Assured Resource Sharing in Ad-hoc Collaboration

DE-FG02-10ER25984
DOE PI Meeting, Sept 16-17 2014

Gail-Joon Ahn
Professor, Computer Science and Engineering
Arizona State University
# Assured Resource Sharing in Ad-Hoc Collaboration

**PI: Gail-Joon Ahn**

## Project Goals

- Develop an innovative framework to enable users to access and selectively share resources in distributed environments
- Investigate secure sharing and assurance mechanisms for ad-hoc collaboration, focused on Grids, Clouds, and Virtual Network Communities

## Current Accomplishments

- Articulated sharing patterns and corresponding access control model and developed analysis module for policy anomalies that violate sharing requirements
- Published and disseminated research results through leading security journals such as *IEEE Transactions on Dependable and Secure Computing* and *Journal of Computer Security*
- Established a software-defined infrastructure to articulate requirements relevant to delegation and access control modules

## Impacts on DOE's Mission

- Enabling research community with a security-aware, scalable framework to sharing resources in a secure and selective manner
- Producing deployment architectures and software modules for establishing trustworthy collaboration environments including access control and delegation management in such dynamic network environments

---

URL: sefcom.asu.edu

The Laboratory of Information Integration, Security and Privacy
Problem statement

- Information sharing in ad-hoc collaboration is always *conditional*, and needs to be *highly controlled*.

- Approaches
  - Secure sharing in Grids and Cloud
    - Effective access control framework
  - Policy analysis for assurance
Secure Sharing: Access Control Requirements

- Access management requirements:
  - The originator needs an effective way to define the virtual collaborative sharing domain and authorize the unknown collaborators inside the domain.
  - Access control should guarantee the sharing occurs within the originator's collaborative sharing domain, and sharing behaviors must be well regulated.
Secure Sharing: sharing patterns

Organizational level

Individual collaborator level

originator

Participant abstraction via role

Normative collaborative sharing role

Virtual Collaborative sharing domain
Secure Sharing with P2P – ShareEnabler

1. start new search + choose credentials
2. invoke
3. create file query msg
4. send file query message
5. broadcast file query message
6. receive response
7. notice
8. show search results

User

ShareEnabler Agent 1
(Requester)

ShareEnabler Agent 2
(Responder)

Collaborative Sharing Group

Executive Services
- Search Service
- Share Service
- Data Sharing Service
- Metadata Sharing Service
- Data Management Service
- Download Service
- RAMARS AuthZ / Enforcement
  - PEP
  - RAMARS

SGL/IG

TLS/TCP

1. receive file query
2. invoke
11. response
12. response
13. send response back to requester

10. enforce the decision
9. access decision
8. parse and evaluate policies
5. invoke
6. query for access decision + requester’s credentials
7. retrieve ROA policies

RAMARS AuthZ / Enforcement

Policy Engine

AC Engine

LDAP

Policy Store

URL: sefcom.asu.edu
Secure Sharing with Grids – RamarsAuthZ service
Secure Sharing with Cloud–ACaaS$_{RBAC}$

- ACaaS$_{RBAC}$ introduces RBAC as a service (RaaS), which is an RBAC module that can be hosted by AWS or any third-party service provider.
Secure Sharing with Cloud–ACaaS_{RBAC}

- In order to measure scalability of ACaaS_{RBAC}, measure average performance overhead while increasing the numbers of simultaneous role activation and deactivation requests from users.

![Diagram](attachment:image.png)

(a) Activation Time
Problem statement (revisited)

- Information sharing in ad-hoc collaboration is always \textit{conditional}, and needs to be \textit{highly controlled}.

- Approaches
  - Secure sharing in Grids and Cloud
    - Effective access control framework
  - Policy analysis for assurance

URL: sefcom.asu.edu

The Laboratory of Information Integration, Security and Privacy
Policy analysis for assurance

**Motivation**

- **Access Control Policies**
  - Handle *complex system properties* by separating policies from system implementation
  - Enable *dynamic adaptability* of system behaviors by changing policy configurations without reprogramming the systems

**Challenge**

- Ensuring the *correctness* of these policies is critical, and yet difficult
  - Demands strong support of automated *reasoning* techniques
  - Demands systematic mechanism for policy *anomaly management*
Anomaly Management for Access Control Policy

- **Policy conflict**
  - Conflicts in a policy may lead to
    - **Safety** problem (e.g. allowing unauthorized access)
    - **Availability** problem (e.g. denying legitimate access)

- **Policy redundancy**
  - Redundancies in a policy may adversely affect the performance of policy evaluation
    - Response time of an access request largely depends on the number of rules to be parsed
Conflict detection approach

