_2, write) are both permitted in that state, then information s can cause information to flow between objects o_1 and o_2. We write $o_1 \rightsquigarrow o_2$ to denote that information can flow between o_1 and o_2. Specifically:

$$state(s, o_1, read) = state(s, o_2, write) = accessing \rightarrow o_1 \rightsquigarrow o_2$$

This information flow is due to subjects of UCON. This is similar to the information flow in the traditional RBAC model. The difference is that both rights have to be allowed and exercised in a single state. Due to the stateless nature of RBAC models, this feature is missing from the traditional models.
Note that, due to the “revoked” and “end” states of UCON, information flow due to subjects requires this concurrent access of two objects by a single subject for information to flow between them.
We formulate this in a theorem as follows:

Theorem 1

In UCON, for information flow to be caused by a subject s from an object o_1 directly to another object o_2, the subject s must be able to exercise right `read' on $o_1$ and simultaneously must be able to exercise right `write' on o_2.

Proof sketch: When s reads some information from o_1, the destination of the information can either be another object or the system memory. If the destination is o_2, s must be able to write to o_2 while reading from o_1 -- hence s must be accessing o_1 and o_2 simultaneously. On the other hand, if the destination of information being read from o_1 is the system memory then s can no longer access that data in the memory after revoke or end state is reached. Hence, this information cannot flow from the memory to o_2 after access to o_1 has either ended or has been revoked. Finally, if the destination of the data being read from o_1 is another object o_x in the system, then the problem is reduced to that of $o_1 \rightsquigarrow o_x \wedge o_x \rightsquigarrow o_2$. The same conditions apply for information to flow between o_1 and o_x and from o_x to o_2. This completes our proof of the statement that for information to flow between two objects, they have to be accessible in the same system state to a subject.

Rule 2

If there exist two update operations such that the first updates an attribute of s using an attribute of o_1 and the second updates an attribute of o_2 using the same attribute of s, then we say that information flows from o_1 to o_2. Formally:

$$\begin{aligned}\text{If } update_1(s.attr_x) &= f(o_1.attr_y) \text{ and}\\ update_2(o_2.attr_z) &= f(s.attr_x) \text{ then}\end{aligned}$$

$$update_1(s.attr_x) \wedge \Diamond(update_2(o_2.attr_z) \rightarrow o_1 \rightsquigarrow o_2$$

Where $update_x$ can be either preupdate, onupdate or postupdate. It is necessary in this statement that the update expression $update_1$ contain some attribute of o_1 and that of $update_2$ contain $attr_1$ of s. This statement would ensure that $o_1 \rightsquigarrow$ s and s $\rightsquigarrow o_2$. Due to transitive nature of information flow, $o_1 \rightsquigarrow o_2$. Note that for the sake of simplicity, we only mention attributes of a subject or object and assume that attributes of other subjects/ objects are involved in the update statements. These will be explicitly mentioned only when they are not intuitively clear.

This information flow is due to the UCON system and not an individual subject. This is an important difference from the RBAC model. There is no equivalent of information flow caused solely by the system in RBAC because attribute update (or mutability) is not a part of the RBAC model.

Information flow specification

Using the rules of information flow specified in the previous section, we now specify information flow in core UCON models. We base our specification on the formal specification of UCON given in Zhang et al. (2005). We deviate from this formal specification of UCON in a few respects: In formal specification of UCON in (Zhang et al., 2005), it is assumed that the time line of temporal operators starts from tryaccess of the usage session. In our formalization of information flow analysis, we have assumed that there are two usage sessions running simultaneously. We widen the time line of temporal operators and assume that it starts from the tryaccess of the first usage process and ends at the endaccess or revokeaccess of the last usage session. This is important for the specification of information flow involving multiple usage processes.

We have identified four cases for information flow analysis which completely cover all UCON core models. These are:

Pre models without an update ($preA_0$, $preB_0$ etc)
Pre models with updates ($preA_1$, $preC_3$ etc)
On models without an update (onA_0, $preB_0$ etc)
On models with updates (onA_1, onC_2 etc)

Note that from the point of view of information flow analysis, authorization, obligation and condition models are the same. Therefore, we only analyze the authorization models in this contribution. Also note that in our viewpoint, the update timings have no effect on information flow because the system is not bound by any predicates or systems while making changes to attributes.

Here, we specify information flow in each of the four cases in detail.

