Semantic - Attribute Based Access Control

Uğur Bayram
University of Oslo
Master of Science - Short Thesis - 30 ECTS
Supervisors:
Christian Johansen, Hamed Arshad
Semantic - Attribute Based Access Control
Master Thesis

Uğur Bayram
University of Oslo
Master of Science (MSc) - 30 ECTS

Supervisors:
Christian Johansen, Hamed Arshad
Introduction

01 Subject: Semantic - Attribute Based Access Control

02 Why did I choose

03 SCOTT

04 Goals & Objectives
Motivation

Access Control
- Access control in general
- Access control in IoT

ABAC
- Fine-grained:
  - Flexibility,
  - Dynamicity,
  - Granularity

Limitations
- Centralized (?)
- System complexity
  - Storage
  - Processor
  - Network

Research Objectives

**Objective #1 : ABAC on fog-area**
External access control systems authorizing distributed devices over network is not ideal.

**Objective #2 : Semantic-ABAC**
Plain ABAC has limitations on context extraction. Semantic reasoner helps to solve this.

**Objective #3 : Decentralized Access Control**
Changing the trend centralized approach to decentralized and running improved S-ABAC on resource-constrained IoT gateway.
Research Questions

01. Question#1
What is the importance of the fine-grained access control for IoT settings?

02. Question#2
Can ABAC work for simple IoT devices on the fog-area?

03. Question#3
What is the ideal gateway device for this need?

04. Question#4
How can we link an ABAC engine to a Semantic Reasoner having a simple ontology?
Methodology

**Step 01**
Identify the areas: IoT Security, Access Control Models, ABAC, Semantic Technology, Ontology

**Step 02**
Literature review on journals, articles, previous thesis on similar fields, external blogs and internet resources.

**Step 03**
Define high-level system architecture, review IoT Gateways and select the ideal one.

**Step 04**
Implement Semantic Reasoner and integrate with ABAC engine

**Step 05**
Prototype assessment, performance tests for ABAC & S-ABAC on cloud & fog area,
### Background - Access Control Systems

<table>
<thead>
<tr>
<th>Discretionary Access Control (DAC)</th>
<th>Mandatory Access Control (MAC)</th>
<th>Role Based Access Control (RBAC)</th>
<th>Attribute Based Access Control (ABAC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Resource owner</td>
<td>• Controlling well-characterized information flow: hierarchical</td>
<td>• Subject role defines permission</td>
<td>• As per NIST, evolved from IBAC, RBAC</td>
</tr>
<tr>
<td>• Access control list</td>
<td>• User cannot change rules</td>
<td>• Can be customized per-application</td>
<td>• Subject attributes, object attributes, environmental conditions define the policy</td>
</tr>
<tr>
<td>• Popular in OS resource and directory control</td>
<td>• Suitable for high level confidentiality required organizations: military</td>
<td>• Easy to define roles and permissions</td>
<td>• Complex infrastructure</td>
</tr>
<tr>
<td>• Weakest access control system</td>
<td></td>
<td>• User-Role-Permission combinations may go out of control for large organizations</td>
<td>• Extensive flexibility, granularity</td>
</tr>
<tr>
<td>• Ideal for small teams, companies</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf](https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf)
ABAC - I

- Multidimensional characteristic
- ABAC entities: Subject, object, operation, environmental conditions, policy
- ABAC makes decision by combining all entities
- Higher number of discrete inputs results larger set of possible combinations; makes ABAC more flexible
- ABAC Result: Access, Deny, Not Applicable, Indeterminate

https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf
ABAC - II

**PROS**

- **Fine-Grained, Attribute-Based**: Attributes are building blocks
- **Externalized Authorization**: Decoupled business logic
- **Centralized Authorization**: Easy deployment, easy maintenance, reduced cost, large hardware capacity (on cloud)
- **Policy-Driven Authorization**: Configurable policy definitions

**CONS**

- **Doesn’t fit to IoT**: Distributed environments with several different domains, collaborating with each other
- **Context awareness**: Surrounded with set of logical functions
### Extending ABAC Capabilities

<table>
<thead>
<tr>
<th>Semantic Tech &amp; Ontology</th>
<th>S-ABAC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Technology:</strong></td>
<td>S-ABAC = ABAC + Semantic Technologies</td>
</tr>
<tr>
<td>● Machine readable to machine understandable</td>
<td>● An ontology management system</td>
</tr>
<tr>
<td>● IoT example: smart sensor sending information to any platform</td>
<td>○ provides the extended user and resource attributes</td>
</tr>
<tr>
<td>● RDF vs XML</td>
<td>● An access control system</td>
</tr>
<tr>
<td><strong>Ontology:</strong></td>
<td>○ uses the extended attributes for access evaluation[1]</td>
</tr>
<tr>
<td>● Providing a shared understanding of common domains</td>
<td></td>
</tr>
<tr>
<td>● Hospital domain: departments, clinics, staff</td>
<td></td>
</tr>
<tr>
<td>○ Staff: Technical, administrative, operational</td>
<td></td>
</tr>
<tr>
<td>● OWL</td>
<td></td>
</tr>
</tbody>
</table>

