Abstraction

Nowadays, NoSQL databases have been rapidly becoming the popular data platform for big data and real-time web applications. Simpler horizontal scaling, flexible schema designing, high performance data access have made NoSQL databases to be alternative approaches for traditional relational databases . However, there are some disadvantages in NoSQL, among which the lack of effective suppprt for access control and privacy protection is the most serious ones. The huger data we have, the more challenge in data protection we have to face. In this thesis, we address this issue by implemeting a comprehensive framework for enforcing attribute-based security policies stored in JSON document. We use Polish notation for modeling conditional expressions which are the combination form of subject, resource, and environment attributes so that the policies are flexible, dynamic and fine grained. Moreover, with the approach of attribute-based access control, we have proposed a flexible model struture for privacy protection so that it can be evaluated not only by access purpose but also by subject, resource, environment attributes. We also build a web application which interacts to our framework so that administrators can easily define and review policies. The experiment is carried out to illustrate the relationship between the processing time for access decision and the complexity of policies.

1. Overview
   1. Introduction

Nowadays, the quanity of data is increasing exponentially by the development of social media appications, sensor for data acquisitions and smart phone utilization. NoSQL databases is the most popular approach to handle those semi and unstructured data for a scalable application. Like relational database, security is highly considered in NoSQL database, especially when working with huge volume data. For the last decade, Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC) have been used almostly to handle security. However, because of the rapid development of large scale dynamic systems, those traditional access controls have gradually reveal their disadvantages, for example, applied for only closed system, role explosion and inflexibility in specifying dynamic policies and contextual conditions. To overcome those problems, Attribute Based Access Control models have been recently investigated and according to Gartner‘s prediction: “By 2020, 70% of all businesses will use attribute based access control as the dominant mechanism to protect critical assets.”[]

Based on the Hibert and Lopez assessment of global information storage over time, Figure[] shows the relative between the increasement of stored information from 1986 to 2007 and the trend of access control. Even when access control systems are successful in restricting unauthorized and unauthenticated users, they are ineffective as privacy protection for a large, decentralized system like the World Wide Web, where it is easy to copy or aggregate information. Most previous studies have considered privacy protection in access control models as constraints on purpose of data usage. ?



* 1. Related work

Hua Wang, Lili Sun, and Vijay Varadharajan[] propose a purpose-based framework for supporting privacy preserving access control policies and mechanisms. They have also developed algorithms to help a system to detect and analyze the conflicts when adding new policies.

Prosunjit Biswas, Ravi Sandhu, and Ram Krishman[] present an attribute based protection model for JSON documents. Their approach is to add a new attibute called “security-label” to JSON elements and specify access control policies using these values.

CODASPY11

* 1. Purpose and Scope

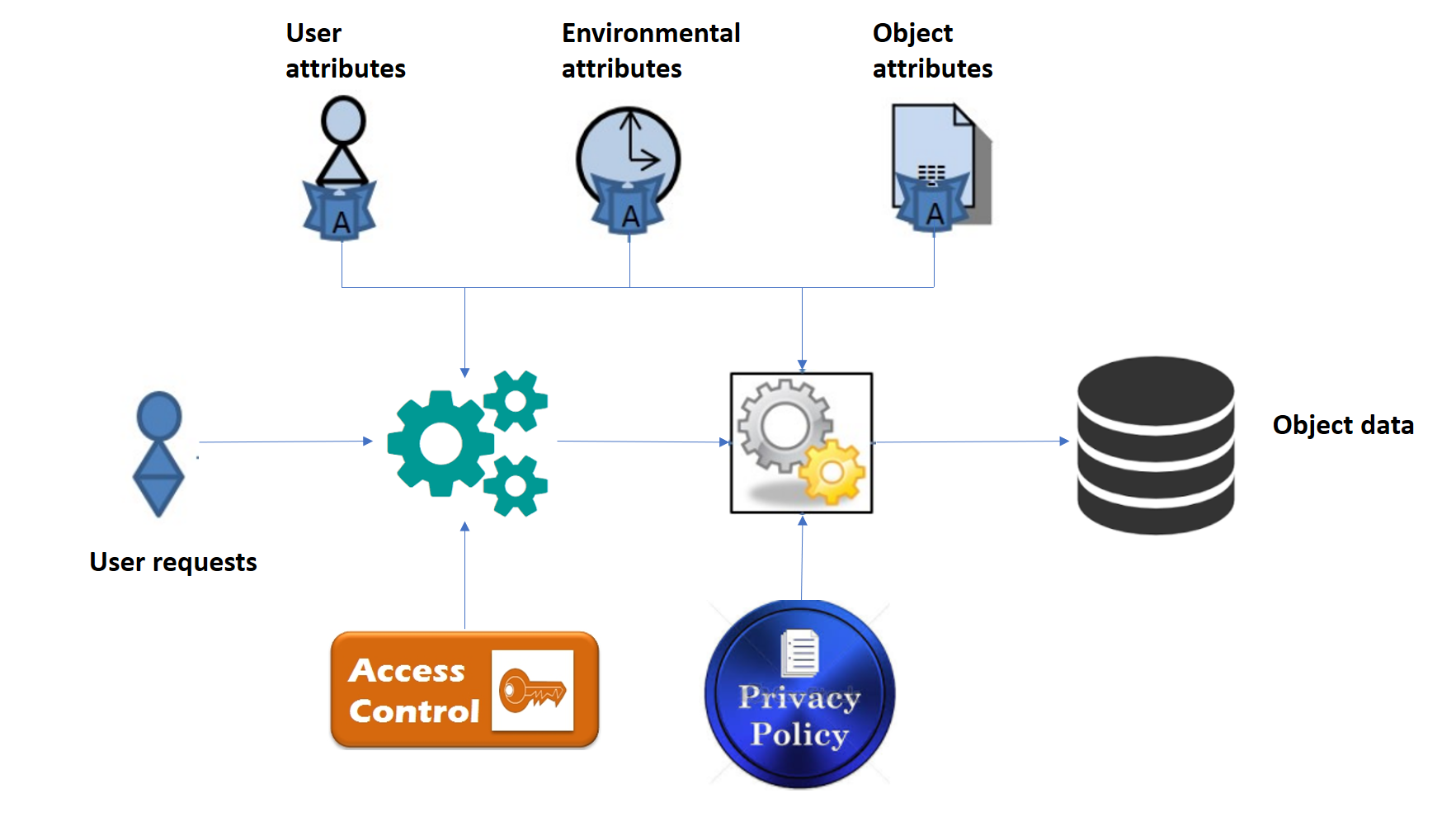
In this thesis, our access control model is built on the principle of NIST Standard ABAC that an access decision is permitted only if the request satisfies conditions on attributes of subject, resource and environment specified in policies. Moreover, with the approach of attribute-based access control, we have proposed a flexible model struture for privacy protection so that it can be evaluated not only by access purpose but also by subject, resource, environment attributes. We use Polish notation for modeling conditional expressions so as to describe complex policies such as user, data, environment driven policies.