Case 1: Pre models without updates

The simplest case of information flow in UCON is that one which involves pre models without updates. These involve only the information flow caused by subjects. Below, we formally specify this case using temporal logic of actions:

$$\begin{aligned}
&o_1 \rightsquigarrow o_2 \iff\\
&\quad \blacklozenge(\quad \blacklozenge(tryaccess(s, o_1, read)\\
&\quad \wedge(p_1 \wedge \ldots \wedge p_i) \wedge \Diamond(tryaccess(s, o_2, write)\\
&\quad \wedge(q_1 \wedge \ldots \wedge q_i) \wedge \Diamond(endaccess(s, o_1, read))))\\
&\qquad \vee \blacklozenge(tryaccess(s, o_2, write)\\
&\qquad \wedge (q_1 \wedge \ldots \wedge q_i)\\
&\qquad \wedge \Diamond(tryaccess(s, o_1, read) \wedge (p_1 \wedge \ldots \wedge p_i)\\
&\qquad \wedge \Diamond(endaccess(s, o_1, read)))))
\end{aligned}$$

This rule depicts that in order for information to flow between o_1 and o_2, they both have to be readable and writable (respectively) in at least one state. Object o_1 will be readable (in state accessing) for s only if there was a tryaccess in a state and the predicates were all true. If eventually, there comes a state such that o_2 becomes writable for s that is, tryaccess was performed and all predicates were true, then both permissions would have been granted to s.

Hence, information can flow from o_1 to o_2 according to Rule 1. However, another necessary condition is that access to o_1 must not end before o_2 becomes writable. If o_1 is made readable for s but access ends before o_2 becomes writable, then information cannot flow between the two objects. This end access is one of the primary differences between information flow in RBAC and UCON. In RBAC, there is no end access or revoke access. Note that since this is information flow in pre models and state revoked is not part of these models, we do not consider revoked in this case.

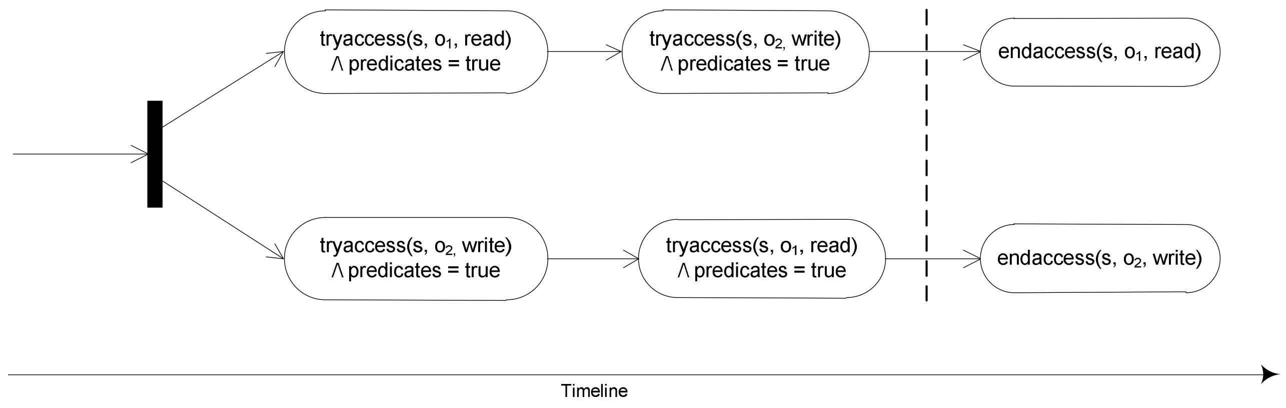
tryaccess(s, o1, read) ∧ predicates = true
tryaccess(s, o2, write) ∧ predicates = true
endaccess(s, o1, read)
tryaccess(s, o2, write) ∧ predicates = true
tryaccess(s, o1, read) ∧ predicates = true
endaccess(s, o2, write)
Timeline

Figure 2. Timeline for pre models without updates.

This sequence of actions is depicted in Figure 2. In the figure, time progresses from left to right. The fork depicts a disjunction. Either choice of the path may lead to situations in which information may flow. The vertical dashed line in the sequence depicts the point on the timeline when information flow may possibly occur. In all subsequent Figures 3-5, we use the same notation.