- Policy-based segmentation technique
  - Partition the entire authorization space of a policy into disjoint segments
- Identification of conflicting segments
  - Each conflicting segment indicates a conflict

Algorithm 1: Identify disjoint conflicting Authorization Spaces of Policy $P$

```
Input: A policy $P$ with a set of rules.
Output: A set of disjoint conflicting authorization spaces $CS$ for $P$.
1. $P$ = Partition the entire authorization space of $P$ into disjoint spaces $\forall$
2. $S$ = New();
3. $S$ = Partition_P($P$);
4. $P$ = Identify the conflicting segments $\forall$
5. $CS$ = New();
6. foreach $s$ $\in$ $S$ do
   7. $R$ = GetRule($s$);
   8. if $\exists r_i \in R, r_j \in R, r_i \neq r_j \text{ and } \text{Effect}(r_i) \neq \text{Effect}(r_j)$
      then
       9. $CS$.Append($s$);
10. Partition_P($P$);
11. $R$ = GetRule($P$);
12. foreach $r$ $\in$ $R$ do
13.   $s_r$ = AuthorizationSpace($r$);
14. $S$ = Partition($S, s_r$);
15. Partition($S, s_r$);
16. foreach $s$ $\in$ $S$ do
17.     if $s_r$ $\subseteq$ $s$ then
18.        $S$.Append($s \setminus s_r$);
19.        $s$ = $s_r$;
20.        Break;
21.     if $s_r$ $\supseteq$ $s$ then
22.        $S$.Append($s \setminus s_r$);
23.        $s$ = $s_r$;
24.     else if $s_r$ $\cap$ $s$ $\neq$ $\emptyset$ then
25.        $S$.Append($s \setminus s_r$);
26.        $s$ = $s_r$ $\cap$ $s$;
27.        $s_r$ = $s_r$ $\setminus$ $s$;
28. else $s_r$ partially matches $s$
29.    $S$.Append($s \setminus s_r$);
30. $S$.Append($s_r$);
31. return $S$;
```
Anomaly Management for Access Control Policy -- Conflict Detection (cont’d)

- Overlapping authorization space for a policy
  - With two dimensional geometric representation

- Space segmentation

URL: sefcom.asu.edu
Anomaly Management for Access Control Policy -- Redundancy Removal

- Segment classification
  - Non-overlapping segment \((s_2, s_4, s_7)\)
  - Overlapping segment
    - Conflicting overlapping segment \((s_3, s_5)\)
      - Indicate a conflict
    - Non-conflicting overlapping segment \((s_1, s_6)\)
      - Indicate a potential redundancy

URL: sefcom.asu.edu
Evaluation (cont’d)

- Evaluation of redundancy removal approach
  - Traditional approach: only identify redundancy relations between two rules

Redundancy elimination rate

Performance improvement
Summary: Next Step

- Information sharing in ad-hoc collaboration is always *conditional*, and needs to be *highly controlled*.

- Approaches
  - Secure sharing in Grids and Cloud
    - Effective access control framework
  - Policy analysis for assurance
    - Policy composition and schema integration
  - Attribute-based multi-party control
Exploring Attributes: OSCARS

End-User Client Interface

1) Sending ID credentials
2) Retrieving User's attributes (roles)
3) Requesting Service

Coordinator Service

4) Requesting Permissions using user's attributes (roles)
5) Retrieving user's permissions and conditions
6) Enforcing AC Policy
7) Retrieving AC Decision
8) Prompting other services
9) Returning Results

Authentication Service (AuthN)

1.5) Calling AuthN from other domains

Authorization Service (AuthZ)

4.5) Calling other AuthZ from other domains

The Laboratory of Information Integration, Security and Privacy
URL: sefcom.asu.edu
Exploring Attributes: perfSONAR

1) Sending Attribute IDs
2) Retrieving Authentication Token
3) Requesting Service
4) Requesting Access using Auth Token
5) Requesting Additional Attributes (Roles)
6) Retrieving Additional Attributes (Roles)
7) Enforcing Local Policy
8) Access Control Response
9) Calling other MPS from other domains
10) Returning Results

End-User Client Interface

Measurement Point Service (MPS)

Authentication Service (AS)

Resource Protector Service (RPS)

5.5) Calling AS from external domains

URL: sefcom.asu.edu

The Laboratory of Information Integration, Security and Privacy
What question does your research motivate you to now ask?

- Can we discover access patterns, provision access privileges, and generate access intelligence?

- How can we cope with the resources handled by multiple parties?
  - Multi-party access control
  - Multi-party policy evaluation

- Is the federation of access control services required?
Selected results


* indicates students