8-2017

Designing Secure Access Control Model in Cyber Social Networks

Katanosh Morovat
University of Arkansas, Fayetteville

Designing Secure Access Control Model in Cyber Social Networks

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

by

Katanosh Morovat
Isfahan University of Technology
Bachelor of Applied Mathematics in Computer Science, 1992
Azad University
Master of Software Engineering, 1998
University of Arkansas
Master of Science in Computer Science, 2015

August 2017
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

_____
Dr. Brajendra Panda
Thesis Director

_____
Dr. Wing-Ning Li
Committee Member

_____
Dr. Dale Thompson
Committee Member

_____
Dr. Paul Cronan
Committee Member

Abstract

Nowadays, information security in online communication has become an indisputable topic. People prefer pursuing their connection and public relations due to the greater flexibility and affordability of online communication. Recently, organizations have established online networking sites concerned with sharing assets among their employees. As more people engage in social network, requirements for protecting information and resources becomes vital. Over the years, many access control methods have been proposed. Although these methods cover various information security aspects, they have not provided an appropriate approach for securing information within distributed online networking sites. Moreover, none of the previous research provides an access control method in case an existing resource encompassing various parts and each part has its own accessing control policy.

In this research, we investigate the access control requirements in order to conserve data and encompassed resources, which are shared in the social network, from users with unapproved access. Under the proposed method, users are able to define policies easily to protect their individual information and resources from unauthorized users. In addition, requestors are able to generate inquiries in easy and efficient way. We define an appropriate format to present rules and queries, which are converted from policies and inquiries respectively. The proposed approach defines a method in case a user would like to access a resource belonging to another user where both users are members of different online networking sites. In order to add more flexibility, this method controls access to data and resources by evaluating requestor's attributes, object's attributes, action or operation taken by requestor, environmental condition, and policies which are created by users or a super user of social network to protect the users' resources. This approach is called Policy-Based Attribute Access Control (PBAAC). The policies defined to secure a resource may conflict with other policies. The proposed method offers an appropriate

solution to resolve this issue. Due to achievement of better performance with regards to efficiency, this research analyzes the method to compromise simple rules, complex rules, or rules including several attributes. The results prove that simple rules provide better performance.

Acknowledgments

I wish to express sincere appreciation to the many individuals who have offered support, inspiration, and encouragement throughout this research effort.

First and foremost, I have to thank my research supervisor, Dr. Brajendra Panda. Without his assistance and dedicated involvement in every step throughout the process, this paper would have never been completed. I would like to thank you very much for your support and understanding. I would also like to show gratitude to my committee, including Dr. Wing-Ning Li, Dr. Dale Thompson, and Dr. Paul Cronan for their lectures and academic advice during my study. My thanks and appreciations also go to Dr. Craig Thompson in his lectures and academic advice during my study and financial support in more than a year of this degree. I would like to extend my sincerest thanks and appreciation to Dr. Patricia Koski and Mr. Mark Larmoyeux who supported me financially and encouraged me without which this study would never have accomplished. Special gratitude is extended to Dr. Joseph Gergerich who helped me, encouraged me and accompanied me in every step of my study. His counsel and assistant were invaluable.

Most importantly, none of this could have happened without my family. I express my gratitude to my parents, my sister, and my brother for their continuous support, encouragement and belief in me. Last but not least, I thank the University of Arkansas for accepting me and offering me an excellent Ph.D. course of studies. This dissertation is dedicated to you, my parents, my sister, and my brother who have made me who I am. Thank you for friendship, time, leadership, guidance, care, and primarily, thank you for your trust.

Contents

List of Figures

List of Tables

1. Introduction

As the number of users of information systems have increased significantly in recent years, the challenge of controlling access to resources has become a hot topic in computer science. Information security protects data and assets from deleterious users in social networks. Lack of appropriate control over the data might incur several security issues. According to U.S law [1], Information security is a process for

"P*rotecting information and information systems from unauthorized access, use, disclosure, disruption, modification, and destruction*".

This means we require a method for protecting our data from malicious users. Furthermore, in a general sense, security means protecting our resources from unmanaged access. Information security minimizes the risk of uncovering information of unauthorized users. Information security is not a technology; instead, it is a strategy comprised of the processes, tools, and policies to protect digital data and resources. Information security concepts become more ubiquitous and enmesh in many aspects of computing technology.

Today's life is mixed with computing technology and using a computer is ubiquitous in various fields, such as taking an online training course, transferring money, and buying goods from merchandise, which are an unavoidable part of our life. Although this technology empowers us to access information easily in a short time, it carries several security issues. On the other hand, technology changes at an increasingly rapid rate and imposes upon us to define new security method to keep up with the changes in our technology. In order to find effective methods to protect our information, good understanding of the basic of information security is a key point. In our effort to protect our resources, we require contemplating the consequences of the protection method we choose to implement. Hickey [2] said

"*The only truly secure system is one that is powered off, cast in a block in concrete and sealed in a lead-lines room with armed guards – and even then I have my doubts*".

If we define a method to protect a system and claim that a system might be regarded reasonably secure, there is no doubt that the system is not functioning very well. As we increase the level of security, we usually lose the level of performance. Hence, it is critical to maintain a balance between strict security and high performance of a system.

Three primary properties of information security are confidentiality, integrity, and availability. These are also known as the CIA triad, as shown as Figure 1, which is a model designed to define policies for information security within an organization [3].

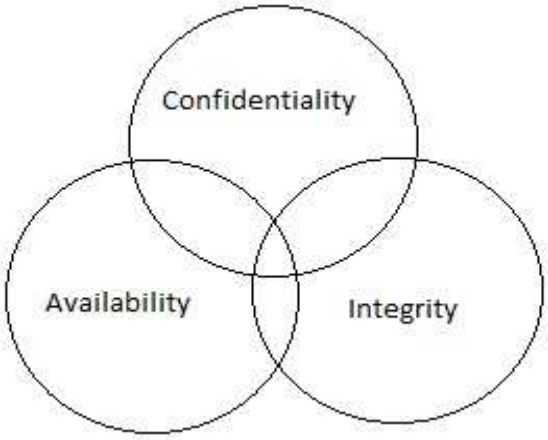

Figure 1- CIA Triad

- Confidentiality: this is a required component of privacy and refers to the protection of data and resources from those who are not allowed to view that data and resources. Confidentiality defines a set of rules that confine access to data or resources.
- Integrity: this is a required component for preventing data from being altered in an unauthorized or unmanaged manner. Integrity associated with maintaining the consistency, accuracy, and trustworthiness of data during its life cycle [4].

- Availability: this is a required component for the ability to access data and resources when they are needed.

1.1.Access control

Access control is an information security technique that is used to determine who, and under which constraints, can access or perform an operation against an object in a computing environment. Access control, which is part of confidentiality, determines what a user will grant other users to access or perform an operation on a resource or an object mediated by the former user [5]. The definition provided by the National Institute of Standards and Technology to describe the access control mechanism, is as follows [1]

"The logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision "

There are two main types of access control, physical and logical [3]. Under our approach, only logical access control will be mentioned, which limits connections to computer networks and accessing data.

Generally, access control manages each user's ability to read, execute, change, or delete information associated with a particular resource. In effect, access control works in two steps: first, either grant or deny interaction with a resource, and second, control the types of operations or activities that may be performed on that resource. Such controls are managed by an access control system. The access control system may be part of a more general resource management and control environment [9]. Today, there are numerous methods of access controls implemented or practiced in real-world settings. These methods are described in the rest of this chapter.

Conventionally, access control is based on the identity of the user requesting to perform an operation on an object either directly or through predefined roles or groups assigned to the user. As per Sandhu's definition [5], authentication, access control, and audit, all together establish

information security. Authentication checks the identity of a user wishing to access the information system. Access activities in the information system analyzes data to find any security violation [5]. Access controls are applied after authentication which was settled [6]. Audit assists in detection of fraud attempt or actual security cracks by recording details of events which are relevant to security [8].

In conclusion, access control is a fundamental part of any information security system. The several main methods of access control systems are: Discretionary Access Control (DAC), Lattice-based access control or Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Policy-Based Access Control (PBAC), and Attribute-Based Access Control (ABAC). It seems that these approaches to access control are cumbersome to govern flexible capabilities directly associated to users. Furthermore, due to lack of appropriate features to express real-world policies and inquiries users often suffered from semi-dynamic access control methods. One solution might empower user to define policies and inquiries precisely by using a natural language and accept or reject user's inquiry in terms of arbitrary attributes of the user, attributes of the object, the rules defined by the owner of the object to protect the object, and environment conditions which are changed based on the time and location and are more relevant to the policies. This approach will be called Policy-Based Attribute Access Control (PBAAC).

1.2. Attribute-Based Access Control

For the first time, Attribute-Based Access Control (ABAC), which was defined by the National Institute of Standards and Technology (NIST), improves data sharing while data are supervised [10]. NIST defines ABAC as:

"An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object,

environment conditions, and a set of policies that are specified in terms of those attributes and conditions". [10]

ABAC is a logical access control model and it checks access to an object based upon the attributes of entities, operations, and environmental elements pertain to the request. As suggested by its name, ABAC is an attribute-centric access control where attributes control what resources a user can access [7]. ABAC can accept a large number of inquiries and evaluate them by using an access control decision engine. Access control decision engines provide a large set of combinations of attributes to represent a large set of predefined rules expressing policies which are confined by the computational language. Under ABAC, access control decision engines may be shared between inquiries by changing attribute values; there is no need to change the subject/object relationships defined in rule sets. This feature adds more robust access control management capabilities and minimizes continuous maintenance requirements of object preservation. Moreover, ABAC empowers object owners to employ access control policy without prior knowledge of the particular subject. As a new subject joins the organization, rules and objects do not require being altered and as long as the subject is assigned attributes required for access to a specific object, no alteration to existing rules or object attributes are required [6]. This is one of the advantages of applying ABAC method. Although ABAC is a robust access control method, we need to define a method in order to be able to control an access to objects by evaluating operations, environmental conditions relevant to a request, and rules which have been stored in the repository of the user. So, another method would be needed to cooperate with ABAC to satisfy this requirement. Policy-Based Access Control combines attributes associated to entities with information of sets of rules that specify whether or not access is allowed.

1.3. Policy-Based Access Control

Policy-Based Access Control (PBAC) is an access control method which enables social networks to govern their resources by defining policies. Social networks define various policies and administration structures to ensure the successful execution of social network's mission and fulfill responsibility and adaptation with relevant law and regulation. Furthermore, they need to create policies that define who should have access to which resources and under what circumstances. The PBAC is a method for controlling user access to one or more resources, while each resource is owned by someone else and is assigned one or more policies to determine what access is either granted or denied. The PBAC implements actual access controls based on abstract central policy and managed requirements. Policies are defined and saved in the system and they are not changed very quickly. It means that the rate of changing policies is rare.

The PBAC is assumed to be a standard version of the ABAC model for supporting particular governance of objectives. The PBAC acquires attributes associated to the resource, the requester, and the environment with information of the set of conditions under which the access request is made. In addition, this model uses a set of rules that specify whether or not the access is allowed. In the ABAC model, attributes, which are defined to have access to a particular resource, are designed on a local level and might be different from one social network to another social network. In the PBAC model, the social network can define one policy governing access to all resources that meet particular criteria, and this policy might be enforced for all attempts to access the resource, no matter where resources are hosted on the social network. The PBAC can be thought as a much more complicated model, and there is a need of a mechanism to define policy in an unambiguous term. It is vital that policies must be clear and unambiguous; otherwise, there is potential for an unauthorized access to a resource with which a specified policy is associated.

There is also required to ensure that PBAC creates attributes by using the way that ABAC uses, and uses these attributes for accessing resources.

1.4. Proposed Model

Recently, people are choosing to pursue their connection and public relations due to the greater flexibility and affordability of online communication. As more people engage in online communication through social network sites, requirement for protecting information and resources becomes essential. In this research, we investigate the access control requirements to conserve data shared in social network sites or organizations from disapproved access. In pursuance of achieving the goal, a new method has been defined enabling users to define policies easily in order to protect their individual information and resources from unauthorized users. Moreover, it allows requestors to generate inquiry in an easy and an efficient way.

This method controls access to an object by evaluating requestor's attributes, object's attributes, an action or operation taken by a requestor (such as read, write, or alter), environment conditions, and policies which are created by users or the super user of social network sites to protect the object. This approach is an appropriate method to enforce access control in a dynamic way and is called Policy-Based Attribute Access Control (PBAAC). The PBAAC uses ABAC and PBAC models. Under this proposed model and unlike the ABAC model, users are able to define attributes arbitrarily; no pre-defined list of attributes exist in this model. Furthermore, unlike the PBAC, this model empowers users to define policies to protect their own resources; no pre-defined policies exist in this model and rate of adding new policies may be high. We also explore the access control requirements of an object in case the object is composed of other objects. This model offers an appropriate solution in case that several policies conflict with each other. Furthermore, this model can be extended. The extended model includes various social

network sites which are able to share their data and assets, whilst these social network sites are not connected to each other directly. One specific social network, which is called supervision, acts as an intermediary among others.

Since data of social network sites are stored on cloud servers, there are many security requirements such as controlling user authentication and establishment of secure communication among servers. These are out of our research scope. In this method, we just focus on access control requirements for social networks.

In the rest of this document, basic concepts, terminologies and entities that exist in this method will be explained. Finally, for the PBAAC model and the extended model, basic notations and formulations of how policy language will be defined will be described.

1.5. Contribution of this Dissertation

Under this dissertation the following contributions are accomplished:

- The PBAAC model built based upon collaboration of ABAC model enables users to define attributes arbitrarily. There is no pre-defined list of attributes for this model.
- The PBAAC model enables users to define various policies to govern their resources and put inquiries in order to access or carry out an operation on a resource in the most effective way.
- The PBAAC is an access control model built based on the authorization decision. This model enforces finer-grained access control in both a central and extended systems.
    - The basic or fundamental PBAAC model is designed for one social network site or one organization. In order to sufficiently share resources and enforce access control, the PBAAC model uses a specific data model, which captures data in a directed graph format. This format allows efficient way to extract information through a graph search.

- The enterprise PBAAC model is designed for a homogeneous collection of either social network sites or organizations. Beside the collection of social network sites, this model includes a supervision to make a connection between these social network sites. Having an effective way for sharing resources and enforcing access control, the enterprise PBAAC model employs several independent graph data models in which one graph contributes to other graphs through the supervision graph.

- Due to using natural language to generate policies and inquiries, the PBAAC model utilizes a language specification convertor to transform policy and inquiry to a well-formed expression. This expression follows the format like: (subject, action, object, condition) to present rules and requests. By adoption of this definition, rules might be a nested expression which means the subject and the object could be a well-formed expression also.
- The PBAAC model offers a technique to enforce access control for an object when it is a compounding of other objects. Each comprised object has its own rules to manage the access.
- The PBAAC model determines a solution, which extracting rules partially fulfilling the received request but all extracted rules may not exist together and all may not be true together. The proposed model uses three criteria to choose a rule that fulfills the request.
- The PBAAC model utilizes rules and attributes to accept or reject a request. This model defines a compromise between having more simple rules, while every rule addresses an entity directly by using the name of the entity and having fewer rules while every rule addresses the entity by using several attributes of entity. Data analysis proves that higher numbers of basic rules improve or speed up the performance of the system.

1.6. Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 offers some background and related works. Chapter 3 introduces a basic or fundamental PBAAC model that serves as the foundation of policy-based attribute access control model. This chapter explains the conversion of natural language to a well-formed format and access control decision engine, and provides a solution in case of existing conflicts among rules. Chapter 4 explains the distributed or enterprise PBAAC model that represents a homogeneous structure of cooperating with several social network sites or organizations. Chapter 5 discusses the evaluation and contribution of the PBAAC model. Chapter 6 presents functions and algorithms used in fundamental and enterprise models. Functions are used for traversing parts of the graph which are related to the request and rules, and extracting information from nodes and relationship among nodes. Algorithms are defined to simulate the proposed model and to determine which parameters played an important role in order to reach a better running time. Chapter 7 provides a discussion of future works and conclusion of this dissertation, and the final chapter is assigned to the bibliography.

2. Background

2.1 Authentication, Access control, Audit

Protection of information in universal infrastructure requires a combined solution from technology, public policy (law), and organizational and individual policies. Advent of various services in the digital world causes the development of effective and flexible secure models and languages. This chapter aims to review various models and languages defined to apply access control mechanism to secure information.

Before defining access control models, investigation of several foundations for information and system security is helpful. Sandhu [5] provided some clear definition for fundamental principle of information and system security. These are authentication, access control, and audit. Authentication provides the identity of a user for using the system, in most cases this reaches by using of password. Authentication might be computer-to-computer, process-to-process, or mutual in both. The most usual authentication is user-to-computer authentication which may be determined based on the following groups:

- Something a user creates and knows, such as password.
- Something a user possesses, such as a smart card or a credit-card-sized cryptographic token.
- Something a user displays in a biometric signature like a voiceprint or a fingerprint.

After access control authentication is applied, access control mechanism determines which party will allow another to do an action against a resource. Access control mechanism requires authentication as a prerequisite. There are several methods for implementing access control mechanism and Sandhu addressed just a few of them [5]. Later in this chapter most of the well-known methods will be investigated.

Finally, audit is a typical way of recording the type of a resource access occurring on a social network. In order to discover any security violation, the audit process collects data of all actions happened in the system and analyzes them. Enormous amount of audit data might be recorded. Because of having lots of auditing places on a system, it is thoughtful to pick and choose which activity types require recording, based upon the organization's security policy [81]. Once any violation is suspected, analysis of the data is performed. Audit has two components such as the collection and organization of audit data [5].

The purpose of defining access control mechanism is to confine the actions or operations that an authorized user of a computer system can perform. Sandhu et al. [47] explained access control mechanism and its relationship to other security models such as authentication, auditing, and administration. Access control cooperates with other security services in a computer system. It is enforced by a reference monitor that intervenes in every request generated by users and objects in the system. The reference monitor uses authorization database to determine if the user who attempts to do an action is authorized to run that action. In this model users may be allowed to alter the authorization database. For instance, they may set permission for their files. Access control mechanism is different from authentication. Access control mechanism assumes that authentication of a user has been proved successfully before to enforcement of access control through reference monitor. Access control mechanism is not a complete solution for securing a system. It must be collaborated with auditing. Auditing controls inductive analysis of all requests and activities of users in the system. Auditing stores all user requests and activities for later analysis.

2.2 Access Control System Evaluation Metrics

In order to evaluate access control methods, Hu et al. [51] defined several metrics. Access control system determines valid activities of legitimate users, filters every attempt to access a resource in an organization performed by users. Access control empowers organizations determining the degree to which their data might be protected and shared among their users. Hu et al. [52] defined properties for quality of access control model in terms of the configurable parameters and limitations of the implemented mechanism. These properties are grouped into four categories as follows:

- Administration: this considers the cost, efficiency, and performance of an access control system.
- Enforcement: this presents mechanisms or algorithms that access control system uses to perform access control models and rules.
- Performance: this considers proficiency of the access control system's processes.
- Support: this presents properties that are not fundamental but that could increase the usability and portability of access control system.

Hu et al. [52] also defined several metrics to be able to compare properties of current configuration or future development of an access control models. As access control policies present the operation requirements of an organization, the metrics evaluate the access control policies operations. Chapter 5 of this document presents metrics that are satisfied by our model.

2.3 Access Control Implementation

The following lists represents methods implementing access control concept.

Discretionary Access Control

Discretionary access control (DAC) is based on the idea that an owner of data determines who is allowed to have access to their resources. DAC allows data to be copied from one resource to another resource freely, so even if the access to the original resource is denied, access to the copied resource is granted.

DAC allows individual to set an access control mechanism to accept or reject access to an object. Hence, object's owner or anyone else who is authorized to access the object is allowed to control the object [55]. Under the DAC model, individuals decide upon the access control policy on their data, regardless of whether or not those policies harmonize with the global policies. Hence, if there is a global policy, the DAC has trouble to ensure consistency in global policies. Moreover, in the DAC model, an owner of an object can easily change the DAC policy, so a malicious program running by the owner can change the DAC policies on behalf of the owner. The following presents several more drawbacks of the DAC model [52].

- Information can be copied from one object to another, so there is no guarantee of controlling the passing information in the system.
- No control defines for the information when a user has received it.
- Having access to an object is defined only by the owner of the object.

Mandatory Access Control

Generally, all access control policies are grouped in the class of non-discretionary access control (NDAC) except the DAC model. As the name implies, policies in this model are rules that are not established at the discretion of the user [52].

In mandatory access control there is a policy for entire organization regulates who is able to have access to an object and individuals cannot alter that access control policy. This means that access control policies are determined by a central authority, not by individual owner of an object, so the owner is not allowed to define or change the access controls. Multilevel security models namely the Bell-La Padula and Biba Integrity models are used to specify the MAC model [52].

List of Access Control – Working Draft

Access Control Lists (ACL) are one of the oldest and most basic forms of access control mechanisms. ACL came into picture in 1970s when multiuser systems require limiting access to data on shared systems. The notion of an ACL is very simple. Each resource, as object, has its own list of mapping between a set of entities that have an access permission to the resource and a set of actions that each entity is allowed to perform on the resource [58]. Although ACL are generally used in the operating systems on a single system, the ACL notion has also implemented in other contexts like network contexts where an access to a target resource might be controlled. Some applications also manage access control lists to determine which users are allowed to view certain data. The advantage of ACL is almost simplicity in implementation. This means that they do not need much complex technological infrastructure to work. On the other hands, ACL have their limitations. The disadvantage of this approach is that ACL cannot manage huge number of users. ACL for specific resources must be checked every time the resource is accessed, and this can be an inefficient access control approach. Moreover, sometimes ACL control applications and system accesses. So if a user wishes to access a file, the file that the user tries to access runs ACL lookup, and the application trying to open the file runs ACL lookup. ACL might be also hard to control an enterprise setting where several people require having

different access levels to several resources. Furthermore, adding, deleting, and changing ACL owned by an individual resource can be time consuming.

Access Control Matrix

Access Control Matrix (ACM) is a conceptual model that determines the rights that each object is gained by a subject. ACM is a two dimensional array in which each row reflects a subject, each column reflects an abject, and each entry is a set of access rights for that subject to that object. This means each cell in the matrix determines the access authorization for the subject in the row to the object in the column. One drawback of using ACM model is wasting memory because generally the ACM is sparse. It means that most subjects are not legible to have access to objects. The ACM model might be presented as a list of triples with form, <subject, right, object>. This method used rarely because searching a huge number of these triples is inefficient [53]. Sandhu et al. [47] audited the ACM model and defined a new approach for implementing the access matrix in practical systems. The access control process might make ensure that only those operations authorized by the access matrix are executed.

Separation of Duty

Under this method no user should take enough privilege to harm the system. Separation of Duties (SOD) can be performed statistically by defining conflicts among roles or dynamically by implementing the control at access time [52]. Hence, under the SOD used in workflow [55] and Role-Based Access Control model, membership in one role might prevent the user from being a member of one or more other roles. There are various types of SOD and an important one is History-Based Access Control Model [52].

Domain and Type Enforcement

This method categorized processes into domains and objects into types, therefore an access is limited from domains to types between domains. Any time, by sending a signal or executing a file in a new domain, a process of one domain is transported to other domains [54]. This method is a static approach to security that supports the fundamental of basic privileges. Tidswell et al. [82] proposed a dynamically configurable of domain and type enforcement. Under this method access control table could be modified in order to configure controls determined in a Rule-Based Access Control. Although this method provides some simplicity and efficiency, updating the access control table is time consuming.

Temporal Constraints

Temporal Constraints are type of access policies that involve time-based limitation on access to resource. In some cases, Temporal Constraints may limit the resource usage and in another case they may control time-sensitive activities [52].

Workflow

Workflow is a presentation of an organizational process in which all resources and tasks are passed from one participant to another; this process is governed by rules or procedures. The goal of the workflow policy is maintaining consistency between the internal data and external expectation of data [57].

Role-Based Access Control

Role-Based Access Control (RBAC) is an access control model. The key concept of the RBAC model is a role. Users are grouped into several roles, although they can be easily denied the roles. A set of permissions assigned to a role and can be modified as needed by the system.

Permissions assigned to roles determine which role is allowed to do which actions against a resource such as access, update, or delete. The roles might be organized in a role hierarchy to represent the organization's authority structure [61].