### Access Control Comparison

<table>
<thead>
<tr>
<th></th>
<th>DAC</th>
<th>MAC</th>
<th>RBAC</th>
<th>ABAC</th>
<th>S-ABAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>Simple</td>
<td>Simple-Medium</td>
<td>Medium</td>
<td>Complex</td>
<td>Complex</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Least</td>
<td>Less</td>
<td>More</td>
<td>Most</td>
<td>Most</td>
</tr>
<tr>
<td>Strength</td>
<td>Weakest</td>
<td>Strong</td>
<td>Strong</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Granularity</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Context-awareness</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No-Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Popular IoT Boards

**Raspberry Pi**
- **Microcomputer**
  - CPU: Quad-core 1.5GHz
  - RAM: 2GB, 4GB or 8GB
  - Dimensions: 88mm × 58mm × 19.5mm
  - WiFi, Bluetooth
  - Unix-Debian, community support, convenient development environment

**Arduino**
- **Microcontroller**
  - CPU: Single-core 16MHz
  - Flash Memory: 32 KB
  - Dimensions: 68.6 x 53.4 mm
  - WiFi
  - No Operating system
  - Arduino programming language
Prototype Implementation - Overview

**FIWARE Authzforce ABAC project**
- Java based, open-source project
- OASIS XACML compliant
- Offers RESTful authorization server

**Minimum Hardware Requirements**
- CPU frequency: 2.6 GHz
- RAM: 4GB min
- Disk space: 10 GB min

**Software Requirements:**
- JRE 8 and above
- Tomcat 8.x

**Raspberry Pi Model 4B**
- CPU: Quad-core 1.5GHz
- RAM: 4GB
- Memory: 16GB Micro SD Card
- Display: Mini-HDMI Port
- USB Ports: Allowing to use keyboard and mouse

Turns to be an ideal development platform
Prototype Implementation - I

High Level Architecture

Flow Diagram

1. Doctor wants to access patient’s smart door lock within work hour if there is an emergency
2. Can doctor access the lock?
3. Intersect and communicate with Semantic Reasoner
4. Ask Ontology if attributes have synonyms
5. Enrich the request with attribute synonyms
6. Evaluate polices
7. Yes, grant access
8. Access patient’s smart door lock

Request Preprocessor Extension (RPE)

Start

Send XACML Request to ABAC

Intercept the request at RPE

Request has required attributes?

Send request to Semantic Reasoner

Ontology response has context

Yes

Enrich request with received attribute synonyms

Yes

Send final decision (Deny or Permit) to user

No

Forward request to PDP for decision

Ontology response

No

Enrich request with received attribute synonyms

No

Reply with ontology decision

End
Prototype Implementation - II

ABAC Engine Setup on Raspberry Pi 4

Install Tomcat and Java
> $ sudo apt install openjdk-8-jre
> $ sudo apt install tomcat8

Download and Instal ABAC
> $ wget
https://repo1.maven.org/maven2/org/ow2/authzforce/authzforce-ce-server-dist/8.1.0/authzforce-ce-server-dist-8.1.0.deb
> $ sudo aptitude install gdebi curl
> $ sudo gdebi authzforce-ce-server-dist-M.m.P.deb

Test ABAC and Deploy Policy
> $ curl http://localhost:8080/authzforce-ce/?_wadl
> curl -X POST \
http://localhost:8080/authzforce-ce/domains/{domain-id}/pap/policies \ -H 'Content-Type: application/xml' \
'<?xml version='1.0' encoding='UTF-8'?>
<PolicySet>...etc</PolicySet>'
Policy

Doctor can access patients’ smart door lock in emergency state during working hours environmental condition

