RBAC Administration in Distributed Systems

Marnix Dekker, Jason Crampton, Sandro Etalle

Distributed and Embedded Systems group (DIES), University of Twente
Information Security Group (ISG), Royal Holloway University of London
Security Group (SEC), University of Eindhoven

SACMAT 2008
Motivation

Modern distributed computer systems require sophisticated access control policies

- Statutory and enterprise requirements
- RBAC is a flexible and widely used form of access control

Access control policies need to be administered simply and effectively

- Policy requirements change

Existing RBAC literature does not provide a model for administration in distributed systems
Role-based access control

“Standard RBAC” (RBAC96, ANSI-RBAC) defines

- Users $U$, roles $R$, actions $A$ and objects $O$
- Permissions $P \subseteq A \times O$
- $UA \subseteq U \times R$, $PA \subseteq P \times R$, $RH \subseteq R \times R$
- An RBAC policy $\phi$ is defined by $(UA, RH, PA)$
Role-based access control

We treat $\phi$ as a directed graph $(U \cup R \cup P, UA \cup RH \cup PA)$

- We write $v \rightarrow_{\phi} v'$ to indicate that there exists a path from $v$ to $v'$ in $\phi$
- We assume all paths are directed (from users to roles to permissions)
- $u \rightarrow_{\phi} r$, for example, means that $u$ is authorized (by $\phi$) for role $r$
Role-based access control

We treat \( \phi \) as a directed graph \((U \cup R \cup P, UA \cup RH \cup PA)\)

- We write \( v \rightarrow_{\phi} v' \) to indicate that there exists a path from \( v \) to \( v' \) in \( \phi \)
- We assume all paths are directed (from users to roles to permissions)
- \( u \rightarrow_{\phi} r \), for example, means that \( u \) is authorized (by \( \phi \)) for role \( r \)
- The upper closure of \( v \), denoted \( \uparrow_{\phi} v \), is the sub-graph comprising all paths in \( \phi \) in which \( v \) is the last node
- The downward closure of \( v \), denoted \( \downarrow_{\phi} v \), is the sub-graph comprising all paths in which \( v \) is the first node
Administration of RBAC

What administrative actions may be requested?

- We only model changes to \(UA\), \(RH\) and \(PA\)
- A user may add or delete a tuple from one of these relations
- \(\Box_u(v, v')\) denotes a request by \(u\) to add tuple \((v, v')\)
- \(\lozenge_u(v, v')\) denotes a request by \(u\) to delete tuple \((v, v')\)
Administration of RBAC

What administrative actions may be requested?

- We only model changes to $UA$, $RH$ and $PA$
- A user may add or delete a tuple from one of these relations
- $\square_u(v, v')$ denotes a request by $u$ to add tuple $(v, v')$
- $\blacklozenge_u(v, v')$ denotes a request by $u$ to delete tuple $(v, v')$

What administrative permissions are required?

- $\Box(v, v')$ denotes a permission to add tuple $(v, v')$
- $\lozenge(v, v')$ denotes a permission to delete tuple $(v, v')$
- We extend $PA$ to include permissions of the form $\Box(,)$ and $\lozenge(,)$
1 Introduction

2 Modeling Distributed Systems

3 Policy Distribution

4 Concluding Remarks

5 Questions
Distributed system model

We assume that a distributed system comprises a number of components (or sub-systems) $S$

- Each sub-system $s \in S$ has its own reference monitor and its own policy for deciding access requests.
- There is a centralized reference monitor for deciding administrative access requests.
- The centralized reference monitor has policy $\phi$. 
We define a privilege mapping $pm$ and a policy mapping $\psi$

- $pm(s) \subseteq P$ is the set of permissions handled by sub-system $s$
- $\psi(s)$ denotes the RBAC policy that sub-system $s$ uses to evaluate requests

We model the distributed system as a tuple $(S, pm, \phi, \psi)$
Soundness

\( \psi \) is sound (with respect to the central policy \( \phi \)) iff

\[
\bigcup_{s \in S} \psi(s) \subseteq \phi
\]