Based on the RBAC definition, an access to a resource is determined in terms of the relationship between the requester and the organization or owner of the resource. As a matter of fact, the roles of users are defined in terms of their responsibilities on the organization. Hence, the role of the requester will determine whether access will be granted or denied. RBAC allows one person has the same role as others have. RBAC might include many groups of persons and each group categorizes individuals in terms of who fulfills a particular role. This means that one set of access control permissions on a particular resource can be set once for all persons of the organization. In modified RBAC model ideal users are allowed to be a member of multiple groups. This means that users can have several roles at the same time. RBAC supports the permission hierarchies and inheritance while more concise permissions override more general permissions. Despite of many advantages, RBAC has its own disadvantages. One of the highlighted is the fact that grouping people in terms of their roles makes it more difficult to define limited number of accessing controls for each person. It is necessary to create precise set of roles to exclude specific persons who fall into particular role, but don't necessarily need to have the full rights access compare to the other members of the group. Hence, there is required a model to determine attributes for individuals to differentiate members of the group and to selectively grant or deny access based on the set of attributes. Due to fulfill this requirement, Attribute-Based Access Control (ABAC) model was designed.

National Institute of Standard and Technology (NIST) initiated a method for assigning users to roles dynamically to enforce access control a resource [73]. Jin et al. [70] extended a method

and added a function to constrain permissions of a role by using user's attributes. Attributes take certain entities and return values for defined properties of that entity. Each entity (e.g. user and object) has a finite set of attributes. This model defined a permission filtering policy which constrains the available set of permissions in terms of a user and an object attributes. This method did not offer a tradeoff between existing the number of roles and the number of attributes.

Privacy-aware Role-Based Access Control (PRBAC) model defined a method to prevent private data and resource from unauthorized users [71, 72]. This model was extended form of RBAC model and defined by Ferraiolo et al. [73]. This model supports the idea of assigning users to roles and grants permission based on the users' attributes and roles hierarchy. The important point of employing hierarchy is because of easy management of permission assignment. Hierarchy PRBAC provides lots of flexibility but after modeling it was disclosed that specifying all policies in the system needs lots of permission assignments. Arora et al. [74] defined a method to address this issue.

Al-Kahtani et al. [59] defined a method named Rule-Based Role-Based Access Control, for assigning users to roles dynamically in terms of a finite set of rules defined by the enterprise security policy. Users have attributes to save information for identifying themselves. Rules use users' attributes and any constraints set determined by enterprise. Under this model rule might have priority to assign a user to a particular role which means a rule could override other rules. A role might be hierarchically related to one or more roles. Besides, a role could have one or more attributes. In this model role must be defined as initializing step. Their previous work did not provide enough explanation for role relationship and its outcome [67].

Park et al. [62] presented RBAC to enforce access control to resources in a large-scale web environment. They defined a method to implement RBAC on the web. In this method system administrator assigns users to roles in terms of users' responsibilities in the enterprise.

Herzberg et al. [63] presented a trust establishment system that maps users to predefined roles. The system uses a well-defined logical rules to map users to roles. Rules associated to each role define how a client can be assigned to that role. This assignment is computed based on the certificate related to a specific client and the decision is made based on the client's qualification for a specific role. This method uses rules individually and does not pay attention to relation that might be exist among rules.

Yau et al. [64] defined RBAC method using dynamic role activation dependency rather than role hierarchy recognition. A set of parameterized rules activates every role. The strong point of this method is good expressing rules and conditions associated to rules. The weak point of this model is lack of defining role hierarchy which causes capturing various relations existed among roles and confusing to assign a user to a specific role.

Zhong et al. [60] defined a method for using Rule-Based Access Control on the Web and assigning a user to a specific role. This assignment is accomplished in terms of the existing policies and trustworthiness threshold determined by the system administrator. User's trustworthiness identifies the degree which the system believes that a user will not do harm to the system. This degree is accumulated steadily over time and decreased if harmful actions are discovered. Under this method a mischievous user can logon to the system for a long time without performing any suspicious acts, so he gains lots of credits which may enable him to make a tremendous damage on the system. This is a major drawback for this method.

Lightweight Directory Access Protocol (LDAP) manages applications and browsers applications that grant permission for read/write access to directories supporting X.500 protocols [65]. LDAP stores roles in directories and retrieves once as needed. LDAP also supports dynamic groups. A dynamic group is assigned a membership list of distinguished names that is assumed using LDAP searching criteria. The dynamic membership list can be used to identify group's access control subjects [66]. This feature could be used to automatically assign users to roles in enterprise system. LDAP returns a list of roles which there are no logical relationships among roles.

Liu et al. [68] presented a method for Attribute and Role-Based Access Control (ARBAC) for web services. This model automatically generated a set of role based on the user's attributes, created a mapping among users, permissions and roles, and defined an access control for web services and resources. Under this model user's attributes are static and users only have pre-defined attributes.

Saunders et al. [69] defined a model of Role-Based Access Control based on the access matrix. This model uses access control matrix concept which specified individual relationship between subjects wishing access to resources and system resources or objects they are given access requests. In a two-dimensional array, for each subject-object pair the allowable access appears in the related entry. In this model each object allows to store the access control matrix, so when a subject wishes to access an object, the system needs to do some search in that matrix and then derives the result. Saving access control matrix for each object might be a drawback of this model.

Cruz et al. [43] described a security framework for participant, users or applications, that dependent to Role-Based Access Control model. In this approach roles are pre-defined and

formed in a hierarchy, although users are not identified previously. Users are allowed to perform an action in terms of their attributes values, and values of the attributes belonging to the resources. Hence, the actions that users can perform are dynamically determined. These values might change over time (e.g. the user's location), hence the result of evaluation of accessing to a resource must be changed over time. When the number of users is more than the number of roles, defining an automated way to allocate roles to users is reasonable. Privileges granted to users might be differ among users. In other words, different users have different authority depending on their status. In this approach the roles assigned to each user are designed in a hierarchy where higher roles have all the privilege of lower roles.

Saffarian et al. [44] proposed a dynamic user-role assignment method for remote access control where an outsider makes inquiry to access to a resource of an organization. In remote access control system, the requestor is not registered and he is not a member of the organization. So he has not been assigned to specific role. This proposed method gives least privilege to outsider without downgrading the efficiency of the access control system. Moreover, this method uses credential and past behavior of outsider to give privilege to access a resource and outsider cannot compensate the lack of credential by having a good past behavior.

Bertino et al. [37] defined a physical extension of RBAC model called Triggered Role-Based Access Control (TRBAC). TRBABC supports regular enabling and disabling role and physical dependencies among actions activated by role triggers. Role trigger actions may be performed immediately or postponed to a particular time. Postponing action may help to solve conflicting actions.

Attribute-Based Access Control

Attribute-Based Access Control (ABAC) was defined by the National Institute of Standards and Technology (NIST) for the first time to improve data sharing where data are supervised [10]. ABAC represents a part of logical access control that consists of Access Control List, Role-Based Access Control, and Attribute-Based Access Control models. ABAC model determines access to resources based upon the evaluation of attributes. ABAC is an access control model which is based on the set of characteristics, or attributes associated with resource, requester, and the environment. Attributes store various data of entity to unify them. Generally, an access is granted or denied to a user, who requests to perform an action like read from an object or write to an object, based on the identity of that user. This identity may be defined either directly or with predefined property types such as roles which are attached to that user [10]. We should have a method to manage the access control in a dynamic way. This means that we need to define a method in order to control the access of objects by evaluating operations, environmental conditions relevant to a request, and rules stored in the property of the user. ABAC method is a solution to grant or deny user access based on attributes of the target user, target resource, and environment conditions. ABAC has been defined in different ways. A paper on web services provides a definition for ABAC like "*grants access to services based on the attributes belonging to the requestor*" [11]. A geographic information system describes ABAC as *"a method which values of attributes belonging to users determine the coalition of users with privileges"* [12]. Another paper defined ABAC as a model that is based on the attributes of subject, object, and environmental condition and supports both discretionary and mandatory access control requirements [13]. Fortunately, there is a reasonable unanimity that ABAC defines access by matching the current value of user attributes, resource attributes, and environmental conditions

with the requirements defined by access control rules. Regarding this definition, if a user requests to access to a resource, access control mechanism evaluates user attributes, object attributes, environment condition, and rules to make a decision. Finally, subject is given permission to access a resource if it is authorized. ABAC offers a subtle access control approach which accepts numerous discrete inputs for making access control decision and then provides a big set of possible combination of those inputs which are match with request. ABAC should be able to empower resource owners to determine access control policy without prior knowledge of the specific requestor. It means that as new users join the social network, rules do not need to be revised. The access control policy implemented in ABAC are confined only by the computational language and the robustness available attributes.

One restriction of ABAC method is that in large environment with large number of resources and users, there might be various attributes and access control mechanisms across the organization. It is often necessary to reconcile access control throughout the organization to meet enterprise administration requirements. To alleviate this limitation, Policy-Based Access Control facilitates organization to have more homogeneous access control scheme across the organization.

Yuan el at. [20] explained the ABAC model by concentrating on its authorization architecture and policy formulation in order to define an approach to apply ABAC model to securing web services. They assumed that ABAC model includes two parts: the policy model which defines all policy in the ABAC model and the architecture model which employs the policies to control the access to the web services. In their approach, ABAC model defines agreements based on the security characteristics which called as attributes. They defined three types of attribute such as subject attribute, object attribute, and environment attribute. As it is

seen, the attributes were attached to three different types of entity such as subject, object, and environment of the web based system. Generally, each type of attribute, provides the identity and characteristic of related entity. After defining attribute, they defined policy by using attributes. It means that policy rule decides whether or not grant access to an object based on attributes of subject, object, and environment condition. These policies are defined by the system. They show the advantages of ABAC by comparing this method to role-based method.

Hai-bo et al. [22] described administrative scalability and control granularity which those are the serious problem in web services. Web services technology enables organization to accomplish their software as service. Web services environment is a distributed system and implemented between heterogeneous systems. Hence they are different comparison of objects typically protected in conventional systems. In order to protect objects in web services system, Hai-bo et al. [22] offered ABAC method as an effective solution. ABAC method identifies which attributes should be employed in order to get access to a certain service. Again in this model attributes are belonging to the entities of the system. Besides, they offered automated trust negotiation mechanism to disclosure the sensitive attribute. The proposed approach used TrustBuilder [23] to describe attribute credentials and trust negotiation policy. In this model TrustBuilder runs on server and decides how sensitive information is disclosed to the other parties. When a user requests on the web browser, the request is sent to the web server. The web server asks ABAC module to evaluate its access control policies and decide whether this request should be accepted, denied, or reevaluated.

Enterprise ABAC concept

As mentioned before, ABAC is able to share data and information. Moreover, ABAC is able to be deployed through enterprise model. In this case the components collaborating to implement

ABAC get more complex. NIST provides a definition to explain enterprise; it is "*a collaboration or federation among entities for which information sharing is required and managed*" [10]. Regarding this definition, most enterprises have various components and form of identity and credential management to manage entities existing in the enterprise. Correspondingly, many enterprises may have some organizational policies for building rules validating subject's access to enterprise objects. NIST suggests enterprise subject attributes will be created, stored, and shared through enterprise regarding subject attribute management capability. Similarly, enterprise object attributes will be established and connect to objects regarding an object attribute management capability. Attributes must be named, assigned a set of valid values, assigned a schema, and companioned to subjects and objects. Attributes shared through enterprise should be placed, retrieved, validated, updated, and revoked [10]. The enterprise ABAC model should have numerous functions that are responsible for computing access decision by enforcing policy decision in response to a request from subject requesting an access to an object protected by own policy, retrieving attributes belonging to specific object or data required for policy evaluation, and executing the workflow that defines the order in which attributes and policies must be retrieved and accomplished.

Policy-Based Access Control

Attributes store various data of entity to unify the entity. Each attribute is a distinct field that a policy can use to determine whether or not to accept or reject an access resource. Policy-Based Access Control (PBAC) is an access control method and enables organization to governance their resources by defining policies. Organizations define policies and administration structures to ensure the successful execution of organization's mission. Furthermore, they need to generate policies that define who and under which circumstances should have access to which resources.

PBAC is assumed to be a standard version of the ABAC model at an enterprise model for supporting particular governance objectives. PBAC combines attributes associated to resource, requester, and environmental condition with information of set of conditions. These conditions define under which the access request is made and uses set of rules that specify whether or not the access is allowed. In ABAC model, the attributes needed to have access to a particular resource are designed on a local level and might be different from one organization to another organization. In the PBAC model, the organization can define one policy governing access to all resources that meet particular criteria, and this policy might be enforced for all attempts to access the resource, no matter where resources are managed by any organization. PBAC can be said that much more complicated model, and there is need a mechanism to define policy in unambiguous term. It is vital that policies must be clear and unambiguous; otherwise, there is potential for unmanaged access to a resource with which a specified policy is associated. There is also a need to ensure enterprise creates attributes by using the same way and uses the same attributes for access resources.

Relationship-Based Access Control

In social network site, users and resources are connected via different types of relationships. Cheng at al. [14] defines user-to-user relationships as a basis form for social network sites. In this model, which is based on the relationship-based access model, user-to-user relationships have an important role in indicating and implementing access control. This research developed a path checking algorithm to investigate if the suitable relationship path between users for a given access request exist.

Access control based on mutual relationships was defined by Cheng at al. [15]. They combined Attribute-Based Policies with Relationship-Based Access Control. Defining the access

requirement is based on the attributes of users, attributes of relationships, and environmental conditions. A Relationship-Based Access Control model is implemented based on three important factors: relationship type, depth, and strength. Consequently, this method uses the path-checking algorithm for finding a path or a series of relationships which satisfy the attribute-based requirement [15]. Carminati at al. [75, 76] recommend a type of Relationship-Based Access Control model where trust level, type, and depth of interpersonal relationships are defined as key decisions for authorization. Nonetheless, using relationship alone is not sufficient for controlling the access to resources. Hence, Cheng at al. [15] consolidated attribute-based policies into relationship-based access control. While these models have their own advantages, one drawback of two models is traversing the long path to accept or deny a request, which is definitely time-consuming.

Social network systems lead a pattern of control access which is distinct from traditional access control approach. Gates [38] identified the term Relationship-Based Access Control to refer to this pattern. According Gates' definition a pattern of access control needs to be developed in terms of interpersonal relationship. Relationship-Based Access Control has been defined based on tracking of interpersonal relationship between users and explanation of access control policies in terms of these relationships. Fong [39] defined the requirements for extending the applicability of Relationship-Based Access Control to application domains rather than social computing. Fong formulated a sample Relationship-Based Access Control model to capture the paradigm. That was an authorization decision which was based on the relationship between the resource owner and the accessed resource. This model defined policy language in terms of the logic model. Fong offered two features. First, the social networks used in this model are poly-relational which means this model tracks not only a relationship existing between two

individuals, but also the type of relationship. Second, the model tracks multiple access contexts. Relationships may be defined in separate contexts and the access contexts are organized into a tree-shaped hierarchy. This hierarchy represents a sharing mechanism known as relationship inheritance. When an access is requested in this hierarchy, the relationship expressed in all the ancestors' contexts are combined with the relationships in the target access context to respond the access request.

Fong et al. [40] added two features to the policy language in the previous Relationship-Based Access Policy. They described that relational policy cannot be expressed in the relationship-based access control, and the limitation is because of the lack of support for disjoint intermediaries and vertex identification. After adding two features, owner-checkable policy and a super set of relational policies, to the Relationship-Based Access Control, the new method was generated. Moreover, they described that this model is able to define every finite relational policy. Finally, they defined a policy language that is adequate for expressing useful Relationship-Based Access Control policies.

Trust Management

Trust management is an approach for authorization and access control in open, decentralized, and distributed systems [45]. In centralized system or traditional access control mechanism all users such as owners of resources and requestors are registered. They have access to the system by using their credential or identity, so authorization decision is made in terms of the identity of the requestors. On the contrary, in the decentralized systems the users may not be registered into system and the resource owner and requestor are usually unknown to each other and access control based on the identity might be ineffective. Hence, based on the trust management approach, access control decision is made in terms of policy statements defined by all parties in

the system. Some statements, which are called credential, are defined and created the time when users signed up to the system. Credential is used for user authentication in future. Some statements which are called access rules, might be defined and stored in trusted storage. They govern access a resource [45]. In trust management mechanism, a requestor generates a request. Authorizer, who determines access control rules regulating access to the requested resource, receives the request and check the requestor's permission.

Ninghui et al. [46] introduced a framework for role-based trust management languages for describing credentials and policies in decentralized authorization. It is a part of addressing the security problems which appears when independent organizations enter into combinations whose all participants change quickly. In the combination of several organizations, some of organizations might be autonomous organizations planning to share resources. In this case every organization has authority over the resources to control the access to those. This system will be called decentralized collaborative systems which does not longer have single central authority.

Golbeck et al. research [16] payed attention to trust in social network and determine how trust information should be mentioned and integrated into applications. They focus on how much we trust to others to give them access to our resources. Their method assigns a value as trust rating exist between two users who are indirectly connected. Marsh [17] formalized trust as a computational concept. Because his model was based on social and psychological factors and highly theoretical, this was complex and difficult to implement. His focus was about interacting agent that could maintain information about history and observed behaviors, so it was improper for use in social networks. For multi-agent systems, Castefranchi and Falcone [18, 19] denoted an analysis of trust. Their work was based on the psychological literature and includes many psychological factors.

Zuo et al. [21] developed a method to make a decision based on object trust management. This decision will be yield in terms of various defined policies and assists users in selecting safe and secure information in an open system. In this method, information integrity will be evaluated based on the quality and security features of a given piece of information. This method accepts external information with the required level of quality and security in an open environment.

Policy Analysis

Kolovski [24] compared all existing approaches for access control policy analysis in web services. This means that all approaches defined methods to protect the sensitive information in web. Hence there is a need for adequate security and privacy support. Finally, all approaches required to define appropriate access control policy languages for environments which are large, open, distributes, and heterogeneous. These languages have objectives to be expandable and flexible. They might have adequate features to be expressive and distributing access control policy. Although some approaches used XACML or WS-Policy languages, these are unclear in terms of impacts and consequences of their access control policy.

Kolovski's research was a survey including policy languages that have formal semantics and provided algorithms for access control policy analysis. Moreover, Kolovski defined the existing policy languages (XACML, WS-Policy, ODRL, XrML) which provide a formal semantics and policy analysis service which previously was inconvenient for the specific language. Based on Kolovski's research policy languages might be categorized by two groups. The first group or logic-based access control policy language are based on Datalog mostly. Hence, if an access request satisfies the access control policy, it will be completed in PTME. Woo et al. [25] is one of the earliest definition of logic-based framework for denoting authorization. They expressed using a default logic to model control rules and authorization. They fragmented the logic to

extended logic program which ran in polynomial time. They also formalized the Bell-La Padula security model [26]. Another approach was Delegation Logic [27] which was based on the logic programs and identified delegation depth and supported a huge spectrum of complex principles. This approach didn't comment policies analysis and verification. Another framework for reasoning in open distributed systems was PeerAccess [28] that provided a declarative description of the behavior of peers that information comes from other specific peers selectively. PeerAccess is a local knowledge base and encodes the basic knowledge of each peer. Using defined policy, PeerAccess made decision to release information to other peers. This model is similar to PeerTrust [29]. Other policy languages supporting delegation are: Abadi et al. [32], Becker [49], Binder [93], and Cassandra [33, 84]. All but Abadi et al use Datalog as fundamental for syntax and semantics. Proof-carrying authorization [34] which is an authorization framework, is based on a higher order logic which used to control the proofs is undecidable, and this problem does not force users to generate proofs by their own. Jajodia [35] proposed a logical policy language for authorization. This policy language allows users to specify policy language in terms of which accessing control decision are to be made. One of the earliest approach is Ulman et al. [36] which allows having various numbers of subjects, roles, resources, and authorizations. Although this method is very expressive and it could model almost production system, there is no practical solution if a given subject can gain an access to a given object.

Multilevel Security

People and information are classified into different levels of trust. These levels represent the well-known security classification such as unclassified, confidential, secrete, and top secret. Bell-La Padula defines a model based on the multilevel security model [26].

Team-Based Access Control

Alotaiby et al. [41] defined a model which is built on the role-based access control. This model allowed certain users to join a team in terms of their existing roles in an organization and team specification. These users are allowed to request to get permission to have access the resources existing in this organization. This model was based on the Role-Based Access Control, so access control management is accomplished by associating roles to users. A team is a collection of users who have different responsibilities in that team. Hence, users having various roles are allowed to join the team and in terms of their role they have permission to access protected object existing in the organization. Sometimes users require to have various roles in team. Therefore, this model defined a session which maps one user to one or more roles. Session might confine the users to accept a certain role which are needed to join the team. A user might have active multiple sessions at the same time, but in different teams.

Thomas [42] introduced Team-Based Access Control model for applying Role-Based Access Control in cooperating environments. The main concept in team-based access control model is a team which is a collection of users who are assigned with specific role in order to accomplish a specific task or goal. Users with various set of qualifications and responsibilities are mapped to various roles. The first consideration was having a hybrid access control model bringing advantages of having Role-Based Access Time. The second consideration was recognizing the context association with collaborative tasks and ability to apply this context to decision in terms of permission activation.

Access Control in Online Social Network

Park et al. [48] developed an access control framework for online social network by grasping the concept of user activity. This framework encompassed individual privacy for user activities

and resources by separating individualized user and resource activities. Under this model three main components have been defined such as user, sessions, and activities. Each activity comprises of an action, zero or more target resources, and zero or more target users. In this framework, users are associated with user attributes and policies. User attributes provides properties and information about the user. User policies provides rules determining confinement and preferences. A session represents active users who logged into the online social network. In this framework ability to distinguish active user those who are online and non-active user those who are not online is important. A session could have additional attributes and policies, and a user might have multiple concurrent session whereas a session associated to exactly one user. Activities consists both general usage activities and users' control activities. A session initiates each activity on behalf of the user. The online social network decides whether or not the activity has permission. A session might have several activities, whereas each activity is initiated by only a single session. Each activity might include an action, target resources, and target users. Each action is a function accessible to users via a session. Target resources include users' shared asset. Target resources are involved in an action. Target users are the recipient of an action.

Ontology

Masoumzadeh et al. [50] proposed an ontology-based access control model for social networks. This model uses semantic web-based technology for describing complex interpersonal relationships. Furthermore, this model considers the individual user's right to have a flexible control over the defined access control related to them.

Fong et al. [77] represented a formal algebraic model for access control in social network site like Facebook. They define two stages for Facebook-style access control: reaching the search listing of the resource owner and accessing the resource. This model recognizes two types of

policies: accessing resources and searching through resources policy. Furthermore, the authors propose a model for social computing application in which user authorization is performed based on user-to-user relationships. In order to specify the policy, this model uses a model logic language [78]. Finally, they improved the model logic language to allow multiple relationship types and directional relationships [79]. Fan et al. [80] defined an attribute-based access control model for web services; web services grant access to a user based on attributes of the related entities. This model uses an automated trust negotiation mechanism to explore some issues regarding sensitive attributes.

3. Policy-Based Attribute Access Control Method

3.1 Introduction

Over the past several decades, online communication as a mean to establish public relations or information and resource sharing has gained a lot of popularity. To ease of information sharing or access to resources, protecting information and resources becomes an unavoidable part of online communication. In the last few years, numerous access control methods have been proposed with a view to maximize protection of resources and information.

Traditionally, access controls are mainly based on identities or role memberships. Therefore, most of the so-called traditional access controls fail to have a proper solution with the scalability and dynamicity of social networks. As a result, it is not efficient to specify users with access to information and resources in a traditional method. Alternatively, Attribute-Based Access Control (ABAC) have been developed and are currently the most extensively-used access control mechanisms for social networks. Under this mechanism, access controls are determined in terms of the properties of users. Users are granted permission to access others' information or resources based on their attributes and properties. Hence, there is no requirement in defining access levels when users are added to social network. Moreover, Policy-Based Access Control (PBAC) is one of the most considerable access control mechanisms for social networks. In PBAC, resource or information owners can determine mechanisms to control received access requests based on the policies, which have been defined and stored in the social network.

In spite of its popularity in both theory and practice, the current PBAC model is still far from the ideal access control mechanism. Users cannot specify policies in order to protect their resources in a simple way. Moreover, the ABAC model is not a very robust access control mechanism. The ABAC can use only the attributes, which are defined as preliminary in the social networks.

To address the above-mentioned shortcomings, the work presented in this dissertation proposes a change in policy-based access control models with the goal of empowering users to be able to specify their own policies for protecting their resources. A new model incorporates attribute-based access control into altered policy-based access control and helps providing a balance between ease-of-use and required level of security will be introduced and discussed. A format will be formalized to present rules and queries that are converted from policies and inquiries, respectively. This format will also address the access requirements in terms of the access to objects by evaluating requestor's attributes, object's attributes, action or operation taken by requestor (such as read, write, or alter), environment conditions, and policies created by ordinary users or super users of social networks or organizations to protect the associated resources. This approach, called Policy-Based Attribute Access Control (PBAAC), is an efficient method to enforce access control in a dynamic way. In order to evaluate the request, the PBAAC uses Attribute-Based Access Control and Policy-Based Access Control models. The proposed method also explores the access control requirements for an object in cases where the object is composed of other objects. This model offers an appropriate solution in cases where several policies conflict with other policies.