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ns5:PolicySet>
      <ns5:Target/>
      <ns5:RuleId>urn:oasis:names:tc:xacml:3.0:example:MyRule</ns5:RuleId>
      <ns5:Effect>Permit</ns5:Effect>
      <ns5:Target/>
      <ns5:Attribute>subject</ns5:Attribute>
      <ns5:Attribute>allow</ns5:Attribute>
      <ns5:Match>
        <ns5:MatchId>
          <ns5x:AttributeValue>urn:oasis:names:tc:xacml:1.0:attribute-category:string EQUAL</ns5x:AttributeValue>
          <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">doorLock</ns5x:AttributeValue>
          <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">MustBePresent=False</ns5x:AttributeValue>
        </ns5:Match>
      </ns5:Match>
    </ns5:Rule>
    <ns5:Rule Id="urn:oasis:names:tc:xacml:1.0:rule:and">
      <ns5:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-and">
        <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">doctor</ns5x:AttributeValue>
        <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">MustBePresent=False</ns5x:AttributeValue>
      </ns5:Apply>
      <ns5:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">doctor</ns5x:AttributeValue>
        <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">MustBePresent=False</ns5x:AttributeValue>
      </ns5:Apply>
    </ns5:Rule>
    <ns5:Rule Id="urn:oasis:names:tc:xacml:1.0:rule:deny">
      <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">deny</ns5x:AttributeValue>
      <ns5x:AttributeValue>DataType="http://www.w3.org/2001/XMLSchema#string">MustBePresent=False</ns5x:AttributeValue>
    </ns5:Rule>
  </ns5:PolicySet>
</ns5:Policy>
Prototype Implementation - V

PDP Request Preprocessor Extension

- PDP Extension Plugin to implement non-core PDP behaviour
- Java based plugin, deployed on ABAC engine
- Intercept incoming request before evaluation on PDP
- Modify if necessary and resume the flow
- REST Client to communicate with Semantic Reasoner

Goals:
- Manage integration bw ABAC-Semantic Reasoner
- Manage request modification
Prototype Implementation - III

Hospital Staff Ontology

- Nurse
- Physician
- Doctor
- Legge
- Patient
- Hospital
- Staff
- Cardiologist
- Department
- Department - Cardiology
- Access - Delete
- Access - Read
- Access - Write
- Review
- Read
- Access
- Write
- Change
Semantic Reasoner Tool

- A tool to parse relationships in ontology and extract synonyms
- Java, Spring boot, OWLApi, Hermit Reasoner, deployed on Tomcat web-server
- Tool offers endpoints:
  - /query-by-role/{roleName}
  - /query-by-access-type/{accessTypeName}
  - /query-by-access-type/{accessTypeName}/for/{roleName}
- Subject synonyms: Doctor, Lege, Physician
- Access type synonyms: Access, Read, Review

Goals:
- Integrate with ABAC and offer API
- Parse the ontology and extract knowledge
- Forward back the knowledge
Evaluation

1. Outcome
   Decentralized S-ABAC with ontology on fog node

1. Performance

<table>
<thead>
<tr>
<th>Load Tests (ms)</th>
<th>Cloud</th>
<th>RPi-4B</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAC</td>
<td>52.36</td>
<td>1537.68</td>
</tr>
<tr>
<td>S-ABAC</td>
<td>69.05</td>
<td>1713.14</td>
</tr>
</tbody>
</table>

1. Cost
   - RPi Model 4B = 729kr
   - Mini-HDMI Adapter = 95kr
   - Total = 824kr

2. Limitation
   - Mock requests for object (smart door lock)
   - Basic ontology works perfect on RPi 4B
   - Further tests can be apply on more inclusive ontology
Discussion

Discussion #1
Function Extension vs Request
Preprocessor Extension

Discussion #2
Cloud ABAC is the trend due to complexity

Discussion #3
Centralized approach advantages and disadvantages

Discussion #4
Cloud (S-)ABAC vs GDPR

Discussion #5
Decentralized S-ABAC
Future Work

Suggestion #1
Synonym repository or dataset is the key for not having very long policy requests

Suggestion #2
PDP Response Postprocessor Extension for future analytics policy responses.

Suggestion #3
Turn the setup into microservices by using docker container for each component.
Back to Research Questions & Objectives

Objective #1 : ABAC on fog-area

Objective #2 : Semantic-ABAC

Objective #3 : Decentralized Access Control

RQ #1: What is the importance of the fine-grained access control for IoT settings?

RQ #2: Can ABAC work for simple IoT devices on the fog-area?

RQ #3: What is the ideal gateway device for this need?

RQ #4: How can we link an ABAC engine to a Semantic Reasoner having a simple ontology?
Conclusion

S-ABAC is a successor of other access control systems, including ABAC.

S-ABAC on fog-area IS POSSIBLE without losing performance.
Thank you for your attention.

QUESTIONS
THANK YOU!

Semantic - Attribute Based Access Control
Master Thesis

Uğur Bayram - ugurb@ifi.uio.no
University of Oslo
Master of Science (MSc) - 30 ECTS
Supervisors:
Christian Johansen, Hamed Arshad