Access Controls and Trust Management

Zach Kissel

References

A State-Transition Model of Trust Management and Access Control
Ajay Chander, Drew Dean, and John C. Mitchell

Reconstructing Trust Management
Ajay Chander, Drew Dean, and John C. Mitchell

Overview

- What is Access Control
- Lampson's Access Matrix
- Model for Access Controls
- Demonstrate correctness of Model
- A Trust Management solution

What is Access Control

Access Control is a way to associate a set of objects *O*, a set of rights *R*, and a set of subjects $S \subseteq O$ such that a right $r \in R$ is enforced with respect to how a subject $s \in S$ interacts with an object $o \in O$

Lampson's Access Matrix (1971)

Let A be an $m \times n$ matrix in $\{0,1\}^{m \times n}$ with columns labeled by the pair $(o, r)_i$ where $o \in O$ and $r \in R$ and with rows labeled with subjects, $s_j \in S$. Moreover, if $a_{i,j} = 1$ then subject s_j can perform $(o, r)_i$.

Two Ways to Look at Lampson's Access Matrix

1. Access Control List (ACL)

- In terms of Lampson's Access Matrix, an ACL for object $o \in O$ with right $r \in R$ is defined as the column corresponding to pair (o, r) in the access matrix.

- This is the Unix model we are all familiar with. Namely, rights belong to objects.

2. Capabilities

- In terms of Lampson's Access Matrix, capabilities for a subject *s* is defined as the row of the access matrix corresponding to *s*.
- Intuitively, the rights reside with the users not the objects
- There are other ways to represent capabilities such as using unforgeable bit strings.

Modeling Access Control

Goals

We want a way to model access controls so we can systematically compare and contrast different types of access control.

A State Transition Model

- A *world* state, *WS*, which contains the state of system at a given point in time.
- A set of *Actions*, Σ, which defines a transition from one world state to another.
- An *Access Judgment* $WS \vdash s \to (o, r)$ which means in the world state *WS* subject *s* can access object *o* with right *r*.

Modeling ACL's

- Define the world state *WS* as the map:
  $A : O \times R \rightarrow P(S)$ where, $S \subseteq O$

- The set of actions for ACL's will be defined as
  $\Sigma = \{\mathrm{Create}, \mathrm{Allow}, \mathrm{Revoke}, \mathrm{Delete}\}$

- Let the access judgment rule be defined as:
  $$WS \vdash s \rightarrow (o, r) \stackrel{\text{def}}{=} s \in A((o, r))$$

Create and Delete Actions

- $Create(s_c, o) = (O \cup \{o\}, R, S \cup \{s_c\}, A')$

  Where, $A'(o, r) = \begin{cases} s_c & \text{if } r = r_e \\ \emptyset & \text{if } r \neq r_e \end{cases}$

- $Delete(o) = (O - \{o\}, R, S - \{o\}, A|_{(O - \{o\}, R, S - \{o\})})$

Allow and Revoke Actions

- $Allow(s, o, r) = (O, R, S \cup \{s\}, A')$

  Where, $A' = A[(o, r) \rightarrow A((o, r)) \cup \{s\}]$

- $Revoke(s, o, r) = (O, R, S \ominus \{s\}, A')$

  Where, $S \ominus \{s\} = \begin{cases} S \text{ if } \left|A^{-1}(\{s\})\right| \geq 2 \\ S - \{s\} \text{ otherwise} \end{cases}$

  $A' = A[(o, r) \rightarrow A((o, r)) - \{s\}]$

Modeling Capabilities

- Define the world state *WS* as the map: $C: S \rightarrow P(O \times R)$ where, $S \subseteq O$
- The set of actions for capabilities will be defined as $\Sigma = \{\text{Create}, \text{Delete}, \text{Grant}, \text{Revoke}\}$
- Let the access judgment rule be defined as:

$$WS \vdash s \rightarrow (o, r) \stackrel{\text{def}}{=} ((o, r) \in C(s))$$

Create and Delete Actions

- $Create\left(s_c, o\right) = \left(O \cup \{o\}, R, S \cup \{s_c\}, C'\right)$

  Where, $C'\left(s_c\right) = \begin{cases} \left\{\left(o, r_e\right)\right\} \text{ if } s_e \notin S \\ C\left(s_c\right) \cup \left\{\left(o, r_e\right)\right\} \text{ if } s_c \in S \end{cases}$

- $Delete\left(o\right) = \left(O - \{o\}, R, S - \{o\}, C_{|\left(S - \{o\}, O - \{o\}\right)}\right)$

Grant and Revoke Actions

- $Grant(s, o, r) = (O, R, S \cup \{s\}, C[s \rightarrow C(s) \cup \{(o, r)\}])$

- $Revoke(s, o, r) = (O, R, S', C')$

  Where, $S' = \begin{cases} S - \{s\} \text{ if } C(s) = (o, r) \\ S \text{ if } C(s) \neq (o, r) \end{cases}$

  $C' = C[s \rightarrow C(s) - \{(o, r)\}]_{s \in S'}$

Reasoning about the Models

Comparing The Models

- In order to compare the models to one another we need to we introduce relations and mappings to reason about the strength of each access model.