Since social networks' data are stored on the cloud server, many security requirements such as user authentication and establishment of secure communication among servers will become issues of concerns. These aspects of social network security are outside the scope of this dissertation. Specifically, the focus of this dissertation is mainly on the access control requirements for social networks.

This chapter presents and discusses the basic concepts, terminologies and entities existing in the proposed method. This chapter further describes the PBAAC model, basic notations used for

describing this model, formulations to define policy language, converting policy and inquiry to a simple format, and evaluating simple forms of policies and inquiries. On the assumption of facing conflicts among basic formed of policies, the rest of the chapter describes how these conflicts will be removed.

3.2 Preliminary

This section describes terminology, basic concepts, and entities of the proposed model.

Terminology

The following terminologies are adopted in this dissertation in order to define concepts existing in the proposed model [84].

- Subject typically represents a class of entities requesting to perform an operation on an object or taking ownership of an object (i.e. resource). Subjects might be categorized into active and passive subjects.

  Active subject or a person can have three roles such as a user, a supervisor, and an owner of an object. A user is able to initiate inquiry in the model. Subject may represent an entity who is the owner of assets and manages all associated information and resources as well. Moreover, subject can be a supervisor who oversees the social network and controls all events happening on the social network.

  Passive subject or a non-person entity, such as anonymous service or application, may generate a request on behalf of a person to access an object or to perform an operation on an object. This process may be triggered as a result of some events in the model. Providing detailed information about a passive subject and defining a method for adding this entity to the proposed model will not be considered in this document. This addition is left for future development.

- Object generally represents a class of entities receiving an access request. Object could be a target resource or a target user. Target resource is a class of entities required to be protected from unauthorized use. Target resources such as information and documents are assets of the model. Target resources, therefore, are belonged to a subject on the social network.

  In order to secure resources against unauthorized access, resources must be managed by a subject that may be the owner of the object or supervisor of the social network. In other words, subjects are responsible for determining policies to protect their resources. As will be mentioned later, a user who protects associated resources will be referred to as controlling user. A controlling user defines policies for every associated resources and stores the policies into the corresponding resource's repository. Once a resource receives an inquiry, that resource is responsible to respond to the inquiry.

  An object sometimes has a role like a subject. In these cases, the object is referred to as target user, who sets the requirements for protecting associated information from unauthorized access.

  In general, there are two types of resources, simple and compound resource. A simple resource cannot be split into other simple resources, whilst a compound resource is derived from other simple resources. The rest of this chapter provides subtle definitions for the two types of resources.

- Relationships represent communication among entities. Relationship is a class, which defines connections among subjects, objects, and subjects to objects under different relationship types. Relationships can also be assumed among entities (i.e. connection between accessing users and target users) in the social network. In addition, relationships in the virtual world may represent relationships in the real world. For instance, in a social network site, users

might connect to each other under various relationship types such as “friend”, “parent”, “colleague”, or “manager”; relations exist among people in the real world as well.

- Environment condition is a class of dynamic factors that may consist of time and location. Environment condition is independent of subject and object and may be used as a parameter for evaluating an inquiry and making a decision to accept or reject that inquiry on the social network.
- Action is a class of operations or accesses. Generally, action is initiated by a user to access an object, or to perform an operation against an object.
- Attribute is a characteristic of a subject, an object, a relationship, an action, or an environment condition. Attributes are used to make each entity unique. Moreover, attributes utilize social networks to be able to group subjects and objects. There are various sets of attributes, naming subject attributes, object attributes, relationship attributes, action attributes, and environment attributes. More descriptions for defining attributes will be provided later in this chapter.
- Policy is written from the viewpoint of an object or a subject that requires protection from unauthorized use. Policy is a regulation and defines who, when, where and under what constraints an inquiry is allowed to perform an action against an object.
- Rule is a converted form of a policy to control all accesses to a particular object. On social network, rules might be supposed as an asset of a subject or object. These are saved on a repository belonging to someone, like the supervisor who generates rules hence these rules must be secured against unmanaged access.
- Inquiry is an expression asking to perform an operation on an object in a social network. Generally, inquiries are generated by a subject or an accessing user, specifically.

- Request is a transformed format of inquiry.
- Metadata provides more information about elements existing in our model, such as relationship metadata, which offers information about the relationship types. We also store information of adjacent and trusted users. More explanation of adjacent and trusted users will be provided in the rest of this chapter.

Later, this document will provide detailed description of each element on the social network and information stored for identifying the element on the social network as well.

In order to present the PBAAC model, the model's entities, the relationships among entities, and the attributes and the policies with simple notation attached to each entity, some basic notations need to be defined at first. These notations are presented in the following.

Quantifier Notation

Table 1 represents all quantifiers and symbols used in the proposed model.

Symbol	Description
∀	Represents all entities.
∃	Represents that there exists an entity.
∄	Represents that there exists no entity.
∋	Represents a membership of a set.
ϵ	Represents an element of a set.
∉	Represents not an element of a set.
€	Represents a resource belongs to a user.
⇒	Represents a connection between two entities.
--$\overset{r_i}{}$-->	This symbol ---> represents a relation and r_i identifies the type of the relation.
*	Represents the multiply of weights assigned to each concatenation in a path. (Asterisk)
+	Represents concatenation of connections in a path. (Plus)
⋁	Represents the union of multiple statements. (Disjunctive connective)
⋀	Represents the intersection of multiple statements. (Conjunctive connective)
¬	Represents reverse value of a statement. (Negation)
:	Separates the type and the value of the entity. (Colon)
⊕	Represents an action or operation, which might be run against an object.
⊗	Represents an action mentioned in a rule or a request.
⊂	Represents a resource including other simple resources.
⋂	Represents intersection of statements.
⋃	Represents union of statements.

Table 1 – Quantifiers and Symbols

Adjacent user

An adjacent user is one who is in a particular distance from another user. This concept is defined in terms of the particular distance or particular number of connections between adjacent

user and another user. Based on the adjacent user definition, a path which connects a user or source user to its adjacent user or destination user might include several connections, where all connections must have the same relationship type. It is obvious that there might be various paths from a source user to a destination user. This definition of who is an adjacent user of another user (source user) might be provided by the user (source user) or the supervisor in the social network. More explanation is provided in section 3.5.

Trusted user

A trusted user of a user is defined by the user or the supervisor in the social network. To define who the trusted user is, social network multiplies the values, each of which is attached to every connection in a path and defines the degree of loyalty, and then generates a new value as a result. This result identifies how much, in percentage, a user or source user trusts another user or destination user. This path connects source user to destination user. If the final value or derived result meets the definition of trusted user, then the destination user can be defined as trusted user. Again, like the adjacent user, in one path, which is started from a source user and ended by a destination user, all connections have to have the same relationship type. It is obvious that there may be various paths from a source user to a destination user. More explanation is provided in section 3.5.

3.3 Secure Access Control Model

In this document, a model is defined to support access control in social networks in a highly dynamic way. In the preliminary section, various elements used in this model were defined. In order to distinguish an element from other elements, several properties are attached to each element. Properties add unique identification to the element. These elements are called entities and the attached properties are called attributes. Attributes are part of the proposed access control

model, and are specifically used to enforce access control to resources. A model using attributes to accomplish access to resources is Attribute-Based Access Control model (ABAC).

ABAC represents a part of logical access control that consists of access control lists, role-based access control, and attribute-based access control method [10]. In brief, ABAC method determines access to the resources based upon the evaluation of the attributes of the entities existing in the model.

On the other hand, every entity should be able to protect associated information and resources from unauthorized access by defining one or more policies. Policy-based access control (PBAC) is an access control method, which enables social networks to govern their resources by defining policies. Policies determine who is allowed to have access to which resources and under what circumstances. The PBAC model may be a complicated model; there is need a mechanism to define policy in unambiguous terms. It is essential that policies must be clear and unambiguous; otherwise, there is potential for an unauthorized user to access a resource with which the ambiguous policy is associated.

The PBAC concept is used in this dissertation to develop the proposed model because the PBAC model combines attributes with sets of rules. Attributes are associated with resource, requester, and environment. Rules define access control constraints and determine under which conditions access to a resource will be granted. Unlike the PBAC model, the proposed model enables entities to define one or more policies governing access to their own resources and even some entities are able to define policies on behalf of other entities to supervise all attempts to access their resources. In this model, there is a unique entity that can regulate other entities and can dominate other entities. In other words, the policies defined by this unique entity can

override other policies defined by other entities. This new model, called Policy-Based Attribute Access Control, is explained in more details in the following section.

3.4 Policy-Based Attribute Access Control Model

Figure 2 shows the basic structure of the Policy-Based Attribute Access Control (PBAAC) model, which simply represents the elements and the connection among elements. Generally, five elements, referred to as major elements, are highlighted in this model. These major elements are subject, object, relationship, action, and decision engine.

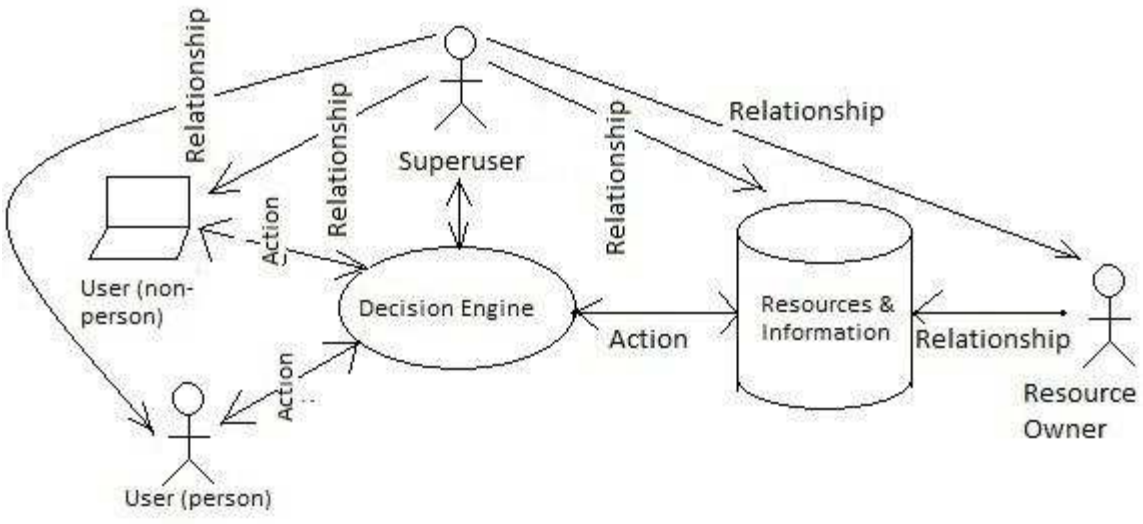

Figure 2- Basic PBAAC model

The following gives a brief description of each element:

- The first element is the subject, which accepts different roles; namely user, resource owner, and supervisor.

  A user, referred to as an accessing user, is the only entity capable of initiating a request to access an object, or performing an operation against an object. An owner of an object, referred to as a controlling user, is responsible to protect associated assets and individual information. A super user, called a supervisor, administers the system and oversees all events

happening in the system. Supervisor defines various policies to manage all operations on the social network. More information about the supervisor's responsibilities in the proposed model will be explained later in this document. It should be pointed out that, in this dissertation, user and accessing user are used interchangeably and controlling user is often used to refer to resource owner.

- The second element is the object, which accepts two roles: accessed user and accessed resource. Accessed users are supposed to receive an access request, such as poke or tagged in a photo, and resources like software resource (e.g. document or photos) and hardware resource (e.g. printer or scanner) are supposed to receive an operation such as update or delete. The object, which will be given a request, is assumed to provide an appropriate response to the received request. In this document, target user is an exemplification of an accessed user and target resource is an exemplification of a resource. We use target resource and resource interchangeably.
- The third element is the relationship, which denotes connection among elements. Relationship may be identified by various types. Later in this chapter, more explanation about relationship types and corresponding metadata will be provided.
- The fourth element is the action, which is requested by a subject to access to an object or to perform an operation against an object; this subject may be supervisor or accessing user. Note that, in a social network, there are various actions that will be addressed later in this chapter.
- The last element is the decision engine. In brief, the decision engine receives inquiries and designates whether or not the subject (or accessing user) is allowed to access the object or perform an operation on the object. In order to approve or deny the given inquiry, the

decision engine evaluates the inquiry. This means that the decision engine collects all necessary information and policies from the model and then reaches a result. This result may be either accepting or rejecting the given inquiry. The information comes from attributes associated with various entities such as subject, object, and relationship. These entities are related to the inquiry. This means that the inquiry determines which entities must be reached and which attributes must be read. Beyond the attributes, appropriate policies, which defined by subject or object, and related to the inquiry, must be reached.

Regarding the main elements of the proposed model, the following entities existing in the model will be recognized:

- Accessing user, controlling user, and supervisor are all instances of the subject class.
- Target user and target resource are both instances of the object class.
- Several relationships are instances of the relationship type class.
- Environment condition is an instance of the environment condition class.
- Several actions are instances of the action class.

The remainder of this chapter provides a comprehensive explanation for each item existing in the proposed model.

3.5 Policy-Based Attribute Access Control Model Components

Figure 3 shows the PBAAC comprehensive model.

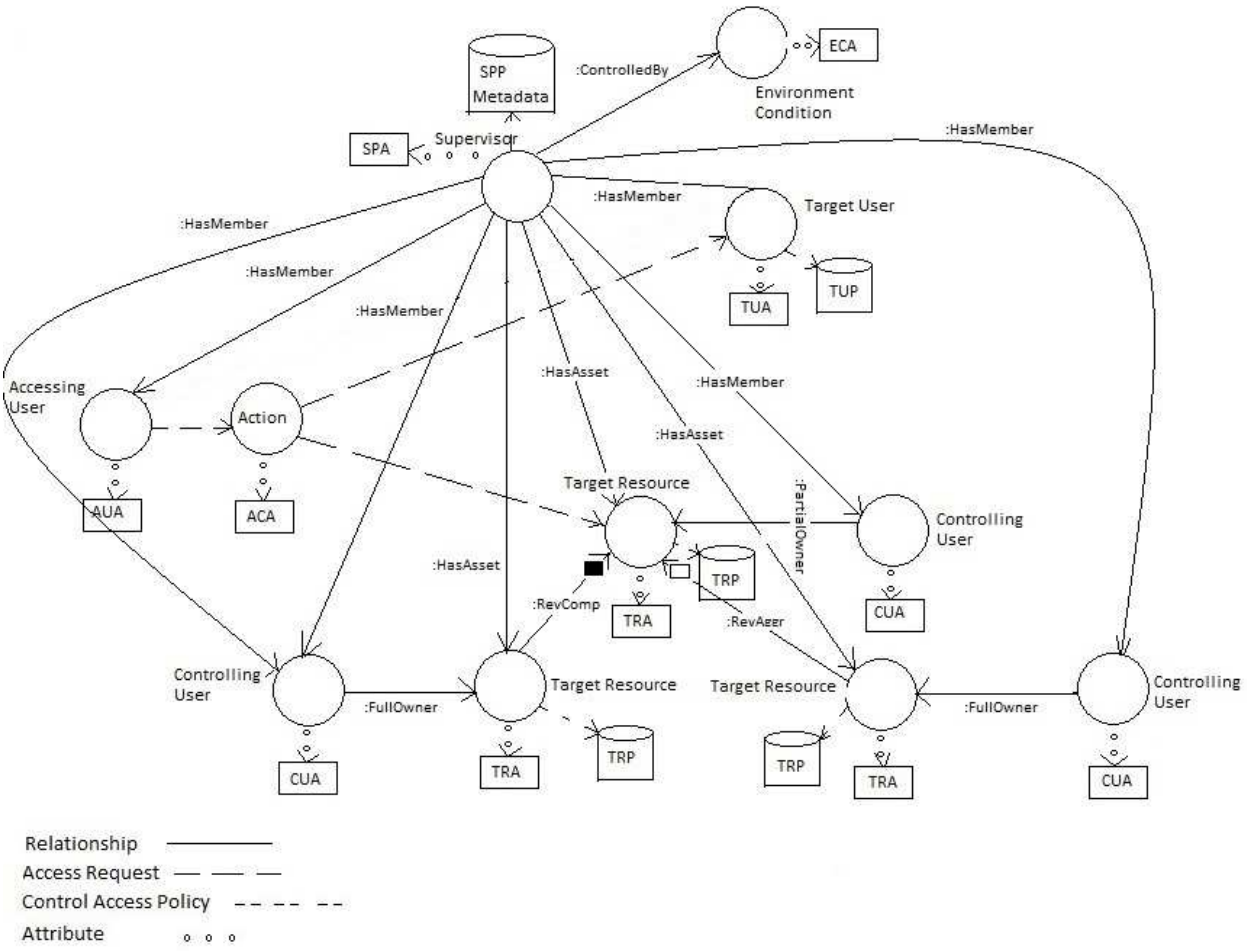

Figure 3- PBAAC model Components

The following listed items explain components of the PBAAC model.

- A set of relationships includes all types of relationships that may exist among entities. As the matter of fact, relationship type defines the characteristics of connections between subjects, objects, and subjects to objects.

  A set R= {r_1, r_2,……., r_n, r^{-1}_1, r^{-1}_2,……., r^{-1}_n,r^{-2}_1,r^{-2}_2,……,r^{-2}_n} represents the various relationship types existing in the proposed model such as "*friend*" or "*parent*". Relationship type may have different characteristics such as:

- Symmetric: A relationship r (e.g. $r_i \in R$) is *symmetric*, if both parties are related under the same relationship. In other words, if x relates to y under a relationship r, and r is a symmetric relationship, then y relates to x under the same relationship. So, for two entities, like x and y, if r is symmetric relation and x relates to y under r, y will relate to x under r.

  Regarding the general definition for symmetric relation, it is rewritten in the proposed model as follows:

  r is a symmetric relation if $\exists$ x, y and x-----$\overset{:r}{}$-----> y then y------$\overset{:r}{}$------> x.

  For representing a symmetric relation r in the proposed model, r^{-1} is added to the set of relationship types. Hence, r, $r^{-1} \in R$ is presented to show that r is a symmetric relationship. For instance, "*friend*" relationship type is a symmetric relationship. If Alice is Bob's friend, so Bob is Alice's friend. Hence if r represents "*friend*", and $r \in R \wedge r^{-1} \in R$, the model indicates that this relationship type (i.e. r) is a symmetric relationship.

- Transitive: A relationship r (e.g. $r_i \in R$) is *transitive*, if x relates to y under the relationship r, and y relates to z under the relationship r, so x relates to z under the relationship r. This relation can be shown as follows: r is a relation between x, y, and z. If r is transitive relation and x relates to y under r and y relates to z under r, x will relate to z under r. Based on the general definition for transitive relation, it is rewritten in the proposed model as follows:

  r is a transitive relation if $\exists$ x, y, z and as x----$\overset{:r}{}$----> y $\wedge$ y----$\overset{:r}{}$----> z then x---$\overset{:r}{}$----> z.

  For representing transitive relation r in the proposed model, r^{-2} is added to the set of relationship types. Hence, r, $r^{-2} \in R$ is presented to show that r is a transitive relationship. For instance, relationship "*manager*" is a transitive relationship, if r identifies "*manager*"

relationship, and r^{-2} is added to the set of relationship type. (e.g. $r^{-2} \epsilon R$). For example, if Jill is Joe's manager and Joe is Bob's manager, Jill is Bob's manager as well. Hence if $r \epsilon R \wedge r^{-2} \epsilon R$, it shows that the relationship type (i.e. r) is a transitive relationship type.

It is obvious that some relationship types are neither symmetric nor transitive, like "*teacher*" relationship type. Hence, if r identifies "*teacher*" and $r \epsilon R$, $r^{-1} \notin R \wedge r^{-2} \notin R$.

In the work presented in this dissertation, it is required to keep information for every relationship type. This information, called relationship metadata, stores several properties related to relationship type. Table 2 represents the relationship types metadata.

Relationship Type	Symbol	Symmetric Symbol	Transitive Symbol
friend	frnd	frnd $^{-1}$	Not Exist
manager	mngr	Not Exist	mngr $^{-2}$
teacher	tchr	Not Exist	Not Exist

Table 2- Relationship Type Metadata

Because the entire social network uses this information, this information is restored into repository of the supervisor. Consequently, the supervisor is the owner of the relationship metadata.

Providing a compound object in the social network requires determining relationships between compound objects (i.e. container objects) and simple objects (i.e. contained objects). This is true since sometimes deleting a part of the object may affect the existence of the whole object. Unified Modeling Language (UML) defines two relationship types to present relationships between container and contained object; those are aggregation and composition relationship. Under the aggregation or composition concept, deleting the container may or

may not affect the existence of contained objects. Unlike the general definition for aggregation and composition relationship, it is required in the proposed model to show that deleting the contained object may or may not affect the existence of the container. Hence, two relationship types are defined to work in reverse direction compared to aggregation and composition relationship. The two relationship types will be called reverse aggregation and reverse composition. The following describes a brief definition for aggregation, reverse aggregation, composition, and reverse composition. Furthermore, it will be explained how erasing of container impacts on the contained objects and vice versa:

- Aggregation relation denotes that if the compound resource is deleted, the simple resource will stay alive. This relation represents a part-whole or part-of relationship. Aggregation may happen when a class is a collection or a container of other classes, and the contained classes do not have a strong dependency on the container. The contents of the contained will not be automatically destroyed when the container vanishes [88]. The following example shows two classes: printer as container and papers as contained class. Deleting the printer does not cause a deletion of the papers. Figure 4 shows the UML notation for aggregation.

Figure 4 – Aggregation Relationship

- Reverse aggregation relation denotes that if the simple resource is deleted, the compounded resource will stay alive. For instance, if papers are deleted, printer will still exist. Therefore, a new notation needs to be defined for reverse aggregation relation. Figure 5 shows this new notation.

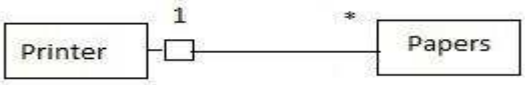

Figure 5 – Reverse Aggregation Relationship

- Composition relation, unlike the aggregation, denotes that if the compound resource is removed, the simple resource(s) will not exist anymore. Composition usually has a strong dependency between instances of the container class and instances of the contained classes, i.e. if the container is destroyed, normally every instance of the contained will be destroyed as well [88]. For instance, let us consider Folder (container) and Files (contained). Files are included in Folder, so Files do not exist separately from a Folder. If the folder is removed, the files will be removed also. Figure 6 shows the UML notation for composition.

Figure 6 – Composition Relationship

- Reverse composition relation denotes that if the simple resource is removed, the compound resource will not exist anymore. For instance, if a book includes one chapter, deleting that chapter causes deletion of the whole book. A new notation needs to be defined for reverse composition relation. Figure 7 shows this notation.

Figure 7 – Reverse Composition Relationship

Moreover, the relationship between a user and their belonging resources may be defined in terms of the ownership status that will be described in the next item.

- Ownership defines a relationship between owner and associated assets. In this model, three types of ownership will be determined, namely FullOwner, PartiallOwner, and

CollectedBy. As the names imply, if a user is the owner of an entire object, the user is the only owner of that object and the user is the only one able to govern the object. If a user is the owner of part of an object, the user is able to define policy only to control that part of the object. In other words, a rule which defines who is eligible for doing an operation on an object is accomplished by evaluating other rules associated with other part(s) of objects owned by other user(s). If a user is not the owner of the object fully or partially, that user is a collector of that object and may not be able to govern the object.

- A supervisor, who is the administrator of the social network, may have one or more attributes called supervisor attribute (SPA). Each supervisor attribute stores a part of the supervisor's information, like name or unique ID. Table 3 represents how supervisor's attributes are stored in the supervisor's repository.

Attribute Name	Attribute Value
Name	Major_Supervisor
Unique ID	SpUser1

Table 3 - Supervisor's Attribute

The supervisor defines general policies to manage all actions arisen on the entire social network and then stores them into its repository. Repository saves rules, which are the converted formats of policies. Rules may be simple or complex. Rest of this chapter provides more explanation for simple and complex rules. It should be pointed out that every rule has four parts, such as subject part, action part, object part, and condition part where subject or object part can be nested. The following structure is suggested for saving both types of rules (i.e. simple or complex), like:

- Rule#: this field stores a unique Id attached to the rule.
- Rule Type: this field stores the type of the rule, “S” for simple, or “C” for complex rule.
- Subject Part: this field stores subject part of the rule. If the rule is complex and the subject is nested, the nested part of the subject is saved in the next several rows.
- Action Part: this field saves the action of the rule.
- Object Part-Target Resource/ Target User: this field stores the target resource or target user of object part of the rule. If this rule is complex and the target resource/user part is nested, the nested part of the target resource/user is saved in the next one or more rows.
- Object Part-Controlling User: this field stores the resource owner of the rule. If this rule is complex and the controlling user part is nested, the nested part of the controlling user is saved in the next several row(s).
- Condition Part-Time: this field saves the time of the environment condition.
- Condition Part-Location: this field saves the location of the environment condition.
- Date & Time: this field saves the date and time when the rule was fed into social network.
- Order: this field saves level of the nested part, in case there exists multi nested part.
- Related To: if the original rule is complex, this field determines the current row is the nested part of either the subject or the object identified in the row before.

