User-Centric Access Control Policy Management Framework for Cloud Applications

Thesis Final Defense
School of Electrical Engineering & Computer Science, NUST
Islamabad
Agenda

- Introduction
- Motivation
- Literature Review
- Problem Statement
- Research Contributions
- User-Centric Access Control Architecture
- Designed Protocol
- Validation and Evaluation
- Conclusion
- Demonstration
Introduction

- Personal Computers (PC)
- Desktop Applications
- World Wide Web (Internet)
- Web Applications
- Cloud Computing
- Cloud Applications
Features of Cloud Applications:

- Online/Offline Availability
- Variety
- Easy Access
- Low Cost
Users are storing more and more resources in the forms of pictures, videos, documents, blogs etc. through the use of laptops, tablets, PCs and smartphones.
Motivation

- **Features of Current Framework of Cloud Environment:**
  - Numerous Applications
  - Host User Resources
  - Authentication and Authorization (Centralized or Federated)
  - Access control mechanisms (Application Centric)

- **Loop Holes:**
  - Application centric access control mechanism are:
    - Tightly bound to the functionality of the application
    - Provide limited access control options
    - Does not cater user’s access control requirements

- **Possible threats:**
  i. Unauthorized access
  ii. Privacy Leakage
  iii. Data theft
  iv. Identity Theft
Motivation
Application Centric Access Control
Challenges

Scenario

- A Cloud user Chris who is a frequent traveler and a hobbyist photographer.
- He takes pictures, make videos and documents his entire trips.
- To store and share these resources, he uses three famous Cloud applications.

- CPics (an online gallery to host pictures)
- CVideos (an online video hosting service)
- CDocs (a Cloud based word processor to create and manage different documents).
Motivation
(Scenario)

To share his resources with his friends Bob and Alice

- Specify access control on pictures
- Specify access control on videos
- Specify access control on documents
Motivation
(Scenario)

- Chris decides to share these pictures, videos and documents with an additional person then
  - He again logs in to all three applications
  - Change access control policies on resources accordingly.

- Chris decides to share more of his resources with the same group of people
  - Composes new access control policies at each Cloud application
  - Applies these policies to resources as necessary.
Weaknesses of the existing access control solutions:

**Isolated authorization mechanisms**
- Access control is tightly incorporated with the functionality of the application.
- Lack fine grained access control functionality.
- Simple solutions data is either made public or private.
- Unable to configure individual user’s access control requirements.

**No standard access control policy language**
- Different Cloud applications use different authorization mechanisms.
- Can not define same access control policies only once and apply to different resources.
- Can not transfer access control policies to other applications.

**Management of access control policies**
- User’s resources and their related access control policies are spread across the Cloud.
- User does not have a consolidated view of all the applied security controls.
- User have to log in to every application to add, update or audit access control policies.
We formulate the following shortcomings of the existing solution based on the weaknesses we observed:

- **Existing Access control solutions are inadequate and does not cater individual user’s security and access control requirements.**
- **User has to employ diverse and complex access control policy languages.**
- **User does not have a unified control point to manage access control policies scattered across numerous Cloud applications.**
Architecture and Protocol for User-Controlled Access Management in Web 2.0 Applications

Theme:
- The authors describe a User Centric Access Control Model for web applications by providing an external authorization server.
- Authorization server provides users with an interface to define access control policies according to their requirements.
- It provides authorization tokens to requestors to access protected resources.

Critical Analysis:
- Designed protocol is complex and comprises of many steps.
- Time consuming process.
- Web application act as policy enforcement point. It may not have the built in functionality to parse resources according to user requirements.

**xAccess: A Unified User-Centric Access Control Framework for Web Applications**

**Theme:**
- This paper presents a **user centric access control** framework for **web applications**.
- **xAccess extension** component enable user to specify access control on their content before uploading on the internet.
- **xAccess server** component resides on the web application server and parse content according to user’s specifications.

**Critical Analysis:**
- Not a generic system and operates in **close environment**.
- Different access control models have to be implemented in xAccess extension in order to **translate** them into **base models**.