Case 2: Pre models with updates

These models differ from the first case because updates are involved in them. This leads to the involvement of Rule 2 described in Section 3.1. Hence, the information flow can be specified formally as:

$$
\begin{aligned}
&o_1 \rightsquigarrow o_2 \iff \\
&\quad \blacklozenge(\quad \blacklozenge(tryaccess(s, o_1, read) \wedge (p_1 \wedge \ldots \wedge p_i) \\
&\qquad \wedge \lozenge(tryaccess(s, o_2, write) \wedge (q_1 \wedge \ldots \wedge q_i) \\
&\qquad \wedge \lozenge(endaccess(s, o_1, read)))) \\
&\qquad\quad \vee \blacklozenge(tryaccess(s, o_2, write) \\
&\qquad\qquad \wedge (q_1 \wedge \ldots \wedge q_i) \\
&\qquad\quad \wedge \lozenge(tryaccess(s, o_1, read) \\
&\qquad\qquad \wedge (p_1 \wedge \ldots \wedge p_i)
\end{aligned}
$$

where $update_x \in$ {preupdate onupdate, postupdate}.
The first part of the disjunction is the same as Case I. The second disjunct (last line) specifies that if there's an update to a subject attribute and later, another update to an attribute of o_2, information can possibly flow from o_1 to o_2.

Note that it is assumed here that the update statement of s.attr involves some attribute of o_1 and the update statement of o_2.attr involves s.attr. The updation in the specified manner is a sufficient condition for information flow from o_1 to o_2.

Case III: On models without updates

In considering the information flow of on models, we have discovered a surprising result. This result is summarized in the following specification:

$$
\begin{aligned}
&o_1 \rightsquigarrow o_2 \iff \\
&\quad \blacklozenge(tryaccess(s, o_1, read) \wedge tryaccess(s, o_2, write) \\
&\qquad \wedge \lozenge(endaccess(s, o_1, read) \vee \\
&\qquad\quad revokeaccess(s, o_1, read)))
\end{aligned}
$$

In pure on models, permitaccess is made conditional only with tryaccess and no predicates are checked during the transition from states requesting to accessing. Due to this reason, if there is a state in which a subject s tries access for two objects, o_1 and o_2 for rights read and write respectively, it is permitted access on both accounts. Access might be revoked in the state immediately following this one but according to Rule 1, information might have flown from o_1 to o_2 in the current state. We note however, that pure on models are not likely to be used alone but in conjunction with some pre policies and this is a problem which is unlikely to surface in the real-world implementations of UCON. This finding of ours is, however, beneficial in emphasizing the fact that pure on models should not be used alone lest they be compromised due to unwanted access.

Another important difference between this case and Case I is that revokeaccess is also part of this specification. Since on models couple decision access with usage, it is possible that access might be revoked by the system. In such a case, unless information has flown before ending or revocation of access, it cannot flow later.

Case IV: On models with updates

The last type of models for information flow specification

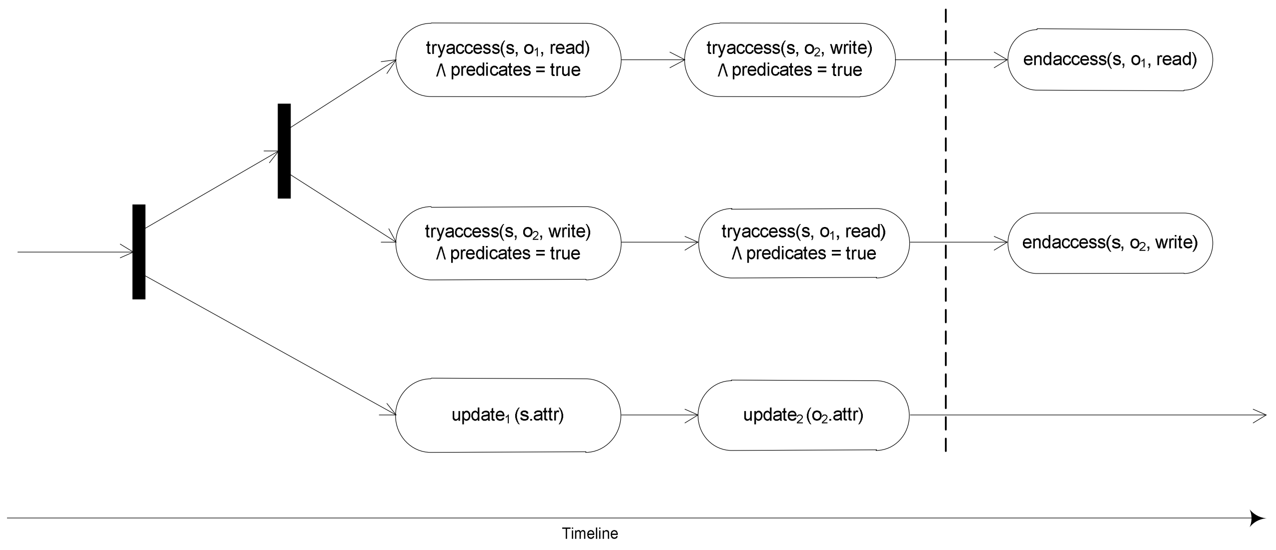
tryaccess(s, o1, read) ∧ predicates = true
tryaccess(s, o2, write) ∧ predicates = true
endaccess(s, o1, read)
tryaccess(s, o2, write) ∧ predicates = true
tryaccess(s, o1, read) ∧ predicates = true
endaccess(s, o2, write)
update1 (s.attr)
update2 (o2.attr)
Timeline