For instance, the supervisor stores a rule to protect all repositories associated with users, which this means each user is allowed to access to his repositories and no other users’ repositories. Hence, the supervisor defines “only one can access to its repositories”. The rule will be generated like “$< (\forall u_a), Ac(action: access), (\forall t_r (name: repository)\ \forall u_c (name: \_self)) >$” . This rule is saved in the supervisor’s repository. Comprehensive explanation of

how rule is generated will be provided in the NLP phase section. Table 4 shows this as follows:

Column	Value
Rule#	Rule1
Rule Type	"S" = Simple
Subject Part	$\forall u_a$
Action Part	Ac (name:access)
Object Part- Target Resource/ Target User	$\forall t_r$ (name: repository)
Object Part- Controlling User	$\forall u_c$ (name: _self)
Condition Part-Time	
Condition Part-Location	
Date & Time	5/12/2016-9:00
Order	0
Related To	Not Defined

Table 4 – Supervisor's Repository

Under the target resource definition, another example shows how a complex rule is saved into this structure.

Every permission for gaining access to a resource is done under the supervisor's surveillance, meanwhile supervisor and resource owner are able to enforce control access to that resource. In general, rules that protect a particular resource shall not conflict with other rules that protect the same resource. If there exists some conflict among rules, the proposed method must resolve this issue. The solution for recognizing and removing this conflict is part of the

decision engine which will be explained in detail in the request engine section of this chapter. This solution may be developed based on three criteria as follows:

- Timestamp: this item represents that conflict elimination is performed based on the date and time when rules are fed into the social network. It defines which rule must be considered. Choosing rule either recently or formerly is defined on the social network.
- Rule domination: this item represents which rule overrides other rule(s).
- Dominant: this item represents who dominates other users.

Because of keeping these criteria on the supervisor's repository, choosing one or more criteria for removing conflict is noticeably dynamic. Table 5 keeps these three criteria as follows. As Table 5 shows, only timestamp is used for solving the conflict issue for all examples mentioned in this chapter.

Criteria	Enabled/disabled
Timestamp	Recently added
Rule domination	Disabled
Dominant	Disabled

Table 5 - Removing Confrontation Criteria

The supervisor is responsible to define which rule overrides other rules, and who dominates who as well. Tables 6 and 7 show this definition.

Dominant Rule	Submissive Rules
Rule #	List of rules which are dominated by

Table 6 – Rule Domination

Dominant User	Submissive Users
User ID	List of users which are dominated by

Table 7 – User Domination

The supervisor is responsible for maintaining a large amount of data such as environment condition and metadata. Later, more explanation for environment condition will be provided. Metadata may include information about the relationship among elements as described thoroughly in the relationship section. Moreover, different types of ownership were described in this section. The supervisor stores relationship and ownership type to utilize social network for adding more relationship types, in case of users need to define new relationship or ownership types. For instance, if there are three relationship types in the social network such as "*teacher*", "*manager*", and "*colleague*", the supervisor keeps them as shown in Table 8.

Relationship Type	Mnemonic	Symmetric Mnemonic	Transitive Mnemonic
Colleague	colg	colg $^{-1}$	Not Defined
Manager	mngr	Not Defined	mngr $^{-2}$
Teacher	tchr	Not Defined	Not Defined

Table 8 - Supervisor Repository for Relationship

Moreover, supervisor stores definition for symmetric and transitive relationship categories as shown in Table 9. Using this table, it is possible to add more relationship type categories in future.

Relationship Property	Definition
Symmetric	r is a symmetric relation if $\exists$ x , y and x $\Rightarrow$ y then y $\Rightarrow$ x.
Transitive	r is a transitive relation if $\exists$ x , y, z and x $\Rightarrow$ y $\wedge$ y $\Rightarrow$ z then x $\Rightarrow$ z.

Table 9 - Supervisor Repository for Relationship Property

Table 10 shows the stored data for representing and defining ownership. Whenever users create a new ownership type, they must determine a definition for it. Afterwards, the decision

engine is able to process and understand this relationship. Decision engine can use this relationship during processing a received request.

Ownership Type	Mnemonic	Definition
FullOwner	Fown	Determines that resource owner is able to control the entire resource
PartiallOwner	Pown	Determines that resource owner is able to control the specific part of the resource
CollectedBy	Clct	Determines that resource owner is not able to control the entire resource

Table 10 – Supervisor Repository for Ownership

The supervisor is also responsible to store definitions for adjacent and trusted users that are used in evaluating requests. For instance, the supervisor defines that adjacent users must have at least one direct connection with the source users, and trusted users must get a trustee score more than 80%. Table 11 represents how data are stored for this example. More description is provided in the rest of the document.

Type	Definition
Adjacent user	$\forall$ SourceUsr, DestinationUsr ϵ SN, $\exists$ r ϵ R: SourceUsr------$\overset{:r+1}{}$------> DestinationUsr
Trusted user	$\forall$ SourceUsr, DestinationUsr ϵ SN, $\exists$ r ϵ R: SourceUsr ------$\overset{:r+*}{}$------> DestinationUsr, Min ($\prod \omega_{ri}$) >=80 , W = {ω_i | is a weight attached to each connection in a path}

Table 11 – Adjacent and Trusted User Definition

Furthermore, the supervisor stores two tables, which are used to determine the parts of speech of all words included in a policy or an inquiry statement or words that may be used in

the future. One of them is keyword table which includes words and related parts of speech for every possible word. For instance, the following table shows a part of the keyword table. Each row saves a word which may exist in a sentence.

Row#	Word	Parts of Speech (Major)	Parts of Speech (Minor)
1	Alice	Noun	Subject
2	Poke	Verb	Main
3	Someone	Pronoun-indefinitive	Subject
4	Whom	Pronoun-relative	Subject
5	Only	Determiner	
6	Me	Pronoun	Object
7	To	Preposition	
8	Documents	Noun-concrete	Object

Table 12 – Keyword Table

The word-entity table maps words in a policy/inquiry into social network entities. Table 13 presents this mapping table. Tables 12 and 13 will be used by decision engine, specifically in the NLP phase. More information is provided in the decision engine section.

Row#	Ref from Keyword Table	Entity Type	Entity
1	1	Subject	Accessing user
2	2	Action	Action
3	6	Object	Controlling user
4	8	Object	Target Resource

Table 13 – Word-entity Table

The supervisor could generate an inquiry for performing an action against an object. This feature will be part of future work and therefore will not be discussed in this dissertation.

Moreover, a supervisor is an entity to which the other entities are connected. As seen in the Figure 3, these relationship types may be "*HasMember*", "*HasAsset*", "*ControlledBy*", and "*ControlData*".

- An accessing user who has a role as a subject is a member of the social network and is able to generate an inquiry for accessing an object (e.g. target user or target resource). Almost all access requests are initiated by the accessing users. Accessing users have one or more attributes, which are called Accessing User Attribute (AUA); each attribute stores a part of accessing users' information such as first name, last name, birthdate, and so on in order to provide unique identification information. The structure of the table used for storing accessing user attribute is the same as the structure defined for storing supervisor's attributes. Later in this chapter, the attribute will be described. It will also be explained why attributes are defined under specific circumstances and how attributes will be used.
- A controlling user who has a role as a subject, controls all accesses to related resource(s) by defining policies. These policies will be transformed to rules and stored in a repository of the controlling user's resource. Structure of the table used for storing controlling user policies is the same as the structure of the table defined for storing supervisor's rules. As can be implied, regulations for accessing to target resources could be configured by the controlling user policies and attribute(s). Hence, controlling user carries one or more attributes to store associated individual information. These attributes are called Controlling User Attribute (CUA). The structure of the table used for storing controlling user attribute is the same as the structure of the table defined for storing supervisor's attribute. Note that, in this document, the terms owner and controlling user are used interchangeably.

- A target user is a user whom other users request an access of or establish a relationship. Since target users govern all corresponding information by themselves, target users require to define policies to manage given requests. This policy is called Target User Policy (TUP) and stored in a repository belonging to the target users. The structure of the table used for storing target user's rules is the same as the structure of the table defined for storing supervisor's rules. Besides, in order to being recognized by the system, target users carry several attributes to make unique identification information called Target User Attribute (TUA). The structure of the table used for storing target user attribute is the same as the structure of the table defined for storing supervisor's attribute. The target user has a role as an object and might inherited some functionality from the subject.
- A target resource is managed by a controlling user who is the resource owner. Unlike the model defined by Cheng et al. [83], the target resource keeps the policies to manage all access requests, although these policies are defined by its owner. This group of policies is called Target Resource Policy (TRP). The structure of the table used for storing target resource rules is the same as the structure of the table defined for storing supervisor's rules. If a controlling user defines a policy to protect belonging photos, it might be like: "accessing user AA wants to share her photo to users who share their photos to her.", so the rule might be like: "$\forall u_a$ *(*$\exists u_a$ *(name: AA), Ac (name: access),* $\forall t_r$ *(name: photo)* $\forall u_c$*), Ac (action: access),* $\forall t_r$ *(name:photo)* $\forall u_c$ *(name:AA)"*. Table 14 shows part of repository associated to resource (i.e. photos), which is an asset of user AA.

Record 1:

Column	Value
Rule#	Rule1
Rule Type	"C" = Complex
Subject Part	∀ u_a (∃u_a (name: AA), Ac (name: access), ∀ t_r (name:photo) ∃u_c)
Action Part	Ac (action: access)
Object Part – Target Resource/ Target User	∀ tr (name:photo)
Object Part – Controlling User	∃ u_c (name:AA)
Condition Part-Time	
Condition Part-Location	
Date & Time	5/12/2016-9:00
Order	0
Related To	N/A

Table 14 – Target Resource Repository

Record 2:

Column	Value
Rule#	Rule1
Rule Type	"S" = Simple
Subject Part	∃ ua (name: AA)
Action Part	Ac (name: access)
Object Part – Target Resource/ Target User	∀ tr (name:photo)
Object Part – Controlling User	∃ u_c
Condition Part-Time	
Condition Part-Location	
Date & Time	5/12/2016-9:00
Order	1
Related To	Subject

Table 14 – Target Resource Repository (cont.)

Target resource carries one or more attributes, which are called Target Resource Attribute (TRA), such as a name of resource, resource owner, and an identification number, which shows the uniqueness of that resource in the social network. The structure of the table used for storing target resource attributes is the same as the structure of the table defined for storing supervisor's attribute. The target resource has a role as object. The rest of the document resource and target resource is used interchangeably.

The proposed model supports two types of resources: simple and compound resources. The simple resource is only one-unit resource and does not include other resources. Therefore, to grant or refuse permission to access the simple resource, only checking the policies kept by

that resource will be adequate. Although in some cases, the social network needs to look up the supervisor policies which confine accessing to resources. In contrast, the compound resource is derived from other resources, and this might be controlled by the supervisor or by another controlling user who are fully or partially owners of a compound resource. In the former case, the resource is under the supervision of the social network, so access to that resource will be computed based on the accessing policies defined by supervisor and the accessing policies defined by the controlling user of each contributing resource. The latter case is similar to the former case except that the controlling user of the compounded resource might have a resource which is one of the contributing resource. This controlling user is also allowed to define the control access policies for the compound resource. As a result, a request to access the compound resource must be checked against several controlling users' access policies and supervisor policies.

As shown earlier, the relationship between the compound resource and its contributing resources may be reverse composition or reverse aggregation. Every controlling user or every owner of contributing resources are allowed to determine the type of relationship. This means that the owner of the contributing resource may have a policy which determines deleting the contributing resource, the compound resource will either be affected or not. For instance, based on Figure 8, suppose that resource 1 is a document which is a user guide for software AA. This document includes three chapters, two of them help users to log into the software application and work with the different screens and menus existing in the system. These two chapters are essential meaning that existence of the document is completely dependent upon the existence of those chapters and if one of those two chapters has been deleted, the document will be deleted as well. Another one chapter of the document is not essential. This

chapter provides the acknowledgement of the document that the document is still alive with or without the existence of this chapter, so deleting this chapter does not affect the entire document. In Figure 8, resource1 or document is contributed of resource2, resource3 and resource4, which these are chapters of the document. Relationship type between resource1 and resource2, as well as resource1 and resource4 are reverse composition because resource2 and resource4 are the fundamental chapters. Resource3 and resource1 are connected under the relationship type reverse aggregation, which means resource 3 is not the fundamental chapter.

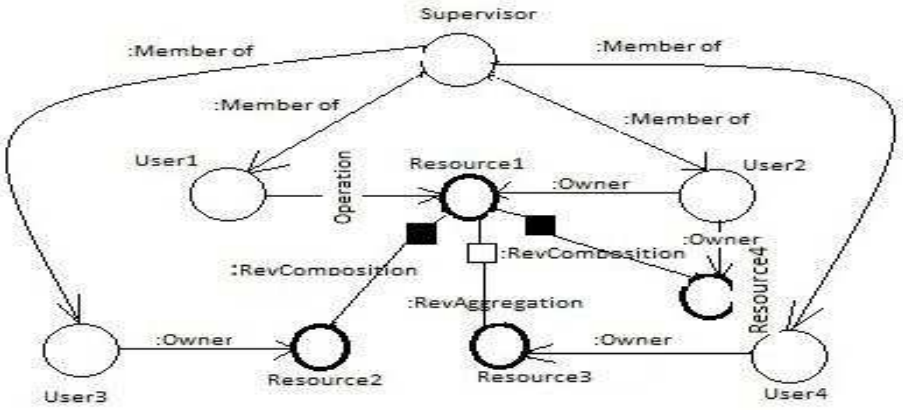

Figure 8- compound resource

- Attributes are attached to an entity to define the entity uniquely, so that entity can be distinguished in the social network. The following explains definition of attributes in each class.
  - S, O, R, A, and E are subjects, objects, relationships, actions and environmental conditions respectively. In our model, subject may be accessing user, controlling user, or supervisor. Object may be target user or target resource.
  - SA_k ($1 <= k <= K$), OA_l ($1 <= l <= L$), RA_m ($1 <= m <= M$), AA_p ($1 <= p <= P$), EA_n ($1 <= n <= N$) are defined attributes for subjects, objects, relationships, actions, and

environmental conditions respectively. The number of attributes attached to each entity may change during the life time of the social network.

- A represents the set of attributes for each entity, and A_i represents one attribute of each set. Each attribute encompasses the name and the value of the attribute, like: $A_i = (N_i: V_i)$.
- Attr(S), Attr(O), Attr(R), Attr(A), and Attr(E) are set of attributes assigned with subject, object, relationship, action, and environmental condition respectively.

  $Attr(S) \subseteq SA_1 x\ SA_2 x\ \ldots\ldots\ x\ SA_k$

  $Attr(O) \subseteq OA_1 x\ OA_2 x\ \ldots\ldots\ x\ OA_l$

  $Attr(R) \subseteq RA_1 x\ RA_2 x\ \ldots\ldots\ x\ RA_m$

  $Attr(A) \subseteq AA_1 x\ AA_2 x\ \ldots\ldots\ x\ AA_p$

  $Attr(E) \subseteq EA_1 x\ EA_2 x\ \ldots\ldots\ x\ EA_n$

- Environment condition includes data generated by a social network. This data, such as time and location, is changed rapidly related to conditions existing in the social network. Environment condition provides information about the current date and time as well as the location. Table15 shows environment information.

Type	Value
Date & Time	Current date and time
Location	Current location

Table 15 - Environment condition

- Actions representing operations (e.g. read, or edit) or accesses are initiated by the accessing user. As mentioned previously, an action is an entity of the proposed model and is almost initiated by a subject in the social network. The following table presents the actions covered by the proposed model.

Action	Description
Access	Accessing to a target user
Read	Reading information or resource
Write or Edit	Writing or editing information or resource
Delete	Deleting information or resource
Create	Adding new information's item or adding a part of resource associated with a target resource
Copy	Coping information or resource
Execute objects	Performing a function against a target resource
Grant permission to someone else to run an operation on an object	Accessing user is allowed to give permission to another accessing user in order to perform an operation on a particular object

Table 16 - Operations

- Different connections among the elements in Figure 3 may represent relationship, access request, attached policy, and attached attribute.
  - Relationship which has different types may connect supervisor to accessing user, supervisor to target user, supervisor to controlling user, supervisor to target resource, accessing user to target user, accessing user to controlling user, controlling user to target resource or repository, and target user to repository.
  - Access request is created when an accessing user creates an inquiry to access a target user or to run an action on a target resource.
  - Attached policy connects policies to an entity such as target user, target resource, or supervisor.

- Attached attribute connects attributes to an entity such as accessing user, target user, or relationship.

As mentioned before, five entities such as subject, object, action, relationship, and environmental condition have their own properties which store attributes; the first two are able to store rules in an associated repository. Attributes offer individual information of entity such as name, and unique ID, and rules are the transformed statements of policies defined by the subject or the object. It is required to define policy specification language for transforming policies to rules, accordingly. This model, unlike the model introduced by Cheng at al. [83], consists of subject-to-subject, object-to-object, and subject-to-object relationships in the social network, although this capture uses subject-to-object relationships for authorization purpose.

3.6 Graph Model

As shown in Figure 3, the social network can be indicated as a directed labeled graph. Generally, a graph is a collection of vertices which are connected by edges. There are two types of graphs, directed graphs and undirected graphs. Edges in a directed graph have arrows and each arrow only goes in one direction. Edges in an undirected graph does not have arrows and does not present any specific direction. In other words, a graph is a general data structure for storing related data which nodes denote objects or subjects and edges represent connections among nodes.

The graph of a social network may be modeled as a $G = <N, E, \Sigma, W>$ where

- N is union of a finite set of subjects (S) and a finite set of objects (O) in the system, represented as nodes on the graph. Each node carries one or more attributes.
- Σ is a finite set of relationship types, represented as labels for connection on the graph.

- W is a number that may be attached to a relationship connecting a subject to another subject or object. This number shows a percentage a user can trust other users to give them a particular permission on the social network.
- E ⊆N × N× Σ × W denotes graph edges and may be a set of existing connection, like: subject→ subject, subject→object, and object→object relationships.

Figure 9 represents a graph for a model which has been defined based on the ABAC model and the PBAC model.

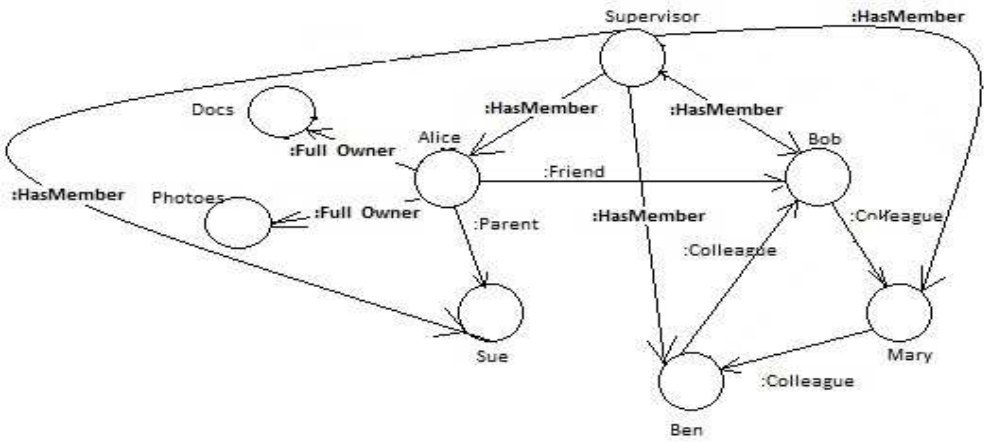

Figure 9 –Graph Model

3.7 Policy-Based Attribute Access Control Method

Before starting to describe the PBAAC method, considering an example which shows how someone can sign up, be a member of a group, and make a connection with other users on the social network would be helpful. Before starting the example, it's worth mentioning that for extracting information from the graph, we use several functions which will be described in chapter 6.

Example: if one signs up to a social network, one fills out a form and enters personal information which is required for the social network's authentication. Afterwards, the social network creates a credential for that user. According to the graph concept that we use to store the entities, behind the scene, the social network creates a node for the new user and attaches one or

more properties that keep data of attributes, such as first name, last name, and date of birth. The user can define a membership in an existing group or create a new group just by choosing a name for the new group and defining oneself as a member of that group. Technically, a group of nodes is defined by a label which is considered as an attribute of the entity. Labels are used to classify nodes. Later, we will show that labels may be used to define which groups and under which circumstances they are allowed to do an operation on an object. Meanwhile, we define functions determining who and under which conditions they are allowed to do an operation on an object. As we mentioned before, in the proposed model we define a supervisor who has a direct relationship with other entities in the social network. The following descriptions explain the entire process of the proposed model.

Description 1: supervisor (SpUser) has a relationship as "HasMember" with other users such as accessing user, target user, and controlling user on the social network (SN). The following shows this:

$\forall$ User ϵ SN, $\exists$ SpUser ϵ SN: SpUser -----:HasMember-----> User

Establishing a relationship among other users is possible and it is established when a user sends a request to another user for creating a specific relationship type. If the receiver accepts the request, the relationship will be established; otherwise that relationship will not be created. It is obvious that several different relationships may connect a user with another user.

$\exists$ $User_1$, $User_2 \epsilon$ SN, $\exists$ $r_1 \epsilon$ Relationship: $User_1$ -----:r_1-----> $User_2$

For instance, two users might settle on a relationship as friend and as colleague.

Description 2: Users can establish a relationship with their own resources. As described previously, there are three different relationship types between a user and associated resources, like FullOwner, PartiallOwner, and CollectedBy. The following shows this definition:

∀ Resource ϵ Asset, ∃ User ϵ SN:

User ----------:FullOwner----------> Resource ⋁ User ----------:PartiallOwner----------> Resource ⋁ User --------:CollectedBy--------> Resource

Asset is a set of resources existing in the social network. This means that all resources existing on the social network belong to a group called asset.

Description 3: Resources belonging to a user are assets of the social network. Therefore, the supervisor has a relation with resources. This relationship type is called "HasAsset". The following shows this definition:

∀ Resource ϵ Asset, ∃ User ϵ SN Resource € User, ∃ SpUser ϵ SN:

SpUser -----: HasAsset-----> Resource

Description 4: Environment condition has a relationship with the supervisor. This relationship type is called "ControlledBy". The following shows this definition:

∀ Environment Condition ϵ SN, ∃ SpUser ϵ SN:

SpUser -----:controlledBy-----> Environment Condition

Description 5: Resources existing in a social network and associated with a controlling user may include other simple resources owned by other users or may be a simple resource owned by just one user. Previously, we described these two types of resources. The following shows if a resource composes of other simple resources: (assume there are n resources and m users in the social network)

∃ res_1, res_2, …., res_i, res_n ϵ Assets, res_n ⊂ (res_1 ⋀ res_2 ⋀….⋀ res_i),

∃ $User_1$, $User_2$, ….., $User_m$ϵ SN, res_1€ $User_1$, res_2€ $User_2$……, res_i € $User_j$, res_n € $User_m$, i<=n-1, j<=m-1

(res_i ----------:RevAggregation----------> res_n ∨ res_i ----------:RevComposition----------> res_n)

Types of relationships exist between object and subject were defined before. It is obvious that the User can be supervisor, meaning, res_n may have relationship with supervisor rather than having relationship with an ordinary user.

Description 6: Users are able to define policies on behalf of their adjacent or trusted users. Regarding the definition of adjacent and trusted users, they have at least one directed or undirected relationship with the accessing user. If a directed connection exists between these two sides, the user will be allowed to define policies to take control of information and resources belonging to the adjacent or trusted user.

The supervisor defines general policies for the social network. The policies defined by the supervisor should not conflict with the policies defined by other users. Also policies defined by a user should not conflict with the policies defined by other users. As a matter of fact, if there are conflicts between policies defined by various entities, the decision engine who collects all policies contributing in an accessing request and generates the result is responsible to recognize these conflicts and solves these conflicts. The proposed model defines an efficient function, which is a part of the request engine, as solution for this problem. The function assigns a priority to each rule which is transformed format of a policy. This priority identifies a rule having higher priority overrides other rules having lower priority. This priority may be defined by timestamp or by other criteria were defined before. By comparing data attached to each rule, a process of the request engine choses the most appropriate rules for making decisions. Later in the request engine section, this function will be comprehensively explained.