* 1. Thesis Structure

1. Background:

In this section, we describe the base theory of this thesis. When a subject access an object, the authorization process is carried out though two stages called as 2-stage authorization:

* First stage: access control policy authorization verifies that the request is legitimate with rights for the subject to access data.
* Second stage: request is transfer to this stage for checking privacy compliance based on privacy policies.



Figure[]

**Access Control Policies**: contain policies which are used to determine whether a subject can access resources. The decision is made based on rules inside policies which are the boolean expressions evaluated by user’s defined function, subject, resource, environment attribute. Those policies are specified and managed by administrators.

**Privacy Policies**: contain policies which are used to determine whether some fields of a record in resources should be shown, hidden or blurred when a user access to it. The privacy protection is made based on rules inside policies which are the boolean expressions evaluated by user’s defined function, subject, resource, environment attribute. Those policies are specified and managed by administrators.

1. Policy Structure:
   1. General Structure:

Class Diagram[]

* 1. Access Control Policy Structure:

We specify access control policy structure as follows:

|  |  |
| --- | --- |
| **Fields** | **Description** |
| policy\_id | identifier of policy |
| collection\_name | name of collection or table containing resource data |
| action | the action performed by subject |
| rule\_combining | to solve the conflict of rules |
| is\_attribute\_resource\_required | a derived field which is used to determine that whether the policy need attribute resource to evaluate condition of target or rules. The necessary of this field will be mentioned in the next section. |
| target |  |
| rules | an array field with each element in it is a rule which contains “id” field, “effect” field (value of this field can only be “Permit” or “Deny”) and condition. |

Example:

* 1. Privacy Policy Structure:

We specify access control policy structure as follows:

|  |  |
| --- | --- |
| **Fields** | **Description** |
| policy\_id | identifier of policy |
| collection\_name | name of collection or table containing resource data |
| is\_attribute\_resource\_required | a derived field which is used to determine that whether the policy need attribute resource to evaluate condition of target or rules. The necessary of this field will be mentioned in the next section. |
| target |  |
| rules | an array field with each element in it is a rule which contains “id” field, “field\_effects” field and condition. “field\_effects” field is an array field with each element specifies which privacy function will be used with the value of “name” field when condition is satisfied. |

* 1. Conflict resolving approach:

Because an access control policy may contain multiple rules and we have many policies. Each rule, access control policy may evaluate to different decisions (Permit, Deny). Therefore our approach is to use combining rule algorithm inherited from XACML[].