Figure 3. Timeline for pre models with updates.

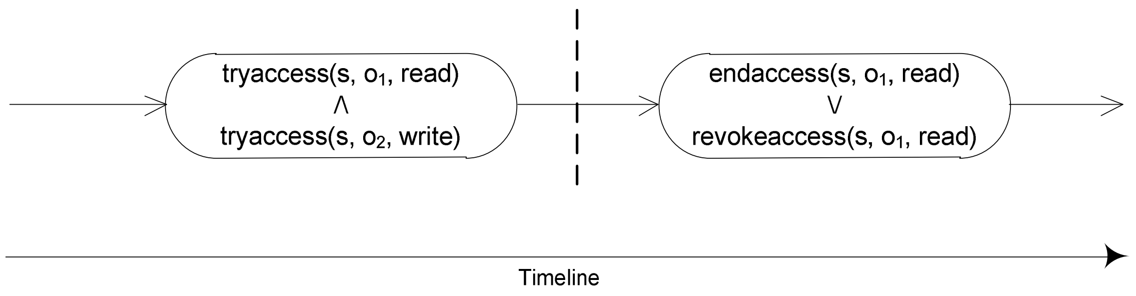
tryaccess(s, o1, read) ∧ tryaccess(s, o2, write)
endaccess(s, o1, read) ∨ revokeaccess(s, o1, read)
Timeline

Figure 4. Timeline for on models without updates.

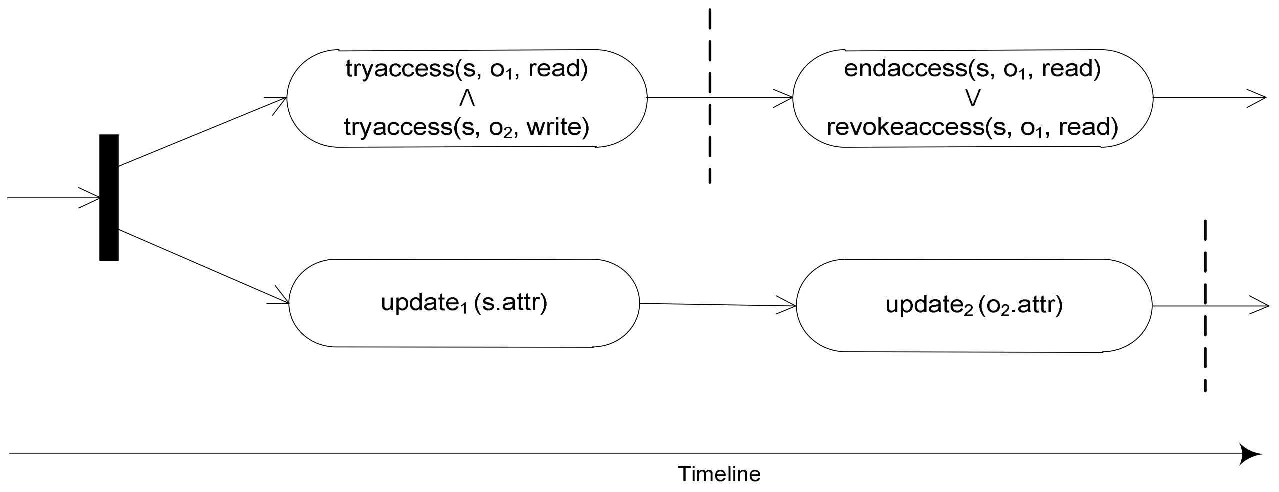
tryaccess(s, o1, read) ∧ tryaccess(s, o2, write)
endaccess(s, o1, read) ∨ revokeaccess(s, o1, read)
update1 (s.attr)
update2 (o2.attr)
Timeline