The accessing user generates an inquiry to do an action against information or resources that are owned by another user. Using notation, the following shows this:

∃ Accessing user ϵ SN, Target user ϵ SN, Target resource ϵ Assets:

Accessing user ⊕ (Target user ⋁ Target resource)

This request may be accepted or rejected in terms of the target user or target resource access control policies. Sometimes supervisor's policies are included to compute whether or not giving permission to requestors.

As explained before, several parameters such as user's attribute, resource's attribute, access control policy, environment condition, adjacent and trusted users are used to determine who is allowed to access a resource. Up to this point, all parameters required in the proposed model to compute a received request will be defined.

Users define policies to protect their individual information and resources. All policies are defined in the English language. In order to store these policies into the social network, these policies must be transformed to a format, which is understandable, by the application and working with that transformed format is much easier than working with the format of a policy presenting in natural language. A transformation engine creates the simple format of those policies and stores them as rules on the social network. Rules are stored in the properties of the subject or object. The transformation engine will be covered in decision engine section of this document.

Description 7: In order to define the adjacent user with a particular relationship type, an algorithm has been defined to count the number of the specified relationship between the target user or controlling user in one side and accessing user on the other side. The following shows this:

Connections = $\sum r_i$, R={ r_i | is a particular relationship type between source and target user, or source and controlling user}

If number of connections is equal to the number determined the adjacent user which defined by supervisor, then the destination user is the adjacent user of the source user. Figure 10 shows an example; an adjacent user is one who has less than or equal to two connections with source user. In this figure Alice is a source user, so Bob and John are the adjacent friend for Alice, but Liz, Sue, and Ben are not.

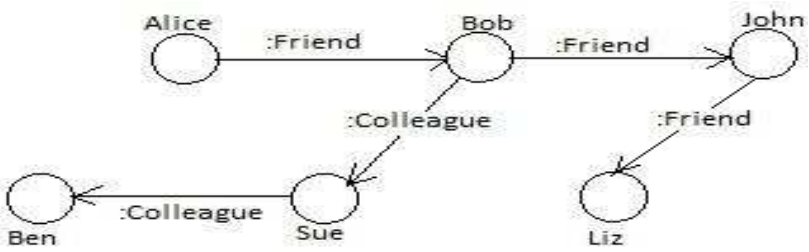

Figure 10 – Adjacent user

For defining who the trusted user is, a weight has been assigned with each relationship. By knowing the accessing user and the target user, or controlling user, the graph is traversed for all paths connecting the accessing user to the target user or controlling user; for each acyclic path the trust value will be computed. For making result and by following conservative approach, the trust value of each path will be compared and the minimum value will be selected as result. The following formula shows this:

$value_p = \prod \omega_i$, $W = \{\omega_i \mid \text{is a weight attached to each connection in a path}\}$

$result = \min \{value_p \mid p \text{ is a acyclic path connects source to destination}\}$

The result identifies whether the destination user is a trusted user or not. Figure 11 shows all paths existing between Hana and Al; each connection has its weight. With regards to this figure, there are two paths between Hana and Al. The first path which starts from Hana and ends with Al (Hana→Ted→Ava→Al) gives us the 72% trust and the second path which starts from Hana and ends with Al (Hana→Pat→Al) gives us 16% trust. Due to following conservative approach

for defining the trusted user, by comparing two numbers, the result is 16% which tells us Hana should not trust Al.

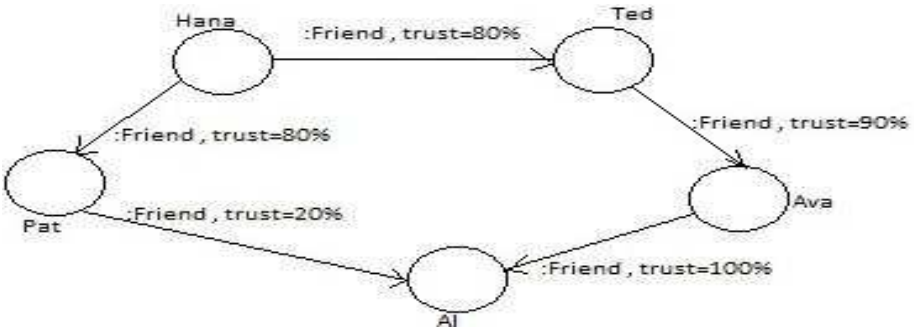

Figure 11 – trusted graph

3.8 Decision Engine

Decision engine is a principal part of the proposed model. As shown in Figure 12, the decision engine is a process standing between subject (e.g. accessing user) and object (e.g. target user or target resource). At first sight, the decision engine receives a policy or an inquiry as an input and converts it into a well-formed format. If the received sentence is a policy, the decision engine stores the output (i.e. rule) into a repository associated with the one who generates the policy, otherwise the decision engine evaluates the request converted format of an inquiry, and either accepts or rejects the request. The decision engine consists of two major engines, the transformation engine and the request engine. The decision engine accomplishes its task through the combination of transformation and request engines. Figure 12 provides a simple diagram of the decision engine.

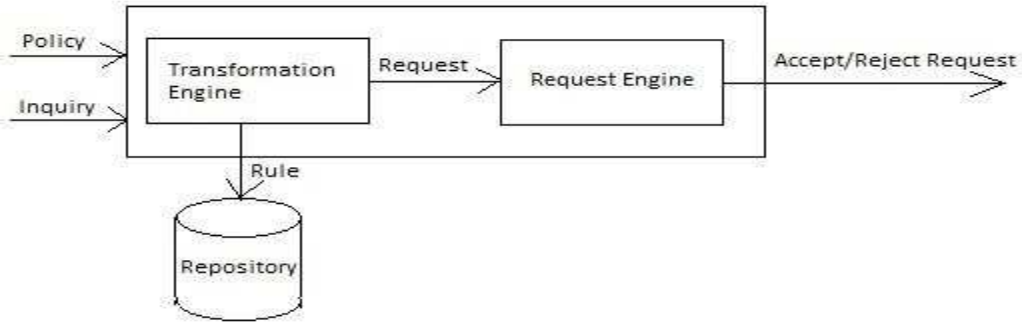

Figure 12 – Decision Engine

The transformation engine, as its name implies, uses natural language processing technique to generate an appropriate and efficient format for representing a sentence provided in the English language. An input sentence might be either a policy or an inquiry fed into the social network by one who needs to protect associated resource(s) or to access other resource(s) belonging to someone else. The appropriate format of the output generated by the transformation engine is called well-formed format. If the input is a policy, the well-formed format is called rule, and if the input is an inquiry, the well-formed format is called request. Under the transformation engine section, the workflow and the whole process of this engine will be explained thoroughly.

The request engine is a recipient of a request generated by the transformation engine. By gathering pertinent knowledge, which is coming from rules and attributes attached to entities interrelating with the request, the request engine evaluates the request and might be able to come to a decision in order to either accept or reject the request. Under the request engine section, more information of the entire process will be provided.

3.8.1 Transformation Engine

All policies enforcing access control and all inquiries representing request for accessing or doing an operation on resources are fed into social networks in a natural language (e.g. English language). Hence, by employing a natural language, users are able to define access control

policies that secure their assets, and they can generate inquiries to get permission to access a resource belonging to other users. This facility empowers users to feed their commands into social networks in a convenient way and enables users to protect their information and resources from unauthorized access effortlessly. On the other hand, social networking software applications cannot understand these commands. These commands are understandable by software application, providing they will be transformed into a simple format which is called a well-formed format.

In this work, the transformation engine converts a complex format into a well-formed format. Due to consistency between the formats generated by the transformation engine and the general definition of a well-formed format [89, 90]. It is possible to offer an appropriate notation for describing rules and queries with this format. This newly generated format is remarkably compatible with the syntactic structure of a policy or an inquiry. This means that words in a policy or an inquiry are appearing with an order such as subject, verb, and object. The syntactic order of elements existing in a rule or a request is as subject, action, and object as well. We assume that verbs in a policy or an inquiry and actions in a rule or a request are the same. Regarding the well-formed format, rules or requests are generated under the following format, which is a list with four elements: (subject, action, object, condition).

This format may be presented by a tree structure where root presents the action. The leaves represent the subject, the object, and the condition of the expression. The subject may represent the accessing user, the adjacent user, or the trustworthy user, and the object may represent the target user or the target resource. Target resource usually is followed by corresponding controlling user. The condition represents the environmental condition. Figure 13 shows this tree.

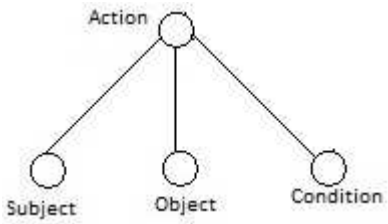

Figure 13 – Rule Tree

Due to the complexity of a policy, subject and object of a rule could be a well-formed format as well. This format is called a nested well-formed format and rules may look as follows:

((subject, action, object, condition), action, (subject, action, object, condition), condition).

The tree for representing the nested well-formed format is shown in Figure 14. In this tree, the root represents an action. the left most leaf is a subtree presenting all entities related to subject part, and the middle leaf is a subtree presenting all entities related to object part. The right most left is condition. Under this structure, all middle nodes show actions or relationships, and all leaves my show accessing users, adjacent users, trusted users, target user, target resource, controlling user, and environment condition. Resource owner is attached to the its own target resource.

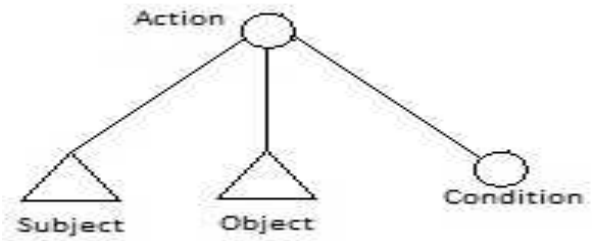

Figure 14 – Nested Rule Tree

In order to have a well-formed format, three criteria, listed in the following, must be met.

- Each list must be opened and subsequently closed
- Each element may be properly nested so it does not overlap

- Each element represents a subject followed by another element representing an action, and this will be followed by an element representing an object. Occasionally, this format may be ended with conditions.

Our model obeys these three requirements to generate the well-formed format [90]. By collaboration of various functions defined in the transformation engine, this engine receives an input and then converts it into a well-formed format.

Figure 15 shows a basic picture of the transformation engine. Based on the nature of an input, inputs are categorized into two groups as follows:

- Policy: it safeguards individual data resources against unjustified access. Policy is fed into the transformation engine, and then it is converted to a well-formed format called rule.
- Inquiry: it denotes a request for doing an action against data or resource. Inquiry is fed into the transformation engine, and then it is converted into a well-formed format called request.

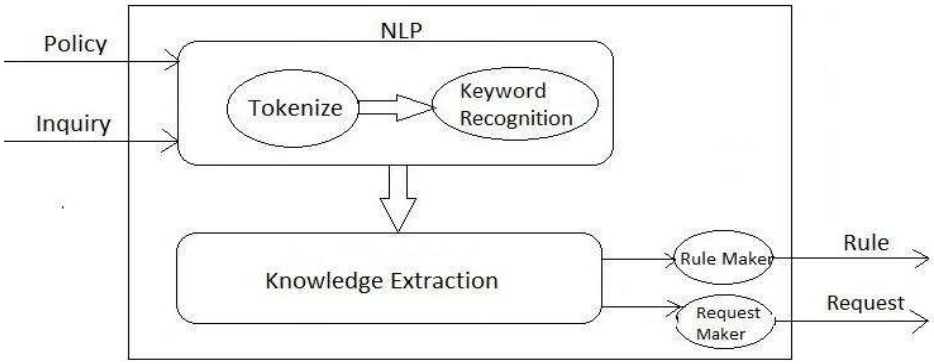

Figure 15 – Transformation Engine

At first glance, the output of the transformation engine would be a rule or a request. The rest of this section explains, in details, how a rule and a request will be generated.

The transformation engine is assumed to complete the conversion process in two phases due to having a robust engine. The two phases are described as follows:

- First phase: this phase generates a list of words existing in the input. Each word is followed by information describing that word. Thus, this process extracts words from input; these words carry the gist of the input. The type of received input has been defined when it was created by subject or object such as accessing user, target user, or controlling user, so the category of input is completely distinguishable by the *transformation engine.* This phase is started by performing the *Natural Language Parsing* (*NLP)* process.

  NLP receives input, and then distinguishes all words from the input sentence and adds grammatical information to every word. In order to achieve this goal, *NLP* uses three processes called *tokenize, keyword recognition,* and *knowledge extraction* processes.

  - *Tokenize* process provides a list of all terms or words existing in the input. *Tokenize* process distinguishes words from a given sentence by finding one or more spaces separating these words. This process ignores symbols such as ",", ":", ";", and parenthesis. Moreover, if this process finds alphanumeric characters connected to alphabetic characters, it separates numbers from alphabets and adds them as different words on to the list. If numbers are surrounded by an alphabet, the numbers will be removed from that word and only the alphabetic part will be added as one word on to the list. These two sub-functions help to mitigate or remove any mistyped word effect. Finally, the output of the *tokenize* process will be generated. This is a list of all words existing in the input sentence. Words in this list are ordered in terms of their appearing in the original sentence. The output might be a list as follows:

    $<word_1, word_2, word_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, word_n>$

    For instance, someone defines a policy in order to secure oneself from unauthorized access and she said "no one can poke me". Hence, the *tokenize* process generates a list as:

<no, one, can, poke, me>

- *Keyword recognition* process attaches some basic information into every word in the list provided by the *tokenize* process. *The keyword recognition* process receives the list of words from the *tokenize* process and searches every word in the *keyword* table to find out the grammatical point which has been provided for that word in the social network. The k*eyword* table as shown before stores grammatical information, specifically parts of speech, for every possible word. Previously, we described that the supervisor is responsible for maintenance and saving information into this table stored in the repository belonging to the supervisor. Besides, if one word of the list carries negative meaning intrinsically like (unable), or intuitively has negative meaning (like not or no), the negative meaning is reflected in the basic information. Every item of the list represents a word and its related information. Hence, output of the *keyword recognition* process may be a list of several elements. Each element may have a structure such as the following:

  <word, (parts of speech (major), parts of speech (detail), negative or positive meaning)>

  The following defines every item in this list:

  ✓ Word: this is a token of the input sentence.

  ✓ Part of speech (major): this may be noun, pronoun, verb, adverb, adjective, conjunction, preposition, article, or determiner.

  ✓ Part of speech (detail): this may be possessive, objective, subjective, reflexive pronoun, possessive adjective, possessive pronoun, auxiliary verb and main verb.

  Having the example provided for the *tokenize* process, the *keyword recognition* process generates the following list: *(< no, (determiner, subject, ¬)>, < one, (noun, subject, -)>, < can, (verb, auxiliary, -)>, < poke, (verb, main, -)>, < me, (pronoun, object, -)>)*

- *Knowledge extraction* process: The k*nowledge extraction* process receives the list generated by those processes. Sometimes, in an input sentence (i.e. policy or inquiry), there are side by side words which are syntactically related and they altogether represent one comprehensive concept in terms of the information extracted from each word. Hence, the *knowledge extraction* process collects these words and puts them on the list as one word and attaches proper grammatical information. Sometimes, an input sentence includes one or more accessing users in order to access one or more several objects (i.e. target user or target resources). In this case, the proposed method generates several rules or queries. The number of accessing users or objects defines how many outputs will be generated by *knowledge extraction*. In addition, using the *word-entity* table, the *knowledge extraction* process maps each word in the list onto the closest entity type and entity defined in the social network. All entities associated with the social networks are stored in this table. Earlier, the *word-entity* table and its data were described.

  The following shows the format of a list generated by the *knowledge extraction* process. The list includes several items including a word and its own basic information which provides remarkable knowledge for that word as follows:

  <word, (entity type, quantifier, entity, negative or positive meaning)>

  The following describes detailed information for every item of the list:

  - ✓ Word: it may be a token or concatenation of more tokens of the input sentence.
  - ✓ Entity type: based on the definition addressed previously in this document, it may be a subject, an object, a relationship, an action, or an environment condition.
  - ✓ Quantifier: this is a symbol and indicates the scope of a term or determines the quantity to which it is attached. Two famous examples of a quantifier are "∀" which

is a quantifier that represents “for all entities” and “∃” which is a quantifier that represents “there exists an entity” [5].

- ✓ Entity: based on the components of the proposed model defined earlier, this may be an accessing user, a target user, a target resource, a controlling user, a relationship type, an action type, or an environment condition.
- ✓ Negative or positive meaning: this is a symbol. If the word carries negative meaning, either explicitly or implicitly, symbol “¬ “will be used. Otherwise, symbol “-“ will be used to show positive meaning.

Subsequently, the *knowledge extraction* process creates a list by collecting the items described above. The following shows the output of this process:

{(T, X), (<word$_1$, (entity type, quantifier, entity, negative or positive meaning)>……..

<word$_n$, (entity type, quantifier, entity, negative or positive meaning)>)}

The final list includes two additional pieces of information about the input as follows:

- ✓ The first variable, *T*, represents the date and time when the input was fed into the social network site. It is called timestamp.
- ✓ The next one, X, represents the type of input. If X in “0”, it means this input is a policy. If X is “1”, it means this input is an inquiry.

To clarify the functionality of these processes, pursuing the following example will be helpful. For instance, a sentence “no one can poke me” was received as a policy input by the transformation engine at time “t_1”. In order to distinguish hidden information from that input, the *tokenize* process, which is part of the NLP, splits the input sentence into its constituent words. Then the *keyword recognition* process defines grammatical roles for every word and

the *knowledge extraction* process provides a short and clean list of words and corresponding entities. So the output of each process for the above example may be as follows:

Output of the *tokenize* process may be such as: *<no, one, can, poke, me>*

Output of the *keyword recognition* process may be written as: *(<no, (determiner, subject, ¬)>, <one, (noun, subject, -)>, <can, (verb, auxiliary, -)>, <poke, (verb, main, -)>, <me, (pronounce, object, -)>)*

Finally, output of the *knowledge extraction* may be presented as: *{(t_1,'0'), (<no one, (subject, ∄, accessing user, ¬)>, <poke, (action, poke, -)>, <me, (object, ∃, target user, -)>)*

Within the above three processes, using ontology concepts, it is possible to create an agreed-upon vocabulary for exchanging information. Ultimately, each vocabulary or word of the input will be mapped onto a specific concept on the social network. Without any doubt, having comprehensive information for each word of the input, decision of granting or revoking access to an object will be more precise.

- Second phase: based on the input type received by the first phase, either a policy or an inquiry, the transformation engine performs a *rule maker* or a *request maker* processes to generate a rule or a request, respectively.
  - *Rule maker* process receives a list of words and their related information of a policy and generates a well-formed format which is a rule. Users are allowed to save their defined rules into their rule repository. These rules will be used when the object receives inquiry for accessing or doing an action against user's resources. If the list generated by the *knowledge extraction* includes a relationship, depending on which parts of the sentence (i.e. subject or object or both part) include the relationship, that parts (i.e. subject part or object part or both) may be represented in a complex format. Hence, the rule will be

presented in a nested well-formed format meaning that the subject, object, or both may be represented in the well-formed format as well. Based upon the object, which can be a target user or a target resource included in the list, *rule maker* process generates a rule under one of the following formats. In both formats, the accessing user may be an adjacent or a trusted user.

- ✓ If the policy is protecting target user's individual information from illegal access, the rule may be presented as follows:

  {([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), (action (attribute_name: attribute_value)), ([quantifier] target_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)])]}- T

- ✓ If the policy is protecting controlling user's resource from illegal access, the rule may be presented as follows:

  {([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), (action (attribute_name: attribute_value)), ([quantifier] target_resource [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)] [quantifier] controlling_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)])]}- T

- o *Request maker* process receives a list of words and related information extracted from inquiry input. With regards to the action existed in the list, this process generates request

under one of the following formats. In both formats, the accessing user is an individual user and usually attribute-name just presents the name of the accessing user.

- ✓ If the inquiry is doing an operation against a target user's individual information, the request may be presented as follows:

  {([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), (action (attribute_name: attribute_value)), ([quantifier] target_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)])]}- T

- ✓ If the request is doing an operation against a controlling user's resource, the request may be presented as follows:

  {([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), (action (attribute_name: attribute_value)), ([quantifier] target_resource [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)] [quantifier] controlling_user [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions [(attribute_name$_1$: attribute_value$_1$), ..., (attribute_name$_n$: attribute_value$_n$)])]}- T

3.8.2 Request Engine

The request engine receives a request and makes a decision about whether or not give permission to accessing users in order to access a particular object. Figure 16 shows a basic picture of the request engine.

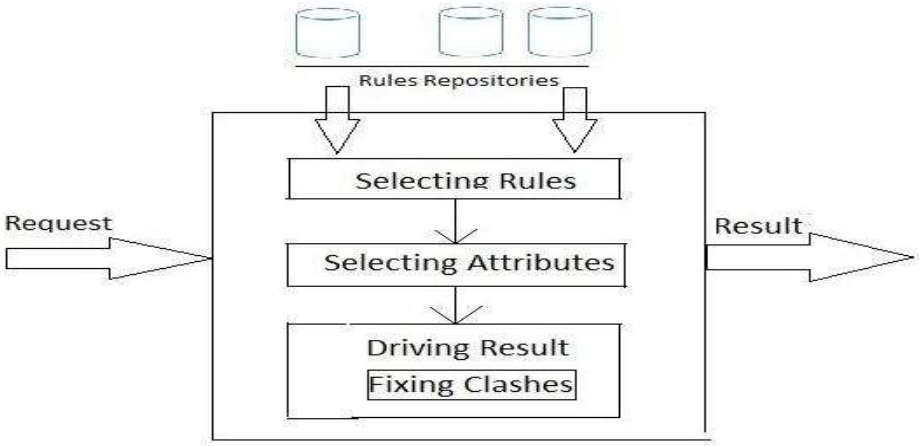

Figure 16 – Request Engine

An accessing user generates an inquiry to access or do an operation against a target user or a target resource. Since an inquiry for accessing a resource or information is received, the recipient, who is given the request, should respond to that request. In order to respond to the received inquiry, the decision engine will be performed for creating an appropriate answer for the inquiry. As mentioned before, the first engine of the decision engine is the transformation engine process. After performing the transformation process, the query, which is a transformed format of the inquiry, will be delivered to the request engine. The request engine collects all parameters playing key roles for evaluating the request and then concludes a result. The main parameters may be accessing user's attributes, action's attributes, relationship's attributes, environment condition' attributes, supervisor's rules, target user's rules, and target resource's rules. The request engine consists of three processes as follows:

- The first process is responsible for selecting appropriate rules from several different sets of rules defined by entities contributing in the request, and the supervisor. This process is called *selecting rules*.

- The second process is responsible for collecting all required attributes from entities contributing in the request, and the supervisor. Using the given request, this process recognizes necessary parameters. This process is called *selecting attributes*.
- The third process is responsible for deriving result from collected information. Moreover, in the case of arising confliction among selected rules, this process provides a solution and chooses the most appropriate rule which either accepts or rejects the request clearly. This process is called *deriving result*.

Selecting Rules

An object, a target user or a target resource, has various rules, each of which confines access to a target user or doing an operation against a target resource based on the type of the operation, the environment conditions' attributes, the target user's attributes, the target resource's attributes, and the accessing user's attributes (or a group which an accessing user is a member of). One or more rules which satisfy the request may be extracted from two different sets of rules. The two sets of rules could be as follows:

- The first set of rules may be reached from the repository belonging to the target user or the target resource to protect belongings against illegal access.
- The second one may be selected from a set of rules defined by the supervisor. That is defined as general rules to dictate policies for monitoring the access to the entire social network.

This process selects rules which protect corresponding object from specific action. This action has been mentioned in the received request.

Generally, selecting appropriate rules is burdened by a process of the decision engine called the *selecting rules* process. As mentioned before, the request engine receives the request, which serves as goal. Hence, the goal is specified and the request engine or *selecting rules* process

specifically, must specifically find a way to achieve that specified goal. A backward-chaining method accomplishes this. Indeed, there are two methods for achieving goals from many rules. These methods are [94,95]:

- Forward chaining: forward chaining is an inference method that may be described as working forward from data. Forward chaining starts with the given data and uses rules to extract more data until a goal is reached. In this case, an inference engine searches the rules until it finds one that meets the conditions. When such a rule is found, the engine concludes the consequent, and adds recent rule to the set of collected rules. The name of forward chaining comes from the fact that the interface engine starts with the data and reasons its way to the answer. Because the data determines which rules are selected and used, this method is also called data-driven [85].
- Backward chaining: backward chaining is an inference method that can be described as working backward from the goal(s). Backward chaining starts with a goal or list of goals and works backwards from the consequences to the antecedent to see if there are data available to support any of these consequences. This inference engine searches the inference rules until it finds one which has a consequent that matches a desired goal. If the antecedent (If clause) of that rule is known to be true, it is added to the list of goals (in order for one's goal to be confirmed, one must also provide data that confirm this new rule). Because the goal determines which rules are selected and used, this method is further called goal-driven [86].