- \( \psi \) is sound if any request granted by \( s \in S \) would also be granted by a centralized reference monitor using policy \( \phi \)
- Soundness is a safety criterion
Completeness

ψ is complete (with respect to the central policy φ) iff for any \( s \in S \) and any \( p \in pm(s) \)

\[
\psi \text{ is complete if } \psi(s) p \text{ implies } u \rightarrow \psi(s) p
\]

- Complete if any request granted by \( \phi \) for a permission for which \( s \) is responsible is also granted by \( s \)
- Completeness is an availability criterion
Leanness

Soundness and completeness are minimum requirements of a policy distribution \( \psi \)

- Trivial distribution \( \psi(s) = \phi \) for all \( s \in S \) is sound and complete
- More economical distributions are desirable
Leanness

Soundness and completeness are minimum requirements of a policy distribution $\psi$

- Trivial distribution $\psi(s) = \phi$ for all $s \in S$ is sound and complete
- More economical distributions are desirable
- It can be shown that the most economical sound and complete distribution is defined by

$$\psi(s) = \bigcup_{p \in pm(s)} \uparrow_\phi p$$

- We call this the lean distribution
Example

\[\phi\]

\[\psi(Sqan)\]

\[\psi(\text{Inq})\]

\[\psi(Sqil)\]
Introduction

Modeling Distributed Systems

Policy Distribution

Concluding Remarks

Questions
Policy updates

When an administrative request is granted by the central reference monitor (CRM), policy updates need to be sent to one or more sub-systems

- It is important that soundness and completeness are preserved
Policy updates

When an administrative request is granted by the central reference monitor (CRM), policy updates need to be sent to one or more sub-systems

- It is important that soundness and completeness are preserved

We propagate policy updates to sub-systems using message commands

- A message command is parameterized by a sub-system, a policy graph and an action (add or delete)
- $\oplus_s(\phi)$ (respectively $\ominus_s(\phi)$) denotes a message for sub-system $s$ to add (delete) $\phi$ to (from) its policy

The operational semantics of our model are defined using a queue and a transition relation
Queues and transitions

Administrative requests and message commands are placed on the queue

- Processing an item in the queue – defined by the transition relation – yields a new queue
Queues and transitions

Administrative requests and message commands are placed on the queue

- Processing an item in the queue – defined by the transition relation – yields a new queue

Given \((S, pm, \phi, \psi)\), the transition relation

- Transforms administrative requests into message commands and updates to \(\phi\)
  - If \(\Box_u(v, v')\) is authorized then create message command

\[ \oplus_s(\{(v, v')\} \cup (\uparrow_\phi v)) \]

for each \(s\) such that \(\downarrow_\phi v \cap pm(s) \neq \emptyset\)

- Transforms message commands into updates to \(\psi\)
- The transition relation preserves soundness and completeness
1 Introduction

2 Modeling Distributed Systems

3 Policy Distribution

4 Concluding Remarks

5 Questions
Multiple administrative systems

We have assumed the existence of a centralized reference monitor for administrative requests

- How does the situation change when we have multiple administrative reference monitors (ARMs)?

The central policy $\phi$ is distributed across the ARMs

- Thereafter we can only infer the global policy from the policies held by the set of ARMs
Policy support

Each ARM must be able to implement the queue transition function:

- This requires each ARM to have the relevant parts of $\phi$
- The **policy support** of an administrative permission $\Box(v, v')$ is defined to be $\uparrow_\phi v \cup \downarrow_\phi v'$
- For each administrative permission handled by an ARM, the policy support for $\Box(v, v')$ must be part of the policy
Conclusions

We presented a distributed system model with RBAC policies

- Formal requirements for RBAC administration in distributed systems
- Administrative procedure for policy changes
- Pseudo-code implementation (in proceedings)
- Preliminary treatment of multiple administrative systems

Future work

- Complete treatment of multiple ARMs
- Constraint specification, enforcement and administration