Figure 5. Timeline for on models with updates.

is concerned with continuous usage of objects and mutability of attributes during usage. We specify this type of models in a similar manner as Case III:

$$o_1 \rightsquigarrow o_2 \iff \blacklozenge(tryaccess(s, o_1, read) \wedge tryaccess(s, o_2, write) \wedge \lozenge(endaccess(s, o_1, read) \vee revokeaccess(s, o_1, read))) \vee \blacklozenge(update_1(s.attr) \wedge \lozenge(update_2(o_2.attr)))$$

Similar to Case II, $update_x$ might be preupdate, onupdate or postupdate. Again, it is assumed that update statement for s.attr is a function of some attribute of o_1 and update statement of o_2 is a function of s.attr. However, note that since the information flow due to subjects (similar to Case III) is conditional only with tryaccess, the second disjunct (flow of information due to the system) becomes insignificant. According to the specification, if the subject tries access for both objects, information might flow between them regardless of whether the system updates some attributes of these two objects.

Algorithms

We provide two algorithms (cf. Algorithm 1 and 2 in Appendix) to automate the process of analyzing information flow in UCON. The first algorithm calculates information flow in pre models with updates. The algorithm takes two UCON policies: one corresponding to permitaccess (s, o_1, read) and the other to permitaccess (s, o_2, write). The output of the algorithm is a boolean value which describes whether, given these two usage policies, information can flow from object o_1 to object o_2 ($o_1 \rightsquigarrow o_2$). The algorithm is coupled with the continuous usage of objects in the UCON system and the calculations are done in each state of the timeline. The algorithm has two parts. The first (lines 8 - 28) captures information flow caused by the subjects. The values of variables allowedread and allowedwrite in any state describe whether subject s is currently allowed to read from o_1 and/or write to o_2. The first if (line 12) checks whether s is trying access to object o_1 and whether the authorization predicates are true. If both conditions hold, s is allowed read access to o_1. Also, if in any condition, read is allowed from o_1 and previously, write was allowed to o_2, then in this state, information can flow from o_1 to o_2 (line 14). Note that if write was allowed for o_2 at some point in the past states but access ended before reading was allowed for o_1 (line 21), then information cannot flow from o_1 to o_2. The second if captures the condition when reading is allowed and later, writing is allowed while read access has not ended (lines 16 - 19).

The second part of the algorithm (lines 22 - 35) captures the information flow due to the system updating attributes of the objects. (Note that this part of the algorithm can be omitted if the model of the policy is `without updates'.) If there is an update to an attribute $s.attr_i$ of subject s such that the attribute update statement is a function of some attribute of o_1 (line 26) then $s.attr_i$ is added to the set attrs of attributes which have received information from o_1. If later, there is an attribute update in some attribute of o_2 such that the attribute update statement contains some attribute of s contained in attrs then information can flow from s to o_2 (line 31). Thus, due to the transitivity property of information flow, $o_1 \rightsquigarrow o_2$.

For the information flow analysis of on models with updates, we provide Algorithm 2. The algorithm is similar to the first one but differs in the way information may flow between subjects. The information flow from o_1 to o_2 requires only that the same subject try access for the two objects in the same state (line 10). In such a case, information may potentially flow regardless of the predicates involved in the policies. The rest of the algorithm is the same as the first one. Note that for on models without updates, the second part of the algorithm may be omitted as with the algorithm for pre models.

Implementation

One of the major reasons due to which adoption and study of the UCON model has been severely limited is the lack of a production-level implementation of the model. Due to the relatively young age of the model, no publicly available implementation exists to date. Therefore, to demonstrate the feasibility of our approach, we created a prototype implementation of the UCON model using the Java language. The reason for this choice was the modular and object-oriented nature of the Java language coupled with an extremely strong tool support. The prototype was designed in modules including:

(1) Attribute Resolver: for retrieval and updation of the subject and object attributes from attribute repositories (currently only local),
(2) Context Manager: for maintaining the state of the usage sessions,
Resource Manager: for secure storage and protected retrieval of resources from physical data stores and
(3) Policy Evaluator: for policy execution and enforcement (including updates).