To achieve the result, the *selecting rules* process looks for the rules that gratify the determined goal. Particularly, *selecting rules* process looks for the rules that can produce this goal. If a rule is found and fired, *selecting rules* process takes one or all rules which completely

or partially gratify the goal. *Selecting rules* process continues searching until there are no more rules matched. Set of extracted rule may be defined as follows:

$$\text{Set} = \{\text{rule}_x \mid (\exists\ \text{Ac} \ni \text{rule}_x,\ \exists\ n_j \in \text{Ac} < (n_1{:}v_1),\ldots, (n_k,v_k)> \wedge\ \forall \text{Ac} \ni \text{request},$$

$$\exists\ n_i \in \text{Ac} < (n_1{:}\ v_1),\ldots\ldots\ldots\ldots, (n_k;\ v_k)> ,\ \text{If}\ \ n_j == n_i\ \ \text{Then}\ \ v_j == v_i\ )\}$$

If set of extracted rule is null, the next two steps will be skipped. This means that the request is accepted.

Eventually, the request engine must extract one result from those selected rules and return that as a final result.

Selecting Attribute

After choosing appropriate rules, and by knowing the format of every rule, determining who are allowed to access or do a specific operation on the object under mentioned circumstances (i.e. environment condition) is accomplished. By observing the format of a rule, the first item represents one of the four pieces of information:

- Name of the subject or accessing user
- One or more attributes of subject or accessing user. These attributes may be used for grouping the subject or accessing user. Each attribute appears with related value
- Adjacent user of the target user or the controlling user
- Trusted user of the target user or the controlling user

As defined previously, adjacent and trusted users may be determined as rules defined by a target user, a controlling user, or the supervisor of the social network. In the proposed model, supervisor is supposed to provide definition for both adjacent and trusted users.

The *selecting attributes* process may generate two sets of rules. The first set includes rules that fulfill the request, and the second set includes rules that do not meet the request. The following list show how one or two sets are generated:

- Set 1: $\{rule_x \mid (\exists\ u_a \ni rule_x ,\ \exists\ n_j \in u_a < (n_1{:}v_1),\ldots\ldots\ldots\ldots, (n_k,v_k)> \wedge$

  $\forall u_a \ni request, \exists\ n_i \in u_a < (n_1{:}\ v_1),\ldots\ldots\ldots\ldots, (n_k;\ v_k)> ,\ If\ \ n_j == n_i\ \ Then\ \ v_j == v_i\ ) \wedge$

  $(\exists\ ec \ni rule_x, \exists\ v_t \in ec <time{:}v_t>, v_t == Current\ Time \wedge \exists\ v_l \in ec <location{:}v_l>, v_l == Current$ Location)}
- Set 2: $\{rule_x \mid (\exists\ u_a \ni rule_x ,\ \exists\ n_j \in u_a < (n_1{:}v_1),\ldots\ldots\ldots\ldots, (n_k,v_k)> \wedge$

  $\forall u_a \ni request, \exists\ n_i \in u_a < (n_1{:}\ v_1),\ldots\ldots\ldots\ldots, (n_k{:}\ v_k)> ,\ If\ \ n_j == n_i\ \ \wedge\ \ v_j\ != v_i\ \ ) \vee$

  $(\exists\ ec \ni rule_x ,\ \exists\ v_t \in ec <time{:}v_t>, v_t\ != Current\ Time \vee \exists\ v_l \in ec <location{:}v_l>, v_l\ != Current$ Location)}

Finally, two lists of rules can be generated. The first list includes one or more rules granting permission to the accessing user and the second list includes one or more rules denying permission to the accessing user. Either Set1 or Set2 may be null.

Note that the entire social network graph must be traversed in order to extract entities and their attributes. Using functions introduced in chapter 6, the entities and related information can be reached through the social network.

Deriving result

This process uses the lists generated by the previous process and concludes the result. If either Set1 or Set2 includes rules, there is no conflict and deriving result process either accepts or rejects the request in terms of which Set includes rule(s). Sometimes the rules defined by controlling user conflicts with the rules defined by the supervisor of the social network. The conflict is recognizable if both sets of rules include one or more rules. To resolve this issue, rules

existing in the two lists need to be compared one by one correspondingly in terms of the criteria addressed earlier in this chapter. This can be done using a process called *fixing clashes*. Fixing clashes engine uses the following three criteria to resolve the conflict.

- The first criterion is the timestamp attached to each rule. As mentioned before, since the transformation engine receives a policy and converts the policy into one or more rules, the transformation engine attaches a timestamp to the recently added rule. This timestamp represents the time when the rule is added to the social network.
- The second criterion defines rules, which override other rules. This can be defined by the target user, the controlling user, or the supervisor.
- The third criterion comes from the policy metadata defined by the supervisor. This basically determines who dominates who. In the proposed model, it is assumed that the supervisor dominates other users on the social network.

  Based on the above descriptions, this process chooses the most appropriate rule in terms of:

  Appropriate rule = $\{rule_i \mid (\forall rule_i, rule_j: rule_i \in Uc_a, rule_i \in Set1, rule_j \in Uc_b, rule_j \in Set2)$ If $rule_i.t > rule_j.t \vee rule_i$ overrides $rule_j \vee Uc_a$ dominates $Uc_b\} \cup \{rule_j \mid (\forall rule_i, rule_j : rule_i \in Uc_a, rule_i \in Set1, rule_j \in Uc_b, rule_j \in Set2)$ If $rule_j.t >= rule_i.t \vee rule_j$ overrides $rule_i \vee Uc_b$ dominates $Uc_a \}$

Fixing clashes selects the rule of the first list and compares it to the corresponding rule of the second list. Comparison may be done based on one or more criteria.

For simplicity, the proposed model is tuned for the first criterion so that *Fixing clashes* process orders rules in ascending or descending order based on the timestamp and related definition provided in the removing confrontation criteria table. Deriving result process looks at the rule of the first list and compares it to the corresponding rule of the second list. Both rules

have the equal position in these two lists. If *fixing clashes* process finds that conflicting rules having the same timestamp, the *fixing clashes* process will use the second criterion. Again if this process does not find any fact to choose the rule, this process uses the third criterion for comparison. *Fixing clashes* process performs this step until between two compared rules, it selects one, and consequently it removes the conflict. This means that in terms of the selected rule, the request engine accepts or rejects the request. If *fixing clashes* reaches to the end of the lists and does not find any rule, it cannot remove the conflict. Due to respecting to conservative solution, the request will be rejected. Ultimately, the request engine is able to provide an answer for a request.

3.9 Examples

In order to describe how the proposed model evaluates a request, several examples will be presented and discussed in the following.

Example 1: Ben pokes Alice.

In this example Bob wants to poke Alice. Previously, Alice defined a policy like "*only my friends are allowed to poke me*". Transformation engine translates this policy into a rule, which is an easy-to-use form and understandable by social network access control engine. To reach this goal, transformation engine sends a policy to NLP, meanwhile the type of input is completely clear for this process. Transformation engine's result is a list of words with comprehensive information. In order to prepare this list, NLP module uses *tokenize* process to split the given sentence into constituent words. The following list denotes the tokenize process result.

<only, my, friends, are, allowed, to, poke, me>

Keyword recognition process, which is a process of NLP module, adds grammatical information to every item in the list. This process uses a *keyword* table to find out parts of speech

for each word and *word-entity* table that maps words in a policy or in an inquiry into social network entities. The result of this step may be expressed as follows:

(<only, (preposition, adjective, -)>, <my, (pronounce, subjective, -)>, <friends, (noun, plural, -)>, <are, (verb, tobe, -)>, <allowed, (verb, main, -)>, <to, (preposition, second verb, -)>, <poke, (verb, second verb, -)>, <me, (pronounce, objective, -)>)

When *knowledge extraction* receives a list, regarding the words' parts of speech, *knowledge extraction* collects words, which appear in the list sequentially and have related meaning, and keeps them as an item in the new list. Hence, the result of the *knowledge extraction* process may be expressed as the following:

{(t_1,'0'), (<only, (subject, ∃, accessing user, -)>, <my friend, (subject, ∀, relationship, -)><are able to poke, (action, poke, -)>, <me, (object, ∀, target user, -)>)

In this example, the type of input sentence is a policy. Therefore, transformation engine calls rule maker to create a rule, which is converted form of a policy. The rule stored in the repository managed by Alice may look like:

"*< ∃ u_a (∀ u_a, Ac (relationship: friend+1), ∀ t_u (name: _self)), Ac (action: poke), ∀ t_u (name: _self) >- t_1*"

When someone tries to poke Alice, one creates an inquiry to access Alice who is object and target user specifically on the social network. Suppose there is an inquiry like "*Ben wants to poke Alice*". Like in the previous step, this inquiry must be changed into a simple format. Transformation engine uses several processes and converts the inquiry to request. The request may look like as the following:

"< ∃ u_a (name: Ben), Ac (action: poke), ∃ t_u (name: Alice)>"

The social network site sends the inquiry to an object who is supposed to manage a given action. In this example, the inquiry was created for accessing (i.e. poke) the object, which is a target user whose name is Alice.

Request engine takes the request. First, *selecting rules* process must find all rules that are defined by the target user or the supervisor, and meet all conditions or criteria in the request. These selected rules must meet several criteria such as the following:

- Type of the action which is requested by the accessing user to perform an operation on an object
- Environment conditions, which are mentioned in the query by accessing user, identifies conditions which have been fulfilled by given request. As mentioned earlier in this document, these conditions are stored and managed by supervisor.
- Adjacent users who have a specific relationship with the object under a certain distance, have been defined in the received request. Adjacent users are defined either by an object, or by a supervisor.
- Trusted users who are in the circle of trustee of the object or subject, have been mentioned in the given request. Trusted users are defined by either object, supervisor or both.

In this example *selecting rules* process returns only one rule, which is stored in Alice's repository. Regarding Alice's rule, only her friends can poke her. It means that every user on the social network site who has a direct friend relationship with Alice is allowed to poke her. Direct relationship means the length or distance of relationship is one. In this example, the accessing user is determined explicitly. In order to find out Alice's friends, *selecting attribute* process should search in terms of the requestor name. Due to existing merely one rule, *selecting attribute* looks for the accessing user attribute of the rule. It is obvious if there are more rules, *selecting*

attribute searches one or more attributes in terms of each rule. Afterwards, *deriving result* process, the third process of the request engine, searches a user whose name is "Ben" and has a relationship type as "friend" with "Alice". In this example, *deriving result* process finds a user whose name is "Ben" even though this user does not have a "friend" relationship with "Alice", in terms of Figure 9. So *making result* process returns nothing. Ultimately, the decision engine rejects the request meaning that Ben is not allowed to poke Alice.

Example 2: None of my colleagues are allowed to be friend with my kids.

In this example Alice defines a policy on behalf of her kids. In the proposed method, members of the social network are allowed to define policies on behalf of other members who are part of adjacent or trusted group. A member for whom another member defined the policy is allowed to accept or reject the policy. If this member accepts that policy, the policy will be transformed into a rule and will be saved in their repository. The acceptance of the policy will therefore affect the member's response. Otherwise, the policy will be rejected. In this example, Alice defines a policy like "*none of my colleagues are allowed to be friend with my kids*" and her kids accept it. Like the previous example, Transformation engine translates this policy into a rule. To reach this goal, NLP receives this policy. NLP module uses tokenize process to split the given sentence into constituent words. The following list denotes the tokenize process result.

< none, of, my, colleagues, are, allowed, to, be, friend, with, my, kids >

Keyword recognition process adds grammatical information to every item of the list. This process uses the *keyword* table to find out parts of speech for each word. Finally, the output of keyword recognition process is a list as follows:

(<none, (preposition, adjective, ¬)>, <of, (preposition, adjective, -)>, <my, (pronounce, subjective, -)>, <colleagues, (noun, plural, -)>, <are, (verb, tobe, -)>, <allowed, (verb, main, -

)>, <to, (preposition, -, -)>, <be, (verb, tobe, -)>, <friend, (noun, singular, -)>, <with, (preposition, -, -)>, <my, (adjective, possessive, -)>, <kids, (noun, plural, -)>)

After doing some processes by knowledge extraction, the result may be presented as the following:

{(t_1,'0'), (<none, (subject, ∃, accessing user, ¬)>, < my colleagues, (subject, ∀, relationship, ¬)>, < friend, (action, friend, -)>, <my kids, (object, ∀, target user, -)>)

In this example, the type of input sentence is a policy. Therefore, transformation engine calls rule maker to create a rule, which is converted form of a policy. The rule stored in the repository managed by Alice's kids (i.e. Sue) may be formatted as follows:

"*< (∄ u_a (∀ u_a, Ac (relationship: colleague+1), ∃ t_u (name:_self))), Ac (action: friend), ∃ t_u (name: Sue) >- t_1*"

Example 3: Alice says "my colleagues are allowed to edit my documents after office hours and at weekends"

In this example Alice defines a policy to give permission to her colleagues for editing her documents during certain time including after office hours and at weekends. Like the previous example, Transformation engine translates this policy into rule. To reach this goal, NLP, a process of transformation engine, receives this policy. Knowledge extraction's result is a list of words with its comprehensive information. In order to prepare this list, NLP module uses tokenize process to split the given sentence into constituent words. The following list denotes the tokenize process result.

<my, colleagues, are, allowed, to, edit, my, documents, after, office, hours, and, at, weekends >

Keyword recognition process adds grammatical information to every item of the list. This process uses a keyword table to find out parts of speech for each word. Finally, the output of keyword recognition is a list as follows:

(<my, (adjective, possessive, -)>, <colleagues, (noun, plural, -)>, <are, (verb, tobe, -)>, <allowed, (verb, main, -)>, <to, (preposition, -, -)>, <edit, (verb, , -)>, (<my, (adjective, possessive, -)>, <documents, (noun, plural, -)>, <after, (preposition, adjective, -)>, <office, (adverb, , -)>, <hours, (noun, plural, -)>, <and, (conjunction, -, -)>,<at, (preposition, -, -)>, <weekends, (adverb, , -)>)

After doing some processes by knowledge extraction, the result may look as the following:

{(t_1,'0'), (< my colleagues, (subject, ∀, relationship, -)>, < edit, (action, edit, -)>, <my documents, (object, ∀, target recourse, -)>, < (time: work hour+, time: weekends)>)

In this example, the type of input sentence is a policy. Therefore, transformation engine calls rule maker to create a rule, which is converted form of a policy. The rule stored in the repository managed by Alice may follow a form as follows:

"< (∀ u_a (∀ u_a, Ac (relationship: colleague+1), ∃ t_u (name:_self))), Ac (action: edit), ∀ t_r (name: document) ∃ u_c (name:_self), Ec (time: >office hour, time: weekend)>- t_1"

Example 4: Only my adjacent colleagues are allowed to read my documents.

In this example Alice defines a policy to give permission to her trusted colleagues to read her documents. Like the previous example, Transformation engine translates this policy into the rule. To reach this goal, transformation engine receives this policy and generates a list of words with its comprehensive information. In order to prepare this list, NLP module uses tokenize process to split the given sentence into including words. The following list denotes the tokenize process result.

< only, my, adjacent, colleagues, are, allowed, to, read, my, documents >

Keyword recognition process adds grammatical information to every item of the list. This process uses the *keyword* table to find out parts of speech for each word. Finally, the output of keyword recognition is a list as follows:

(<only, (preposition, adjective, -)>, <my, (adjective, possessive, -)>, <adjacent, (adjective, -, -)>,<colleagues, (noun, plural, -)>, <are, (verb, tobe, -)>, <allowed, (verb, main, -)>, <to, (preposition, -, -)>, <read, (verb, , -)>, (<my, (adjective, possessive, -)>, <documents, (noun, plural, -)>)

After doing some process by knowledge extraction, the results may be as follows:

{(t_1,'0'), (<only, (subject, ∃, trusted user, -)>, < my adjacent colleague, (subject, ∀, relationship, -)>, < edit, (action, edit, -)>, <my documents, (object, ∀, target recourse, -)>)

In this example, the type of input sentence is policy, so transformation engine calls rule maker for creating a rule which is converted form of policy. The rule stored in the repository managed by Alice might be as following:

"< (∀ u_a (∀ u_a, Ac (relationship: colleague+1), ∃ t_u (name:_self))), Ac (action: edit), ∀ t_r (name: document)>- t_1"

Example 5: Alice wants to share her photo with users who share their photos with Alice.

Regarding this example, Alice defines a policy to keep her photos safe from unauthorized access. She defines "someone whom I can access their photos, is allowed to access my photos". The transformation engine receives the policy. After tokenizing the input, it generates the following list:

< someone, whom, I, can, access, their, photos, is, allowed, to, access, my, photos >

After running knowledge recognition, we have:

(<someone, (pronoun-indefinitive, subject, -)>, <whom, (pronoun-relative, subject, -)>, <I, (pronounce, subject, -)>, <can, (verb, auxiliary, -)>, <access, (verb, main, -)>, (<their, (adjective, possessive, -)>, <photos, (noun-common, object, -)>, <is, (verb, tobe, -)>, <allowed, (verb, main, -)>, <to, (preposition, -, -)>, <access, (verb, main, -)>, (<my, (adjective, possessive, -)>, <photos, (noun-common, subject, -)>)

Knowledge extraction receives the list and generates a new list as follows:

{(t_1,'0'), (<someone, (subject, ∃, accessing user, -)>, <whom, (subject, ∃, controlling user, -)>, <I, (subject, ∃, accessing user, -)> <can access, (verb, main, -)>, <their photos, (object, target recourse, ∀, -)>, < is allowed to access, (action, access, -)>, <my photos, (object, target recourse, ∀, -)>)}

Finally, the rule might be shown like:

"< (∀ u_a (∃u_a (name: Alice), Ac (name: access), ∀ t_r (name:photo) ∃ u_c)), Ac (action: access), ∀ t_r (name:photo) ∃ u_c (name: _self)>- t_1"

Example 6: Alice wants to tag Bob in her photo.

Regarding this inquiry, Bob defined a policy which is no one is allowed tagging me in its documents. On the other hand, the supervisor defined a policy which mentions everyone is allowed tagging someone in its document. The decision engine generates the query and the rules corresponding to the inquiry and the policies defined by users. The following shows these outputs:

Alice's request: *"< (∃u_a (name: Alice), Ac (name: tag), ∃ t_u (name: Bob) >- t_3"*

Bob's rule: *"< (∄ u_a, Ac (name: tag), ∃ t_u (name:_self)>- t_2"*

Supervisor's rule: *"< (∀ u_a, Ac (action: tag), ∀ t_u >- t_1"*

When the request engine is given the request, the selecting rules process will be performed. This process searches all rules stored in the repository of the supervisor and the repository of Bob. Using the backward chaining mechanism, this process extracts rules and generates a list. Each rule mentions about “Alice” in accessing user’s name or mentions “tag” as action, no matter the rule accepts or rejects the request.

The selecting attributes is given the list. This process searches the attribute of entity participated in the rules and the request and certifies which rule accepts the request and which rule rejects the request. Based on this recognition, two lists will be generated. the first list includes rules accepting the request and the next list includes all rules rejecting the request. So list 1 consists one rule defined by the supervisor, and list 2 consist a rule defined by Bob.

The deriving results is given two lists. Using the rules contradiction table, this process concludes the result. Based on the data saved in this table, this process uses timestamp to remove the conflict. Compare to the supervisor’s rule timestamp, the Bob’s rule has been fed to the social network recently, so the Bob’s rule will be credited. Hence, the request will be rejected and Alice cannot tag Bob in her photo.

4. Enterprise Policy-Based Attribute Access Control Model

4.1 Introduction

Online communication sites need to share their resources among themselves. A user of one site should be able to access a resource of another site. Hence, under extend social networks model, each social network site must secure their assets against an access request received from other social network sites. The model defined in Chapter 3, consists of only one social network site in which all requests are initiated and computed internally. This chapter concentrates on the enterprise model that encompasses several social network sites, which wish to share their resources.

Several assumptions in the enterprise model are offered in this chapter. Firstly, no entity of a social network is allowed to define policies on behalf of another entity of another social network in order to protect its associated resources, although this feature supports in the fundamental model described in Chapter 3. Secondly, users from different social networks cannot be adjacent or trusted user to others. This means that adjacent or trusteed user can be define within one social network.

This chapter is considering how access control mechanism manages the entire enterprise model. The access control mechanism, which depends upon the users' requirements and distribution of the resources, plays an important role in the success of the model. The components of an access control mechanism may be separated and distributed within an enterprise model and may be different from one social network to another.

This chapter investigates the access control requirements within the enterprise model. Each social network site participating in this model has its own access control mechanism (i.e. decision engine) which evaluates a given request and provides an appropriate response. This

chapter describes a coordinator that is situated among all social network sites and is responsible to route a request and corresponding response among participants. The response to a given request is part of each social networks' responsibility. This approach is an appropriate method to manage the enterprise access control model in a dynamic way and is called Enterprise Policy-Based Attribute Access Control (EPBAAC).

4.2 Enterprise Policy-Based Attribute Access Control Model

EPBAAC represents the association of various social networks in which they wish to share resources and information, not only within one social network but also among several social networks. Regarding the NIST definition, "*an enterprise is a collaboration or coalition among several participants requiring them to share and manage their information*" [10]. These participants could be either social network sites or organizations. An enterprise model could be a heterogeneous or homogeneous collection of either social network sites or organizations. Hence, participants might have different or identical governing principles for sharing resources and information. The EPBAAC model supports homogeneous participants. In other words, each participant has its own identical governing principle. An enterprise model should have appropriate management for efficient resource sharing, using of policies and attributes, and forcing access control mechanism through the extended model.

Figure 17 presents a basic picture of the EPBAAC.

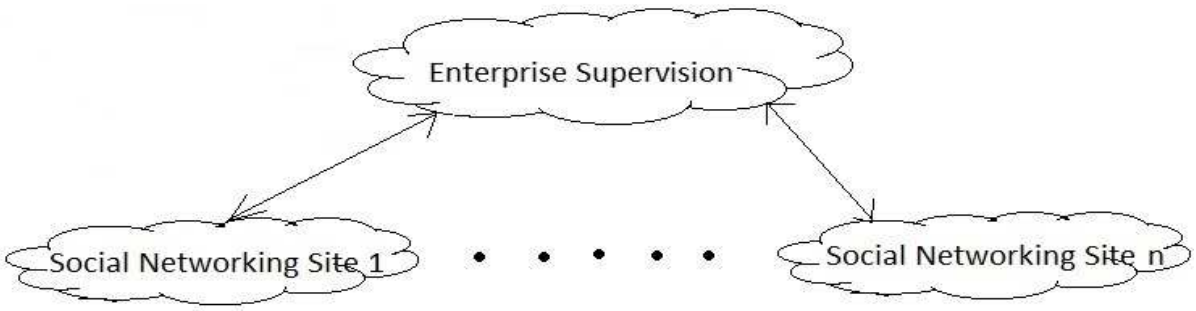

Figure 17- Enterprise Access Control Model

In this model, two noticeable sections can be seen. The first section is the management section, which presents enterprise supervision, and the second one creates the body of the enterprise model and represents the gigantic part of this model. This part is a collection of various independent participants, which might be either social network sites or organizations. The following gives a brief description for each section:

- The first section or management section is called enterprise supervision. At first glance, this section is the administrator of the enterprise model and oversees all events happening on the enterprise model. Briefly, the enterprise supervision receives a request from a participant (sender), performs some process on the request, and finally delivers the request to the particular participant (recipient). The enterprise supervision might use metadata to route messages among social network sites. In fact, enterprise supervision is a bridge connecting all participants. Later in this document, more explanation for the supervision responsibilities in the enterprise model will be provided.
- The other section is a set of social network sites. Every participant uses the PBAAC model to enforce access control for the corresponding resources and information, as well as to manage them. Chapter 3 provides the comprehensive definitions for the PBAAC model. Within the enterprise model, each participant is required to determine one or more policies to manage not only the internal requests (i.e. sent by a local user) but also the external requests (i.e. sent by users from other participants). Social network sites contributing to the enterprise PBAAC model are completely independent, and they have access to one another's resources through the enterprise supervision. Under our approach, there are two types of social networks. Figure 18 shows these two types. The following list provides definitions for both.

- The source social network is the one that initiates an inquiry to access information or do an operation on resources, which are governed either by the current or by another social network.
- The destination social network is the one that receives a request generated by the source social network.