- In our present case, we can show that we can map an ACL model to a Capabilities model in such a way that the models behave the same

Bisimulation Relation

Given a set P of states and a set T of transitions let $p, p' \in P$ and S be a binary relation over P such that if it holds that pSq then if $p \stackrel{\alpha}{\rightarrow} p'$, then $\exists q, q' \in P$ such that $q \stackrel{\alpha}{\rightarrow} q'$ and $p' S q'$ The relation is known as a stong simulation.

A Mapping from ACLs to Capabilities

Define a mapping f from WS_A to WS_C as follows:

$$f\left(Create\left(s_c, o\right)\right) = Create\left(s_c, o\right)$$

$$f\left(Delete\left(o\right)\right) = Delete\left(o\right)$$

$$f\left(Allow\left(s, r, o\right)\right) = Grant\left(s, o, r\right)$$

$$f\left(revoke\left(s, r, o\right)\right) = Revoke\left(s, o, r\right)$$

Capabilities strongly Simulate ACLs

- We can show that the previous mapping sends an ACL model to a bisimilar Capabilities model

- We can also show that we can go in the other direction.

Disadvantage of ACLs and Capabilities

- One of the major drawbacks of the access control methods presented thus far is they can not easily handle cascading revocation of rights.

- Can we use the formalism presented to help us in determining a better access control policy?

Trust Management

(A Stronger form of Access Control)

What is a Trust Management System?

- A system in which an access request is accompanied by a set of credentials which together constitute a proof as to why the access should be allowed.

- Access is enforced by using a *root access control list* composed of a small group of "super users" and policies implemented by delegation

Modeling Trust Management

- Define the world state *WS* as the maps:

$A: O \times R \rightarrow P(O \times \mathbb{N})$ and $D: O \times R \times O \rightarrow P(O \times \mathbb{N})$

- The set of actions for capabilities will be defined as:

$$\Sigma = \left\{ \text{Create}, \text{Add}, \text{Remove}, \text{Delegate}, \text{Revoke}, \text{Delete} \right\}$$

Access Judgment in Trust Management

- Two set membership functions:

$ACL(s, o, r, d)$ is true *iff* $(s, d) \in A((o, r))$

$Del(s, o, r, r_s, d)$ is true *iff* $(r_s, d) \in D(s, r, o)$

- One Rule

Subject s can access the (o, r) pair *iff* it can produce a proof of $Access(s, o, r, d)$, for some d, from the world state and the provided inference rules.

Access Proof Inference Rules

- Root ACL: $ACL(A, B, r, d) \supset Access(A, B, r, d)$
- Delegation: $Access(A, B, d+1)$
  $\wedge \; Del(A, B, r, C, d)$
  $\supset \; Access(C, B, r, d-1)$
- Ord1: $Access(A, B, d+1) \supset Access(A, B, d)$
- Ord2: $Del(A, B, r, c, d+1) \supset Del(A, B, r, c, d)$

Create and Delete Action

- $Create(o_c, o) = (O \cup \{o\}, R, A', D')$

  Where, $A'(o, r) = \begin{cases} (o_c, 1) \text{ if } r = r_e \\ \emptyset \text{ if } r \neq r_e \end{cases} \quad \forall r \in R$

  $D' = D[(s, r, o) \to \emptyset \mid s \in O, r \in R]$

- $Delete(o) = (O - \{o\}, A|_{O-\{o\}}, D|_{O-\{o\}})$

Add and Remove Actions

- $Add(o, r, o_s, d) = (O, R, A', D)$
  Where, $A' = A\left[(o, r) \rightarrow A((o, r)) \cup \{(o_s, d)\}\right]$
- $Remove(o, r, o_s, d) = (O, R, A', D)$
  Where, $A' = A\left[(o, r) \rightarrow A((o, r)) - \{(o_s, d)\}\right]$

Delegate and Revoke Actions

- $Delegate\left(o_s, o, r, o_d\right) = \left(O, R, A, D'\right)$

  Where, $D' = D\left[\left(o_s, r, o\right) \to D\left(\left(o_s, r, O\right)\right) \cup \left\{\left(o_d, d\right)\right\}\right]$

- $Revoke\left(o_s, o, r, o_d\right) = \left(O, R, A, D'\right)$

  Where, $D' = D\left[\left(o_s, r, o\right) \to D\left(\left(o_s, r, O\right)\right) - \left\{\left(o_d, d\right)\right\}\right]$

Comparing ACLs and Trust Management

- It can be shown, similar to how we showed ACLs were equivalent to Capabilities, if the delegation depth is limited to zero then trust management will strongly simulate ACLs

- It can also be shown that ACLs can't simulate the general Trust Management, because of the cascading effects of a deletion and revocation of rights.

Completing The Trust Management Model

- The trust management system shown is incomplete.

- In a later paper Chander, Dean, and Mitchell extend there model to take into account Fully Qualified Names (FQNs). A way of accessing objects in a distributed system.

- They argue that FQNs are irrelevant to the actual analysis of Trust Management.

Conclusions

- In the papers it was shown that Trust Management offers a stronger solution to the access control problem, as opposed to the currently implemented methods.

- This was accomplished through a rather simple model.

- For a discussion of implementation in a kernel and how FQNs are used see “Reconstructing Trust Management.”

Questions?