After the creation of the core UCON engine, we created a frontend application that allowed the rendering of and commenting on PDF documents encapsulating financial records of an organization. To capture information flows, we incorporated the two algorithms presented in Section 3.3 in the Policy Evaluator module of the UCON engine. The end result of the execution of the algorithms is a graph representing the information flows between different objects. The graph can then be verified against the organizational policies.

CONCLUSION

UCON is concerned with the usage of an object after it is released to a client. It adds two novel features -- mutability of attributes and continuity of access decisions -- to traditional access control models. These two features make it much more expressive than access control models. They also lead to considerable differences in the way information may flow between objects. Continuity of access decisions puts certain constraints on, and mutability of attributes opens new conduits for information flow. We have analyzed both these aspects and have specified the rules for information flow in core UCON models.

Information flow analysis of UCON described in this paper is useful for solving several different problems. Our own interest in this analysis is for remote attestation – an important aspect of trusted computing which allows a challenger to verify that a remote platform is in a trusted state. We aim to utilize these information flow rules as a benchmark for remote attestation in order to verify that a remote platform claiming to implement UCON does not allow any illegal information flows. This information flow analysis can be used in the high-level framework (Alam et al., 2008) proposed in an earlier work. This will lead to a more dynamic attestation mechanism as opposed to the static hash based methods (Sailer et al., 2004; Jaeger et al., 2006) currently in use. Previous attempts at dynamic attestation have limited themselves to kernel integrity measurement (Loscocco et al., 2007) and recording system calls made by applications (Gu et al., 2008). We have previously defined a technique (Nauman et al., 2009) for attestation of information flows based on static analysis. The technique presented in this paper supports dynamic analysis of information flow in a UCON system and can thus lead to more accurate remote attestation results.

REFERENCES

Alam M, Zhang X, Nauman M, Ali T, Seifert J (2008). "Model-based behavioral attestation". In Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (Estes Park, CO, USA, June 11 - 13, 2008). SACMAT '08. ACM, New York, NY, 175-184.

Bell D, LaPadula L (1973). "Secure Computer Systems: Mathematical Foundations," Technical Report MTR-2547.

Brewer D, Nash M (1989). "The Chinese Wall Security Policy," in 1989 IEEE Symposium on Security and Privacy, 1989. Proceedings, pp. 206–214.

Gu L, Ding X, Deng RH, Xie B, Mei H (2008). "Remote attestation on program execution". In Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (Alexandria, Virginia, USA, October 31 - 31, 2008). STC '08. ACM, New York, NY, 11-20.

Jaeger T, Sailer R, Shankar U (2006). "PRIMA: policy-reduced integrity measurement architecture". In Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies (Lake Tahoe, California, USA, June 07 - 09, 2006). SACMAT '06. ACM, New York, NY, 19-28.

Johnston W, Mudumbai S, Thomp- son M (1998). "Authorization and attribute certificates for widely distributedaccess control," Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collabora- tive Enterprises, 1998.(WET ICE'98) Pro- ceedings., pp. 340–345.

Joshi J, Bertino E, Latif U, Ghafoor A (2005). "A Generalized Temporal Role-Based Access Control Model," IEEE Transactions on Knowledge and Data Engineering, 17(1): 4–23.

Lamport L (1994). "The temporal logic of actions". ACM Trans. Program. Lang. Syst. 16(3): 872-923.

Lampson B (1974). "Protection," ACM SIGOPS Operating Systems Review, 8(1): 18–24.

Loscocco PA, Wilson PW, Pendergrass JA, McDonell CD (2007). "Linux kernel integrity measurement using contextual inspection". In Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing (Alexandria, Virginia, USA, November 02 - 02, 2007). STC '07. ACM, New York, NY, 21-29.

Nauman M, Alam M, Zhang X, Ali T (2009). "Remote Attestation of Attribute Updates and Information Flows in a UCON System". In Proceedings of the 2nd international Conference on Trusted Computing (Oxford, UK, April 06 - 08, 2009). L. Chen, C. J. Mitchell, and A. Martin, Eds. Lecture Notes In Computer Science, vol. 5471. Springer-Verlag, Berlin, Heidelberg, 63-80.

Osborn SL (2002). "Information Flow Analysis of an RBAC System," in SACMAT 2: 163 –168.