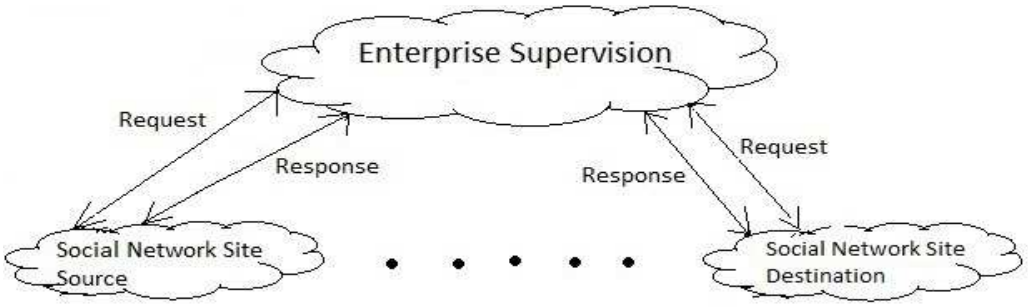

Figure 18 – Source, Target in Enterprise PBAAC

Besides the two important sections, there are some relationships, which are established between the enterprise supervision and other social network sites. These relationships identify each social network as part of the enterprise system, and sending and receiving an access request or a response by assistance of the enterprise supervisor. In the rest of this chapter, a complete explanation of the enterprise model will be provided.

4.3 Enterprise Policy-Based Attribute Access Control Model Components

Figure 19 shows the EPBAAC model.

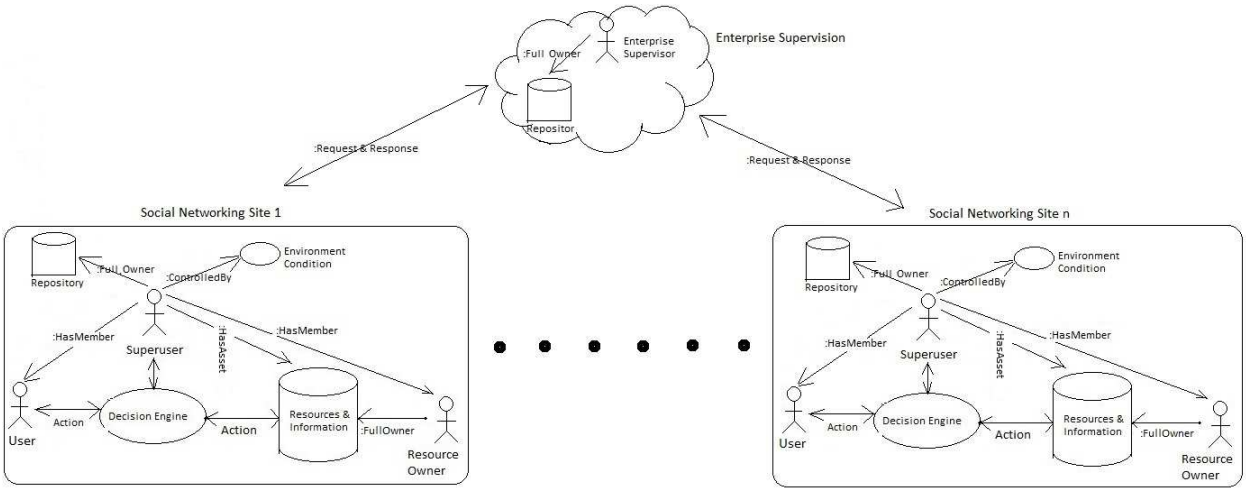

Figure 19- EPBAAC Model Components

The following listed items explain components of the EPBAAC model.

- Enterprise supervision includes several elements such as a supervisor and one or more repositories for storing access control policies and metadata. Enterprise supervision is a bridge among other social networking sites. Therefore, social networking sites can send a request to or receive response from other social networks. Through the enterprise supervisor, requests and responses travel through the social network sites as messages. A request could be an access resource or a list of the names of attributes associated with the accessing user who exists on the source social network. A response could be an answer to the request or a list of the values of attributes associated with the accessing user who exists on the source social network. Enterprise supervision is the administrator of the enterprise model and may define various policies to govern all operations on the enterprise social networks. These policies will be defined and covered as future work. An element of the enterprise supervision, called enterprise supervisor, is responsible to guide the enterprise model. The enterprise supervisor organizes all events, which happen across the enterprise scope. For instance, an

event could be a request raised by an accessing user in the source social network site to access a resource supervised by a target social network site. The supervisor of the enterprise model, like the supervisor of each collaborated social network site, has one or more attributes, which is called Enterprise Supervisor Attribute (ESPA); each attribute stores a part of the enterprise supervisor's information represented in Table 3, Chapter 3.

The enterprise supervisor is responsible for maintenance a large amount of data such as metadata (e.g. social networks' information) and environmental conditions.

The enterprise supervisor stores volatile data related to environmental conditions, which accommodates information of location or time. The structure of this table is the same as the structure mentioned in Table 15, Chapter 3. Developing environmental conditions controlled by the enterprise supervisor is not supported in this research, and it may be part of future work.

The enterprise supervision may have one or more repositories. One repository stores various information related to every social networking site. Table 17 shows this information. The enterprise supervisor is allowed to define which participant dominates other participants. This data item is used when rules defined by social networks have conflicts. This item helps choosing the rule determined by a social network, which has a higher priority to control the accessing of the associated resources. In order to show this, the last column of the table stores a priority number; a participant with smaller number is able to overshadow others with bigger numbers. Based on Table 17, SNS1 dominates SNS2. Several participants may have the same dominant value. In this case, the function, described in Chapter 3, uses another criterion for defining who is able to overshadow others.

Participants' ID	Participants' Name	Dominant Value
SNS1	Social networking site1	1
SNS2	Social networking site2	2

Table 17 - Participants' Information

Policies defined by the enterprise supervisor are converted to rules and then rules are saved into the repository owned and managed by the enterprise supervisor. The structure saving the rule is the same as the structure for saving rules defined by the supervisor existing in the PBAAC model in Chapter 3.

- Social network sites or organizations are the main body of the EPBAAC model. In Chapter 3, social network site architecture and its associated elements have been described thoroughly. As the social network described in Chapter 3, all social networks in the enterprise model have their own supervisor. Social networks contributing to the EPBAAC model are completely independent, and as mentioned before they have access to one another's resources through the enterprise supervision.
- Multiple connections exist between the enterprise supervision and every social network site. Every connection has been assigned a relationship type such as "PartOf". If a social network site generates a request to access a resource supervised by another social network site (i.e. target), the request must be routed to the enterprise supervision before received by the target social network site.

In the enterprise model, all policies and inquiries are in natural language. Hence, we require defining a process to transform policies to rules and inquiries to requests. By using the policy language definition, the transformation process will be accomplished.

4.4 Relationship Type Basic Notation

In the enterprise model, every social networking site has a set of relationship types, which has been comprehensively described in Chapter 3. Furthermore, the enterprise model includes a relationship type such as "PartOf" established between enterprise supervision and each social networking site. This relationship type is neither symmetric nor transitive.

Similar to the fundamental PBAAC model described in Chapter 3, we require keeping information for this relationship type. This information is called relationship metadata, and stored under the enterprise supervision section. The structure storing relationship metadata is identical to the structure storing relationship metadata described in the Chapter 3. The enterprise supervisor is the owner of this relationship metadata.

4.5 Graph Model

As shown in Figure 17, the enterprise model is a combination of two types of graphs.

- The first type is a connected graph. This graph shows the enterprise supervision including one or more repositories and environment condition. The enterprise supervision is connected to every collaborated social networking sites.
- The second type is a disconnected graph, which is a set of many connected graphs such as social networking sites or organizations. These social networking sites are completely independent and disconnected from one another. Each social networking site is a directed graph as explained in Chapter 3.

The graph of the enterprise model is a set of graphs, which has at least four items.

$G= \{EG, G1, G2, \ldots\ldots, Gn\}$, such that $\forall\ \alpha \in G, \alpha = <N, E, \Sigma, W>$

$\forall\ Gi \in G, Gj \in G: \exists\ EG \rightarrow Gi, EG \rightarrow Gj, \nexists\ Gi \rightarrow Gj$

- N is union of a finite set of subjects (S) and a finite set of objects (O) in the system, represented as nodes on the graph.
- Σ is a finite set of relationship types, represented as labels for connections on the graph.
- W is a number that is attached to a relationship connecting a subject to another subject. This number shows the percentage of trust one user has in another user in a social network site.
- $E \subseteq N \times N \times \Sigma \times W$ denotes graph edges and might be a set of existing connections such as subject→ subject, subject→object and object→object relationships.

Figure 20 represents a graph for the EPBAAC model. For developing the graph model, a graph database management system may be used as ideal solution.

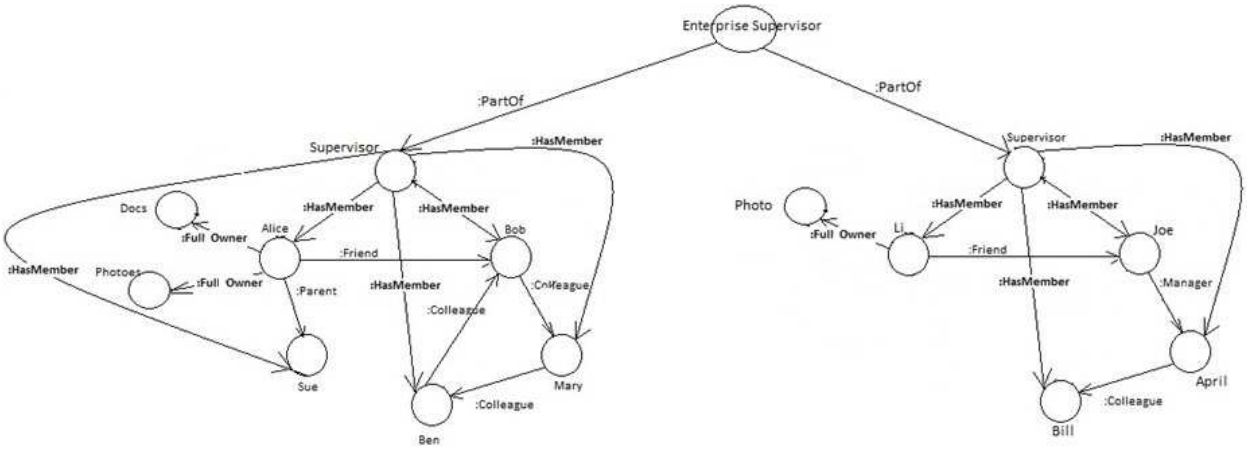

Figure 20 – Enterprise PBAAC Graph

4.6 Enterprise Policy-Based Attribute Access Control Method Description

Under the EPBAAC model section, like the fundamental PBAAC model, every element is allowed to define policies to protect individual information and resources. Once an element generates a policy, this policy is routed to the decision engine. The decision engine transforms a policy into a rule. Afterwards, the rule will be stored in the repository associated with the element generating the rule. This element is not allowed to determine a policy to protect associated resources on behalf of another entity existing on other social networks.

Some elements, namely accessing users, may generate a request for accessing or doing an operation on an object. In order to investigate how our approach processes a request, as mentioned before, we identify two different types of architecture for the EPBAAC model. One architecture describes a set of independent social networking sites, and the other describes the enterprise supervision. Regarding these two architectures, the EPBAAC model includes two types of operations:

- The first set of operations is internal operations, which performs on a social network locally; accomplishment of these operations is completely autonomous. This means that each social networking site is responsible for processing the received internal request and then deriving a result. The whole process is the same as the whole process explained in Chapter 3.
- The other set of operations is external operations, which are generated by the other social networks in the enterprise model. These operations are mainly accomplished by enterprise supervision. In other words, when a request is generated by a specific social networking site, it is delivered to the enterprise supervision. Since the basic information (i.e. metadata) of every social network is stored in the repository of enterprise supervision, the enterprise supervision compares the basic information to the received request in order to obtain the address of the receiver, and then sends the request to the social networking site destination (i.e. receiver). The enterprise supervision includes various items such as enterprise supervisor (ESpUser), environment conditions (EEnvCond), and the repository (ERep) for saving various metadata. So we have

  ∃ ERep ϵ ESN, ∃ ESpUser ϵ ESN:

  ESpUser ----------$\overset{:\text{FullOwner}}{\text{----------}}$> ERep

  ∃ EEnvCond ϵ ESN, ∃ ESpUser ϵ ESN:

ESpUser ----------:ControlledBy---------> EEnvCond

Based upon having two sets of operations, there exists two groups of inquiries in our model as follows:

- Inquiry for accessing or running an operation on a local object
- Inquiry for accessing or running an operation on a non-local object

The decision engine accommodated in each social network site distinguishes between these two groups of inquiries. Chapter 3 thoroughly explained all required operations for the first group of inquiries. The first group of inquiries will not be sent outside, but the second group will be sent outside of the local social network.

The second group will be subject to the following processes:

- A request is created by the transformation engine. The request is attached to the addresses of the source social network and the address of the destination social network. Because no social network in the enterprise model has information about another social network, the source social network sends the request to the enterprise supervision. Earlier, we showed that every social network existing in the enterprise model connects to the enterprise supervision under the relationship type "PartOf". Within an enterprise social network model (ESN), there is an enterprise supervisor (ESpUser) and one or more social networking sites (SNS) which are related to ESpUser as follows:

  $ESN = \{SNS_1, SNS_2, \ldots\ldots, SNS_n\} \cup ESpUser, \forall \alpha_i \in \{SNS_1, SNS_2, \ldots\ldots, SNS_n\}$:

  ESpUser -----:PartOf-----> α_i

- The enterprise supervision processes the request; based on the addresses attached to the source social networking site's request, the enterprise supervision sends the request to the particular destination social networking site. If a request wishes to access a resource of a

destination social network site, which does not exist in the enterprise supervision repository, this request will be rejected.

- In the final step, the decision engine existing on the destination social networking site receives the request. The request combines several items. The first item shows the original request and the rest of the items represent the addresses of the source and destination social networking site. The format of the request will be described under the decision engine section. The decision engine collects all rules and attributes of entities, which contribute to the request, and evaluates the request. Those rules may come from the target user, target resource, local supervisor of the target social networking site. Besides, the decision engine needs values of various attributes belonging to the local and external elements included in the request. An external element is the accessing user who is a member of the source social networking. The decision engine is also responsible to recognize contradictions between rules and removes them. Chapter 3 described the method for fixing rules contradictions.

4.7 Decision Engine

The decision engine is the principal part of the proposed enterprise model and consists of two major engines: the transformation engine and the request engine. Like the PBAAC, each social network participating in the enterprise PBAAC has its own decision engine. Under the PBAAC and the enterprise PBAAC models most functions defined in the decision engine are similar except the request engine. The decision engine within the enterprise PBAAC model performs various types of inquiries, which require adding some new functions in addition to the PBAAC model. This section explains these new functions.

4.7.1 Transformation Engine

The transformation engine receives policies, inquiries, and queries, converts polices into rules, and inquiries into queries. This section describes the processes of receiving an inquiry or a query.

If the transformed engine receives an inquiry, it generates a request, which is in a well-formed format. If this engine receives a request, this engine does not process this request and send it to the request engine for further processing.

Due to have a robust transformation engine, this engine runs its task in two phases, listed as follows:

- First phase: the transformation engine receives an input, which has a subject, a verb, and an object. The object may be either internal or external, defined as follows:
  - Internal object: if an inquiry contains an object located in the current social network, the object is an internal object. In this case, the subject (i.e. accessing user), the object (i.e. target user or target resource), and the input (i.e. inquiry), which is generated by internal subject and performed by the local decision engine are in the same social network.
  - External object: if an inquiry consists an object located in another social network, so that object is an external object. In this case, the social network where the requested object is located is not the same as the social network where the requestor is located and inquiry has been generated.

  The input could be an internal inquiry or external inquiry. The following describes both.
  - Internal inquiry is created by a subject (i.e. accessing user) located in a social network where the inquiry will be initiated and accomplished. This inquiry is in natural language format, so NLP process is called as described in Chapter 3 to convert the inquiry into the well-formed format. This inquiry may hope to access either an internal or an external

object. For both scenarios, a request will be generated and sent to the request engine settled on the source or destination social network.

- External inquiry is received by a social network (i.e. destination social network) which is different from a social network (i.e. source social network) where the original inquiry has been generated. Because external inquiry is in well-formed format, so the transformation engine will be skipped and the external inquiry, which it may be assumed as request will be sent to the request engine settled on the destination social network.

In order to process the internal inquiries, this phase starts by calling *Natural Language Parsing* (*NLP)* process, which addressed completely in Chapter 3. As mentioned previously, *NLP* uses two process, namely *tokenize* and *keyword recognition* process. The output of the *tokenize* and the *keyword recognition* process might be like the following:

$<word_1, word_2, \ldots\ldots, word_n>$

$<(<word_1, (parts\ of\ speech_1\ (major), parts\ of\ speech_1\ (detail), negative\ or\ positive\ meaning_1)>, \ldots\ldots\ldots, <word_n, (parts\ of\ speech_n\ (major), parts\ of\ speech_n\ (detail), negative\ or\ positive\ meaning_n)>)>$

Knowledge extraction process adds extra information to the list in addition to the same process, which has been addressed comprehensively in Chapter 3. The list generated by the *knowledge extraction* process in the enterprise model includes identification of either source or destination or both social network. The following shows the format of the list generated by *knowledge extraction*. The list includes several items such as the list of words and their associate basic information, which provides remarkable knowledge for corresponding word as follows:

- First item T defines the date and time when the input is fed into system. It will be called timestamp.
- Second item X defines type of the input, "*0*" for policy, "*1*" for inquiry, or "2' presents attributes list.
- Third item S defines source of the input, "0" represents internal input, or "*1*" represents external input.
- Fourth and fifth items address source social network and destination social network.
- Last item is a list of several elements, one of each might be such as follows:

  <word, (entity type, quantifier, entity, negative or positive meaning)>

  The following describes detailed information for every item of the list:

  - ✓ Word: it might be a token or concatenation of more tokens of the input sentence.
  - ✓ Entity type, based on the definition addressed previously in this document, this might be subject, object, relationship type, action, or label.
  - ✓ Quantifier is a symbol and indicates the scope of a term or determines the quantity to which it is attached. Two famous examples of quantifier might be "∀" that represents "for all entities" or "∃" that represents "there exists an entity" [5].
  - ✓ Entity, based on the components of our model, which we defined earlier, this may be accessing user, target user, target resource, relationship type, or action type.
  - ✓ Negative or positive meaning: this is a symbol. If the word carries negative meaning, either explicitly or implicitly, symbol "¬ "will be used. Otherwise symbol "-"will be used for showing positive meaning.

Subsequently, *knowledge extraction* process creates a list by collecting items described above. Therefore, we have:

{(T, X, S), (SourceName, DestinationName), (<word$_1$, (entity type, quantifier, entity, negative or positive meaning)>, ……..……, <word$_n$, (entity type, quantifier, entity, negative or positive meaning)>) }

To clarify the functionality of these processes, pursuing the following example would be helpful. For instance, a sentence "A wants to poke B who is a member of social network site 2" generates an inquiry in social network site 1. The following shows the output of each process:

Output of the *tokenize* process might be such as: *< A, wants, to, poke, B, who, is, a, member, of, social_ network_site2>*

Output of the *keyword recognition* process might be such as: {(*<A, (noun, subject, -)>, <wants, (verb, main, -)>, <to, (preposition, -, -)>, <poke, (verb, main, -)>, <B, (noun, object, -)>, <who, (pronoun, object, -)>, <is, (verb, auxiliary, -)>, <a, (determiner, object, -)>, <member, (noun, object, -)>, <of, (preposition, object, -)>, <social_network_site2, (noun, object, -) >)}*.

Finally, output of *knowledge extraction* might be such as: *{(t$_1$,'1', '0'), (SNS1, SNS2), (<A, (subject, ∀, accessing user, -)>, <wants to poke, (action, poke, -)>, <B, (object, ∃, target user, -)>)}*

- Second phase: if the input is an internal inquiry, the *request maker* process receives a list of words and related information extracted from inquiry input. Regarding the action existed in the list, this process generates request under one of the following format.
  - If the inquiry is doing an operation against target user's individual information, the request might be as follows:

{([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_2$: attribute_value$_2$)]), (action (attribute_name: attribute_value)), ([quantifier] target_user [(attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions (attribute_name$_1$: attribute_value$_1$)[, ……., (attribute_name$_n$: attribute_value$_n$)])]} - T - S, (Sname, Dname)

- If the input is doing an operation against controlling user's resource, the request might be as follows:

  {([qualifier] accessing_user [(attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_n$: attribute_value$_n$)]), (action (attribute_name: attribute_value)), ([quantifier] target_resource [(attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_n$: attribute_value$_n$)]) [quantifier] controlling_user [(attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_n$: attribute_value$_n$)]), [([quantifier] environment_conditions (attribute_name$_1$: attribute_value$_1$)[, ……., (attribute_name$_n$: attribute_value$_n$)])]} - T – S, (Sname, Dname)

Besides, each entity is supposed to protect their information by defining policies, which are transformed to rules and saved in the corresponding repository. Under the enterprise model, rule must mention which user from which social network is allowed to have access to this information. Rule maker generates a well-formed format for the rule. Target users and target resources are allowed to save their rules into the rule repository. These rules will be used when the target user or the target resource receives the inquiry.

The rule identifies access control for internal, external, or both accessing user. Based upon the object, which could be a target user or a target resource included in the list, this step generates a rule under one of the following formats.

- If the policy is protecting target user's individual information, the rule might be as follows: *{([qualifier] accessing_user [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)]), (action (attribute_name: attribute_value)), ([quantifier] target_user [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)]), [([quantifier] environment_conditions [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)])]}- T[, (Sname, Dname)]*
- If the policy is protecting user's resource, the rule might be as follows: *{([qualifier] accessing_user [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)]), (action (attribute_name: attribute_value)), ([quantifier] target_resource [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)] [quantifier] controlling_user [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)]), [([quantifier] environment_conditions [(attribute_name1: attribute_value1), ..., (attribute_namen: attribute_valuen)])]}- T[, (Sname, Dname)]*

4.7.2 Request Engine

As shown in Figure 14, Chapter 3, a request engine receives a request and then makes a decision whether or not to grant permission to an accessing user who may then access or run an operation on an object. Two types of request exist:

- Internal request: this request is generated by the decision engine located in the source social network site and received by the same social network site. Internal request may wish to access either an internal or an external object. Accessing to an internal object was described in the PBAAC model, Chapter 3. By the assumption of accessing to external

object, this request will be sent to the enterprise supervision to route to the destination social network site.

- External request: this request is generated by the decision engine located in the source social network site and received by the decision engine located in the destination social network site. The destination social network site extracts different sets of rules and attributes associated to the accessing user who is a member of the source social network site.

In the enterprise PBAAC model, there are two groups of rules.

- The first group is associated with the target users or target resources to protect their assets.
- The second one is associated with the local supervisor located in each social network site, which governs all local events.

A request, which accesses an external object, traverses a longer path than a request, which accesses an internal object. The request may pass, be accepted, or may not pass, be rejected, by several sets of rules: the enterprise supervisor, the destination supervisor located in the destination social network site, and the destination object located in the target social network site.

The transformation engine certifies that the request is in the well-formed format and passes it to the request engine. The request engine collects all necessary parameters. These parameters might come from the internal entity or the external entity. The main parameters are attributes of the accessing user, the environmental condition, or the rules of the destination supervisor, and resources. The request engine evaluates the request and then accepts or rejects the request. The request engine consists three processes as follows:

- The first process or s*electing rules* process is responsible to choose appropriate rules from large number of rules defined by entities contributing in accessing control evaluation. These entities could be internal, external, or both.
- The second process or *selecting attributes* is responsible to collect all required parameters from entities, which might be internal, external, or both. Receiving a request, this process recognizes required parameter.
- The third process or *deriving result* is responsible to conclude result from collected information. Moreover, in case of conflicts among rules, this process provides a solution as well.

The three processes will be described in detail as follows:

Selecting rules

A target user or target resource has various rules, each of which confines access to that target user or target resource in terms of the type of operation, environmental condition, target user's attributes, target resource's attribute, and accessing user's attribute (or a group of which the accessing user is a member). The selecting rules process selects related rules from different sets of rules. The two sets of rules could be as follows:

- The first set of rules was defined by the target user or owner of the resource to protect that target user or resource against illegal access.
- The second one might be selected from a large number of rules defined by the local supervisor existing on each social network site. These are defined as general rules to dictate policy to control the resources for the entire social network site.

Generally, a process of the decision engine burdens the reasoning system, which is defined for selecting appropriate rules. As we mentioned before, the request engine receives the request,

which may be pretended as goal. This term is used in rule engine system. Hence, the goal is specified and *selecting rule* process must find a way to achieve this specified goal. A backward-chaining system accomplishes this goal. This process was explained comprehensively in Chapter 3, so we do not explain it here. Finally, the *selecting rule* looks for the rules that match the specified goal. Chapter 3 described how rules are selected.