A new structure named “AccessControlPolicyCombiningConfiguration” is added to resolve conflict between multiple access control policies.

|  |  |
| --- | --- |
| **Fields** | **Description** |
| policies\_id | A list identifiers of policies |
| algorithm | The name of algorithm is used to solve conflict when multiple policies are contained in “policies\_id” field |

Example:

In privacy policies, the conflict situation occurred when there are multiple rules in a privacy policy which are satisfied the condition. It results to that we have many privacy functions to be applied to one field of object. Therefore we have added a new structure named “PrivacyDomain” to solve conflict. We also specify a constraint that a field in resource can only belong to at most two domains, one is default domain which contains two basic privacy functions: show and hide, another domain is configured by administrator.

|  |  |
| --- | --- |
| **Fields** | **Description** |
| domain\_name | Name of domain. |
| fields | The names of fields in resource which are belong to this domain. |
| hierarchy | To configurate the priority for each privacy function or sub-privacy policy. |
| is\_sub\_policy | To check whether this is domain for privacy function or sub-privacy policy. |

Example:

{

"domain\_name" : "SsnDomain",

"fields" : ["Student.personal\_info.ssn"],

"is\_sub\_policy" : false,

"hierarchy" : [

{

"name" : "AreaNumber",

"priority" : 1

},

{

"name" : "GroupNumber",

"priority" : 2

},

{

"name" : "SerialNumber",

"priority" : 3

}

]

},

{

"domain\_name" : "DefaultDomain ",

"fields" : [],

"is\_sub\_policy" : false,

"hierarchy" : [

{

"name" : "Hide",

"priority" : 1

},

{

"name" : "Show",

"priority" : 2

}

]

},

{

"domain\_name" : "DateTimeDomain",

"fields" : ["Student.personal\_info.birth\_date"],

"is\_sub\_policy" : false,

"hierarchy" : [

{

"name" : "ShowYear",

"priority" : 1

},

{

"name" : "ShowMonthAndYear",

"priority" : 2

}

]

}

Let consider below example to see how conflict resolving process work:

Student Resource:

{

"name": "John",

"personal\_info": {

"birth\_date": "15/01/1994",

"ssn": "457-55-5462"

}

}

Privacy policy:

{

"policy\_id": "policy 1",

….

"rules" : [

{

"rule\_id" : "rule 1",

"field\_effects" : [

{

"name" : "name",

"effect\_function" : "Optional"

},

{

"name" : "personal\_info.birth\_date",

"effect\_function" : "DateTimeDomain.ShowYear"

},

{

"name" : "personal\_info.ssn",

"effect\_function" : "SsnDomain.SerialNumber"

}

],

"condition" : {//assume that this condition is satisfied}

},

{

"rule\_id" : "rule 2",

"field\_effects" : [

{

"name" : "name",

"effect\_function" : "DefaultDomain.Show"

},

{

"name" : "personal\_info.birth\_date",

"effect\_function" : "DateTimeDomain.ShowMonthAndYear"

},

{

"name" : "personal\_info.ssn",

"effect\_function" : "SsnDomain.AreaNumber"

}

],

"condition" : {//assume that this condition is satisfied }

},

{

"rule\_id" : "rule 3",

"field\_effects" : [

{

"name" : "name",

"effect\_function" : "DefaultDomain.Show"

},

{

"name" : "personal\_info.birth\_date",

"effect\_function" : "DefaultDomain.Show"

},

{

"name" : "personal\_info.ssn",

"effect\_function" : "Optional"

}

],

"condition" : {//assume that this condition is satisfied }

}

]

}

First we will explain more detail about the “field\_effects” field in privacy policy structure. It is an array field and the number of elements in this field is equal to the number of single value field in resource. Each element has the following structure:

"name": is the path to the single value field.

"effect\_function": This field has only 2 value patterns. First is "Optional" value, second is "X.Y" value where X is privacy domain, and Y is name of privacy function in that domain.

Back to the example, we have the conflict table now.

|  |  |
| --- | --- |
| **Fields** | **Conflict Privacy Functions** |
| name | Optional, DefaultDomain.Show |
| personal\_info.birth\_date | DateTimeDomain.ShowMonthAndYear,  DateTimeDomain.ShowYear,  DefaultDomainPrivacy.Show |
| personal\_info.ssn | SsnDomain.AreaNumber, SsnDomain.SerialNumber, Optional |

The privacy function will be chosen using the following rule:

P(“Optional”) < P(“DefaultDomain.Show”) < P(X.Y1) < … < P(X.Yn) < P (“DefaultDomain.Hide”)

where P(X.Y) stands for priority of privacy function Y in domain X. The priority is configured by admininstrator in “PrivacyDomain” structure.

Applying this rule to above conflict table, we will have the following result:

|  |  |
| --- | --- |
| **Fields** | **Privacy Function Chosen** |
| name | DefaultDomain.Show |
| personal\_info.birth\_date | DateTimeDomain.ShowYear |
| personal\_info.ssn | SsnDomain.AreaNumber. |

Applying those privacy function chosen, we will the returned data:

{

"name": "John",

"personal\_info": {

"birth\_date": "1994",

"ssn": "457"

}

}

* 1. Advanced privacy policy support for muti value field:

continue

<https://www.avatier.com/products/identity-management/resources/gartner-iam-2020-predictions/>

https://www.tripwire.com/state-of-security/security-data-protection/security-controls/rbac-is-dead-now-what/