Park J, Sandhu R (2002). "Towards Usage Con- trol Models: Beyond Traditional Access Con- trol," in SACMAT '02: Proceedings of the seventh ACM Symposium on Access Control Models and Technologies. New York, NY, USA: ACM Press, pp. 57–64.

Sailer R, Zhang X, Jaeger T, van Doorn L (2004). "Design and implementation of a TCG-based integrity measurement architecture". In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13 (San Diego, CA, August 09 - 13, 2004). USENIX Security Symposium. USENIX Association, Berkeley, CA, 16-16.

Sandhu R (1996). "Rationale for the RBAC96 Family of Access Control Models," in RBAC '95: Proceedings of the first ACM Workshop on Role-based access control. New York, NY, USA: ACM Press, p. 9.

Sandhu RS (1993). "Lattice-Based Access Control Models," Computer, 26(11): 9–19.

Wang L, Wijesekera D, Jajodia S (2004). "A logic-based framework for attribute based ac- cess control," in Proceedings of the 2004 ACM workshop on Formal methods in security engineering. ACM New York, NY, USA, pp. 45–55.

Zhang X (2006). "Formal Model and Analysis of Usage Control," PhD Dissertation. George Mason University, VA, USA, 2006, available at: http://www.list.gmu.edu/dissert/xinwen diss.pdf .

Zhang X, Parisi-Presicce F, Sandhu R, Park J (2005). "Formal model and policy specification of usage control". ACM Trans. Inf. Syst. Secur. 8(4): 351-387.

Appendix: Algorithms

Algorithm 1

```
Algorithm 1: Flow In Pre Models
Input: Two UCON Policies r and w
// r corresponds to a policy (s, $o_1$, read) and
// w corresponds to a policy (s, $o_2$, write)
Output: Boolean value representing information flow between $o_1$ and $o_2$
Method:
$F_{12}$ := false; // no possible information flow from $o_1$ to $o_2$ yet
//Checking for Information Flow due to Subjects:
allowedwrite := false; // is s currently allowed to read from $o_1$
allowedread := false; // is s currently allowed to write to $o_2$
for each state in timeline
    if (tryaccess($s_1$, $o_1$, read) $\wedge$ predicates = true) then
        allowedread := true;
        if (allowedwrite = true) then $F_{12}$ := true; end if
    end if
    if (tryaccess($s_1$, $o_2$, write) $\wedge$ predicates = true) then
        allowedwrite := true;
        if (allowedread = true) then $F_{12}$ := true; end if
    end if
    if (endaccess($s_1$, $o_1$, read)) then allowedread := false;
    if (endaccess($s_1$, $o_2$, write)) then allowedwrite := false;
end for each
//Checking for Information Flow due to System:
attrs := $\varphi$ // no information flow from any attributes yet
for each state in timeline
    for each updatestatement u in r
        if (u is a function of o1 .$attr_x$ updating s.$attr_i$) then
            attrs := attrs $\cup$ { s.$attr_i$ }
        end if
    end for each
    for each updatestatement u in w
        if (u is a function of s.attrx updating $o_2$.$attr_i$ $\wedge$ s.$attr_x$ $\in$ attrs) then
            $F_{12}$ := true;
        end if
    end for each
end for each
return $F_{12}$
```

Algorithm 2

```
Algorithm 2: Flow In On M odels
Input: Two UCON Policies r and w
// r corresponds to a policy (s, o1 , read) and
// w corresponds to a policy (s, o2 , write)
Output: Boolean value representing information flow between o1 and o2
Method:
//Checking for Information Flow due to Subjects:
$F_{12}$ := false; // no possible information flow from $o_1$ to $o_2$ yet
for each state in timeline
    if (tryaccess($s_1$ , $o_1$ , read) $\wedge$ tryaccess($s_1$ , $o_2$ , write))then
        $F_{12}$ := true;
    end if
end for each
//Checking for Information Flow due to System:
attrs := $\varphi$ // no information flow from any attributes yet
for each state in timeline
    for each updatestatement u in r
        if (u is a function of $o_1$.attr$_x$ updating s.attr$_i$ )then
            attrs := attrs $\cup$ { s.attr$_i$ }
        end if
    end for each
    for each updatestatement u in w
        if (u is a function of s.attr$_x$ updating $o_2$.attr$_i$ $\wedge$ s.attr$_x$ $\in$ attrs) then
            $F_{12}$ := true;
        end if
    end for each
end for each
return $F_{12}$
```