Policy Management as a Service: An Approach to Manage Policy Heterogeneity in Cloud Computing Environment

Theme:
- **Policy management framework** for Cloud environment is presented in this paper.
- **Policy Management Service Provider (PMSP)** server provides users with an interface and tools to generate and manage access control policies on their resources.
- PMSP translates these policies into Cloud service provider’s (CSP) policy format and then transfer these policies to the CSP’s policy repository.

Critical Analysis:
- **Complex process** (Conversion of user defined policies into different CSP’s policy languages).
- **Compatibility issues** (no relationship and coordination between the policy creation and policy enforcement modules).

A User-Centric Privacy Access Control Model

➢ Theme:
  • User’s privacy policies are separated from access control policies.
  • User is given control to create policies regarding its private information and stores them on the server.
  • Requests to access private information is redirected to the server which generates a decision by evaluating user defined policies.

➢ Critical Analysis:
  • Does not provide user with the control to specify access control policies.
  • User can not protect resources according to his requirements.

Application centric authorization mechanisms does not cater individual user’s access control requirements, which leads to unprotected resources susceptible to different security threats. In order to address these concerns we propose a user centric access control framework which provides users with the control to create and manage access control policies according to their security requirements.
Objectives

Objective 1

• To design an authorization framework which enables users to protect their resources according to their security and access control requirements.

Objective 2

• To provide users with an integrated central control point to manage access control on all its resources scattered across the Cloud.
Contributions
Research perspective


User-Centric Access Control Architecture
The designed protocol consists of three sub protocols which are:

- Authentication and Authorization
- Access Control Policy Specification
- Accessing Protected Resources
Authentication and Authorization

- User first needs to authenticate himself with the SA server.
- Authentication is done through FIPS 196 protocol.
- After authentication User gets access to his desired Cloud application.

Authentication Ticket = \{(userID, TimeStamp, LifeTime) + \{(userID, TimeStamp, LifeTime)\}EPRsa \} EPUGw
User can create or upload resources on Cloud applications.

Policy specification module allows users to create access control policies on their resources.
Accessing Protected Resource

Requestor

Request "Resource"

Resource

Gateway

XACML request (Resource URL + Requestor ID)

Response (Permit/Deny)

Cloud Application

Resource URL
Information Flow in a Nutshell

User A
- SAML ticket + application URL
- Define Access Control Policies

Gateway
- Verify Ticket
- User ID, Application URL
- Upload Policies
- Resource Access Request
- Parse Resource

Authorization Server
- Evaluate Policies
- Access Decision
- User ID, Login Request
- User Login Access
- Create Resource
- Request, Requestor ID
- Evaluate Policies
- Access Decision
- Resource Request
- Resource

Cloud Application

Requestor
Google Spreadsheet Case Study:

- User can store different types of data on spreadsheet.
- Can perform different functions on the data.
- Google applies Access Control List mechanism to define the sharing process.
- User can make the whole spreadsheet either:
  - Public (visible to everyone)
  - Private (only visible to owner)
  - Anyone with the link
  - Specific people (by providing their email addresses)
- Lack fine-grained access control functionality.
- Does not allow users to share different records of the spreadsheet to different users.
Example Scenario 1:

A User who is a teacher often uses Google Spreadsheet to share different kinds of information to students and other faculty members for different purposes.

He needs to share a spreadsheet containing certain information according to the following policy.

“Only Faculty members of the information security department having education equal to PhD can read the spreadsheet ‘mySheet’ in working hours i.e. 9 to 5 pm.”

Subject Attributes:
- role = Faculty
- Department = Information Security
- Education = PhD

Rule Condition = time b/w 9 to 5
Example Scenario 1:

Requestor

Gateway

Authorization Server

Cloud Application

“mySheet” access request

XACML Request
(requestor ID, Res URL, read)

Response (permit/Deny)

Fetch attribute values from PIP

Resource URL

Resource

Example Scenario 1:
Example Scenario 2:

- A user needs to set access control on different records of the spreadsheet for different users e.g.
  
  "Alice is permitted to access spreadsheet ‘mySheet2’ but can only read rows 2, 5, 10, 15"

- User store row numbers which Alice can read 2, 5, 10, 15 into Database.
Example Scenario 2:

Alice

Gateway

Authorization Server

Cloud Application

"mySheet" access request

XACML Request (requestor ID, Res URL, read)

Response + Obligation

Resource URL

Resource

Row

Authorization Function

Rows 2, 5, 10, 15 of mySheet
EVALUATION

SECURITY

EVALUATION

FUNCTIONALITY
Aspects of Evaluation

- Qualitative and Quantitative Properties
- Organizational Security Objectives
- Correctness and Effectiveness
- Leading versus Lagging Indicators
- Measurement of Large versus the Small
## Threat Model

<table>
<thead>
<tr>
<th>Threat</th>
<th>Protection Granted</th>
<th>Protection Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unauthorized access</td>
<td>Yes</td>
<td>User centric access control mechanism protects resources according to user's requirements and ensures granular level authorized access on resources.</td>
</tr>
<tr>
<td>Privacy leakage</td>
<td>Yes</td>
<td>Policy specification module allow users to specify access control on their personal information in the form of profiles on blog and social networking websites according to their requirements protecting personal information leaking to unauthorized personnel.</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>Yes</td>
<td>Requests need to be validated by authorization server to get application and resource access.</td>
</tr>
<tr>
<td>Weak Authentication</td>
<td>Yes</td>
<td>FIPS-196 Strong Authentication Protocol is used for authentication which mitigates attacks on passwords.</td>
</tr>
<tr>
<td>Bypass</td>
<td>Yes</td>
<td>Designed system does not allow request to bypass PEP or PDP to access resources.</td>
</tr>
</tbody>
</table>
JUnit Testing:

```java
@RunWith(Parameterized.class)
public class PolicyBuilderTest {

    String sub;
    String res;
    String action;
    String rows;
    String Effect;

    private PolicyBuilder policyBuilder;

    @Before
    public void initialize() {
        policyBuilder = new PolicyBuilder();
    }

    public PolicyBuilderTest(String sub, String res, String action, String rows, String Effect) {
        this.sub = sub;
        this.res = res;
        this.action = action;
        this.Effect = Effect;
        this.rows = rows;
    }

    @Parameterized.Parameters
    public static Collection values() {
        return Arrays.asList(new Object[]{1, 2, 3, 4, 5});
    }

    ....
```
<table>
<thead>
<tr>
<th>Category</th>
<th>Test Cases</th>
<th>No. of Test Cases Planed</th>
<th>No. of Test Cases Executed</th>
<th>No. of Test Cases Executed Successfully</th>
<th>No. of Defects Found</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-Centric Policy Creation Test Cases</strong></td>
<td>Policy Creation Test</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Policy Upload Test</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td><strong>Authorization Test Cases</strong></td>
<td>Authorization Test-Accept</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Authorization Test-Deny</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td><strong>Granular level Access Control Test Cases</strong></td>
<td>Row Information Storage Test</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Row Information Retrieval Test</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>120</strong></td>
<td><strong>120</strong></td>
<td><strong>120</strong></td>
<td><strong>0</strong></td>
</tr>
</tbody>
</table>
The designed framework protects user’s resources from unauthorized access and different other security threats by enabling users to specify access control policies according to their own security and access control requirements.

The designed framework also facilitates users by providing them with a single authorization and access control mechanism and a standard policy definition language to specify access control.

Users can define, edit and manage all the applied access controls from a central control point on all its resources independent of their location on the Cloud.
Future Directions

- Incorporation of other access control models like Role based Access Control (RBAC) and Content Based Access Control models to provide user with more fine grained options to protect resources.

- Real time user consent.

- Protection of policies stored on authorization server.

- Protection of messages passed among different components of the designed framework.
References


References


Thank You

Special thanks to my Supervisor & Committee Members
Implementation Demo

User Centric Access Control Policy Management Framework for Cloud Applications