Selecting attributes

This process provides a mechanism for searching attributes corresponding to the entity, which participated in accessing control evaluation and could be internal or external entity. For extracting internal entities, Chapter 3 provided a comprehensive explanation. If this process requires more attributes of the external entity, this process generates a list of these attributes. Through the enterprise supervision, this process sends the list to the particular social network site, which has generated the request (i.e. source social network). This list may have the following format:

{(-,2, -), (Sname, Dname), (subject, (attribute_name$_1$: -), ……., (attribute_name$_n$: -))}

In this list, Sname represents the social network site generating the list of required attributes, and Dname represents the social network site hosting the specific entity and associated attributes. The social network site hosting the entity and its attributes, extracts the data from the attributes of the entity, updates the list (i.e. adding value for each attribute in the list) and through the enterprise, supervision returns the list to the social network site, which has started this process.

{(-,2, -), (Sname, Dname), (subject, (attribute_name$_1$: attribute_value$_1$), ……., (attribute_name$_n$: attribute_value$_n$))}

In this list, Sname represents the social network site updating the list (or the social network site generating the access request), and Dname represent the social network site generating the list (or hosting the specific object, which is given the request).

Similar to the PBAAC model, after choosing appropriate rules provided by the *selecting rules* process and by knowing the format of every rule, determining who are allowed to access or doing the operation on the object under particular circumstances, (i.e. environment condition) is accomplished. The detail information about this process will not be offered here, since the entire process has been addressed in Chapter3. Defining adjacent user and trusted user is completely out of the scope of the enterprise PBAAC model.

Ultimately, list of accessing user who are allowed to do operation on the object will be created. Besides, the last item of list denotes the environment condition of the enterprise social networking site. Environmental conditions as its name presents will be computed based on each social network site conditions and the enterprise conditions which usually might been changed time by time. Every social network site and enterprise model store the values of the environmental conditions. Without any doubt, for extracting entities and their associated attributes, we require to traverse the entire social network graph existing in the enterprise PBAAC model and by using functions provided in attribute-base function specification section we are able to extract the entities and related information.

Deriving result

Sometimes rules defined by controlling user clash with rules defined by the supervisor of the social networking site or the supervisor of the enterprise model. To solve this issue, we require to compare two sets of rules considering the criteria addressed in this chapter earlier. Solving issues

is accomplished by a process which is called *fixing clashes*. Fixing clashes engine uses three criteria to remove the conflict.

- The first criterion is the timestamp, which has been explained in Chapter 3.
- The second criterion defines rules, which overrides other rules.
- The third criterion comes from the policy metadata defined by the supervisor and the enterprise supervisor. This determines who dominates who. For instance, the supervisor of the enterprise model dominates the supervisor of each social networking site and supervisor of each social network site dominates other users on the social networking site. There is a hierarchal model for defining who dominates others.

Fixing clashes process selects a rule of the first list and compares to corresponding rule of the second list. Comparison may be achieved based on one or more criterion. This means if *fixing clashes* process found several conflicting rules having the same timestamp, then *fixing clashes* process uses the second or third criteria for comparison. For simplicity our model was tuned for the first criteria, so *Fixing clashes* process sorts sets of rules in ascending order, and selects the rule of the first list and compare to the corresponding rule of the second list. *Fixing clashes* process performs this step until finds a rule that overrides another rule which means conflict will be removed, or reaches at end of the lists and does not find any rule overrides others which means conflict will not be removed. In former case, the selected rule is able to provide an answer for the request which could be accepted or rejected. In the latter case, due to using conservative approach, the request will be rejected.

4.8 Examples

In order to describe how the enterprise model evaluates a request, the following example helps to denote all steps.

Example 1: An enterprise model includes two social network sites, social network A and B. A user "x" of social network B would like to read a document which is an asset under management of social network A and belongs to the user "y". The related inquiry might be like: "x wants to read document1 belonging to y of social network A" The user "y" determined a policy that only local user is allowed to access her resources. So it is expected that this inquiry will be rejected. By pursuing the proposed method, output of each step would be as follows:

- *tokenize* process : < x, wants, to, read, document1, of, y, on, social, network, A>
- *keyword recognition* process: *(<x, (noun- concrete, subject,-)>, <wants, (verb, main, -)>, <to, (preposition, -, -)>, <read, (verb, , -)>, <document1, (noun, singular, -)>, <y, (noun- concrete, object,-)>, <A, (noun- concrete, object,-)>)*
- *knowledge extraction* process: *{(t_1,'1', '1'), (B, A), (<x, (subject, ∃, accessing user, -)>, < read, (action, read, -)>, <document$_1$, (object, ∃, target recourse, -)>, <y, (object, ∃, controlling user, -)>)}*
- *request maker* process: *< ∃ ua (name:y), Ac (action: read), ∃ tr (name: document1) >- (t1, ("1"), (B, A)*
- request engine receives the request, because this request tries to access a resource located on another social network, so the request is sent to enterprise supervision. Enterprise supervisor searches in the repository, and then routs the request to the final destination.
- Social network A receives the request and local request engine makes decision like what we showed in chapter 3. Due to stored rule belongs to user y, this request will be rejected and then send the result to social network B through enterprise supervision.

5. Evaluation

5.1 NIST Standard

In any given social network, the number of users might be significant, the number of resources that must be protected might be in millions, and hence the number of access control policies that need to be defined might be in billions. If only one permission is incorrectly granted, a user will be given unsupervised access to information and resources which could jeopardize the security of the entire given social network.

Presently, security of information is an indispensable responsibility for all media keeping and sharing information with others. In practice, all applications employ access control methods to protect their information. Access control identifies activities of legal users and governs every attempt performed by these users to access a resource.

Access control systems are built upon three fundamental concepts: policies, models, and mechanisms. Access control policies are the first crucial requirement that define a method for managing access, and a user who is allowed to access information. Access control mechanisms translate a user access request to a system defined format like the well-formed format, which was defined early in this document. Access control models stand between policy and mechanism, and describe the security properties of an access control system.

The quality of access control policies affects the level of the security generated for the social network, so it is reasonable to define various metrics to evaluate the quality of access control systems and verify the access control properties.

Hence, four categories of metrics are defined by NIST [91]:

- Administration: defines cost, efficiency, and performance of an access control system.

- Enforcement: defines mechanisms and algorithms used in access control systems to enforce the embedded access control models and rules. These functions affect the access control's decision-making efficiency.
- Performance: defines the efficiency of the access control system.
- Support: defines the usability and portability of the system, although these are not essential.

Based on the definition provided by NIST [92], this chapter presents several metrics for evaluating the fundamental or enterprise PBAAC methods which we use in our approach to implement access control methods in a social network.

Hence, the following metrics are classified based upon the four categories mentioned above:

1. Ability to combine several related rules. The PBAAC decision engine is able to collect different access control rules, consolidate similar rules and derive a result under the specified condition. These rules can be defined by the controlling user, the target user, and the supervisor of the social network.
2. Ability to combine access control models. Under our approach, two access control models are combined, namely ABAC and PBAC models. By using the ABAC model, access constraints will be defined for each entity, and by using PBAC, policies enforcing access to a resource will be defined. Under our model, policies will be defined by controlling user, the target user, or the supervisor of the social network.
3. Ability to enforce the least privilege principle. Our model includes an entity as supervisor who is the administrator of the social network. The minimum privilege principle will be provided by rules defined by the supervisor. Our model accepts new users with various associated attributes. In order to access control mechanisms supporting the principle of the minimum privileges, constraints are saved as the attributes of a user.

4. Ability to resolve conflict rules. Rule conflicts appear when multiple access control rules are involved granting and denying permission simultaneously. They provide different results in terms of accepting or rejecting a demand to do an operation on an object. This means that rules clash over granting a user's access. To support conflict resolution, the decision engine provides a solution for removing conflicts and chooses the rule which fulfils the given request.
5. Ability to define rule-specific language. Every user is able to feed policies and inquiries in a natural language. The decision engine uses a specific syntax and schema to represent policies and inquiries which is much more efficient for disclosing information existing in them.
6. Ability to integrate or support identification and authentication of social networks. Under our approach, the attributes of users can be presented as the identification of the users.
7. Ability to decrease response time for granting or rejecting an access request. The next section of this chapter shows performance of the system based upon the response time of computing a request and generating a result. In order to achieve this, we run the application for different numbers of simple rules, different numbers of complex rules, and different numbers of attributes associated to entities. This section provides the analysis of several scenarios using different types of rules, the collecting results, and the drawing a graph.
8. Steps required for assigning and denying user capabilities into a social network. Under our approach, users can be granted new capabilities by adding extra attributes associated to the user or by defining new rules, which generally could be generated by the controlling user, target user, or the supervisor of the social network. Furthermore, these capabilities could be revoked by erasing the attributes or removing the rules from the social network.

9. Steps required for assigning and denying object access control entries into a social network. If an object is added to the social network, the owner of the object is allowed to define multiple policies to govern that. Moreover, the supervisor can add one or more policies to its repository to control the object. During the lifetime of the object, the owner of the object might require creating new rule(s), erasing some old rules, or updating the existing rule(s).
10. Support for separation of duty. One of the most fundamental access control methods is to protect information from unintended accesses. Some users are allowed to access an object and some users are not. Users who are duty-related to the objects are able to access the objects. This duty might be determined by the user's attributes or the object's policy. Our approach supports both mechanisms for defining the user's duty and how this duty can be related to a specific object.
11. Number of relationships to create an access control policy. Under our approach, there might exist several numbers of rules. Each rule includes the essential access control elements such as subject, action, and object; this is a simple form of rule. Sometimes we need to define a complex rule which does not fit the simple format. The complex rule includes a nested logical phrase of essential elements. This means that each element, except action element, can be a logical phrase. For instance, the subject might be a logical phrase including subject, action, and object. Access control rules are built by converting the user's policy. The decision engine converts natural language to a logical expression. If a user defines a complex policy, the rule will be complex as well.

5.2 Implementation

To validate our solution, the model was implemented based on various assumptions for two social networks including different number of members, the first social network with 20

members and the next with 50 members. The first assumption compares running time when the resource owner has been given a request for access a particular resource. The resource owner defined several rules having no-attribute, one, two, or five attributes to protect his resource. In Figure 21 and Figure 22, linear graph shows result of running time for each set of rules, where there are 20 and 50 members in the social network.

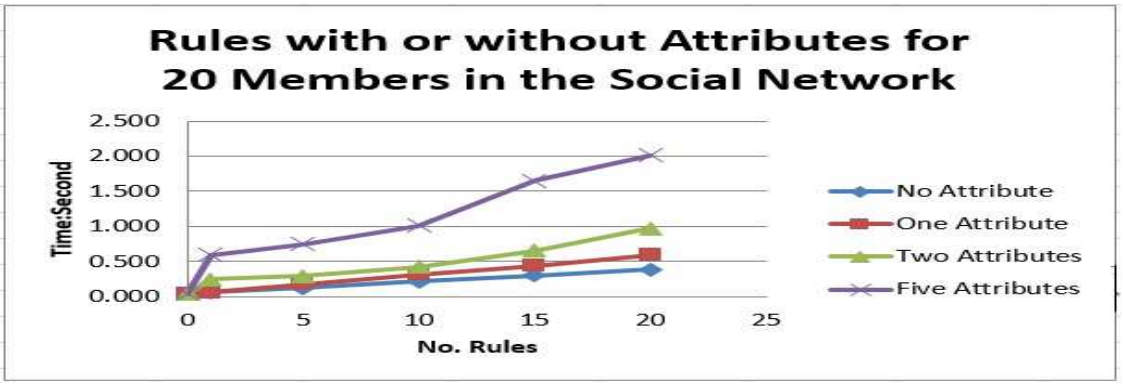

Figure 21 – Rules with/without Attributes for 20 Members in Social Network

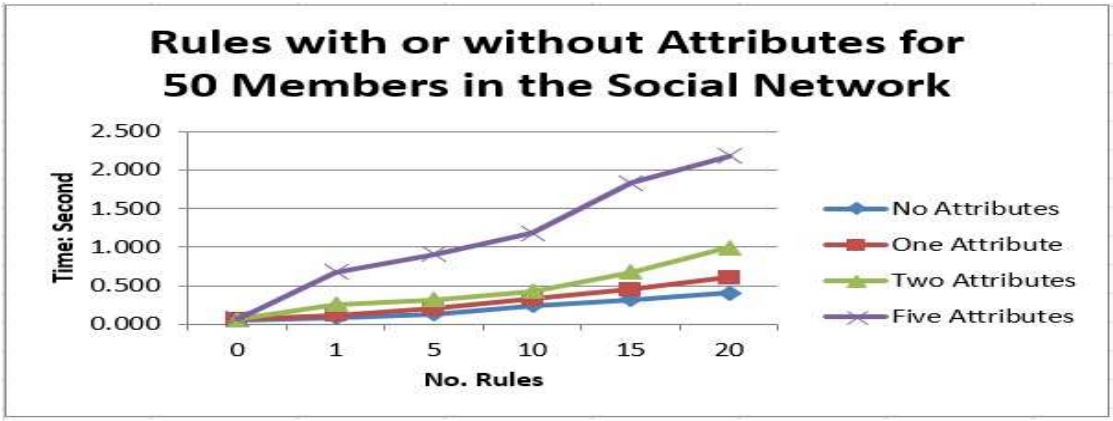

Figure 22 – Rules with/without Attributes for 50 Members in Social Network

As can be seen from two graph, we have the best running time when no rules exist for the particular resource. It can clearly be seen that when rules do not include any attribute, the running time will be slightly increased. Compared to the previous case, the linear graph shows a small increase if every rule includes only one attribute. Rules with two or five attributes need extra running time, notably when the number of rules will be increased. This graph presents

nearly rapid rise in case of having rules from fifteen to twenty. The following graphs compare running time of different assumptions for two social networks. The following graphs show a little increase where there are 50 members in the social network.

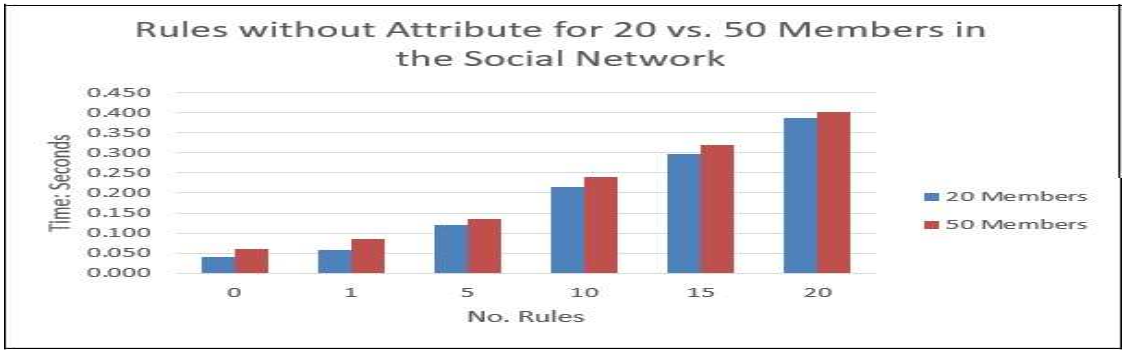

Figure 23 – Rules without Attribute for 20 vs. 50 Members

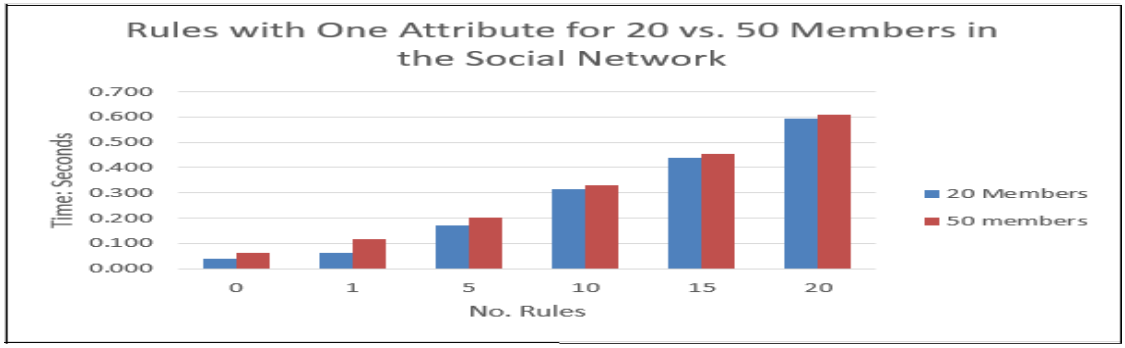

Figure 24 – Rules with One Attribute for 20 vs. 50 Members

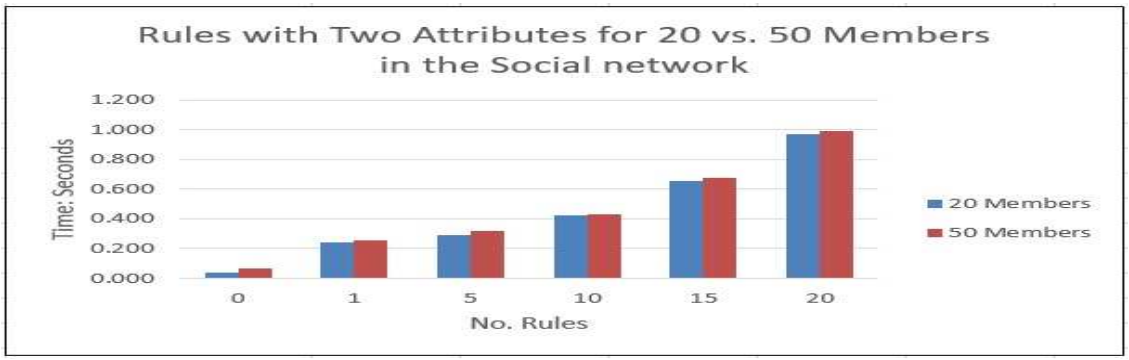

Figure 25 – Rules with Two Attributes for 20 vs. 50 Members

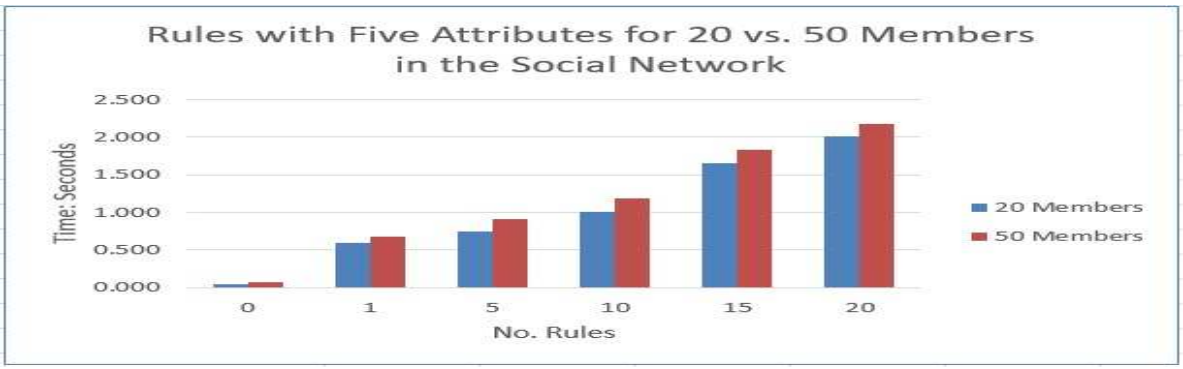

Figure 26 – Rules with Five Attributes for 20 vs. 50 Members

Figure 27 and Figure 28 show running time of the model with or without rules having conflict, for 20 and 50 members. Interestingly, the running time for conflicting rules increases sharply. This trend continues for up to twenty conflicting rules.

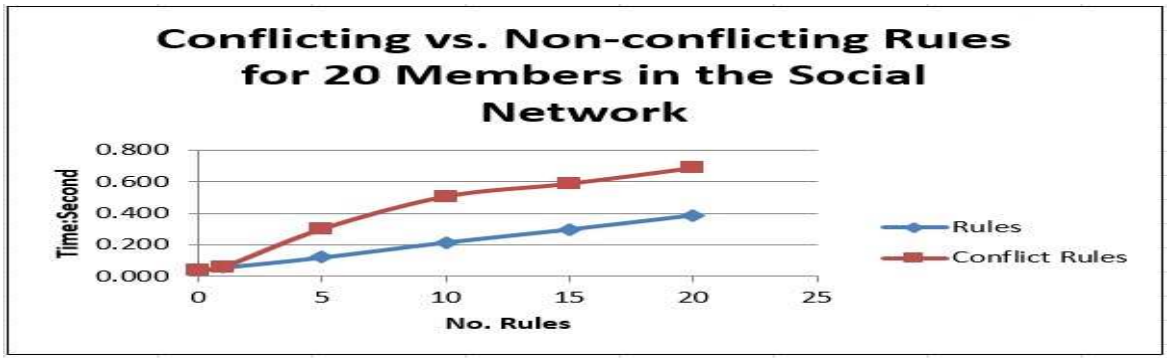

Figure 27 - Conflicting vs. Non-Conflicting Rules for 20 Members in the Social Network

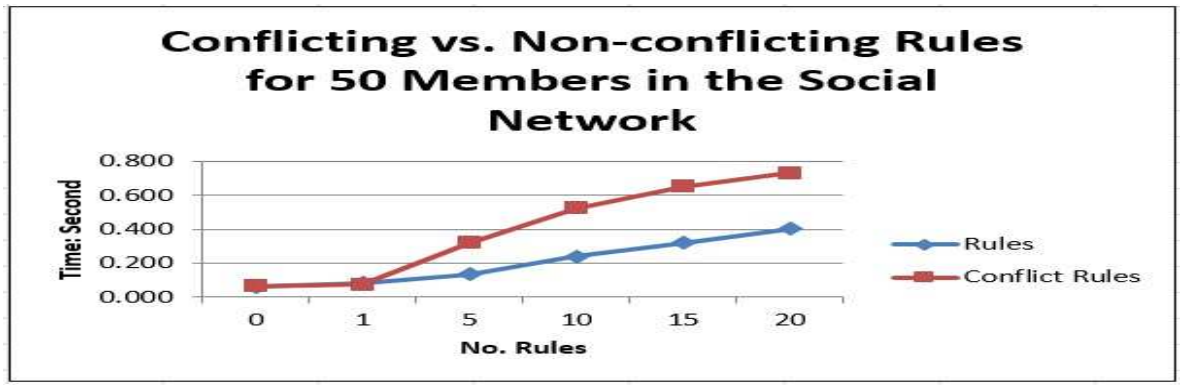

Figure 28 - Conflicting vs. Non-Conflicting Rules for 50 Members in the Social Network

Figure 29 compares conflicting rules with non-conflicting rules for two social networks having 20 and 50 members. This graph shows increase in the running time for conflicting rules.

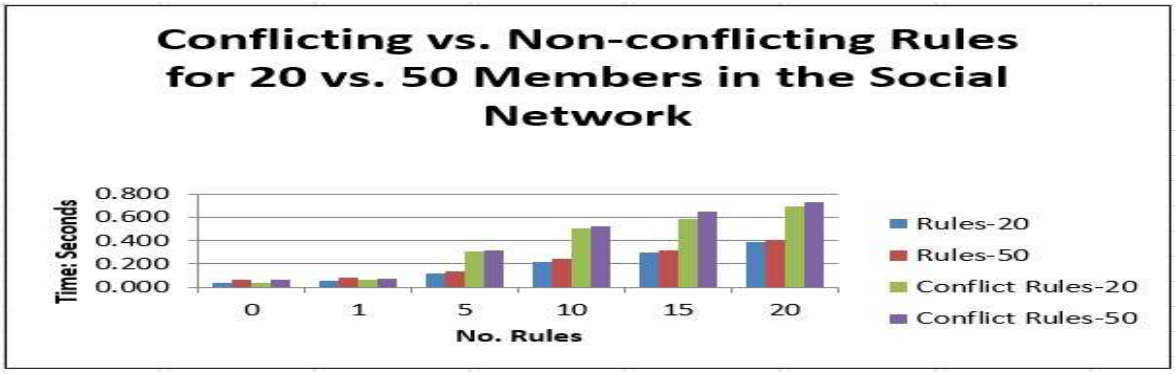

Figure 29 - Conflicting vs. Non-Conflicting Rules for 20 vs. 50 Members

Figure 30 and Figure 31 present running time when resource owner defines complex rules in two social networks having 20 and 50 members. As shown before, running times are not affected by the type of the rules (simple versus complex) for cases of one to five rules. For more number of rules, there is a dramatic difference in running time between the two types of rules. Specifically, running time increases sharply when there are 20 or more complex rules. Figure 32 gathers result derived from two social networks and shows in one graph.

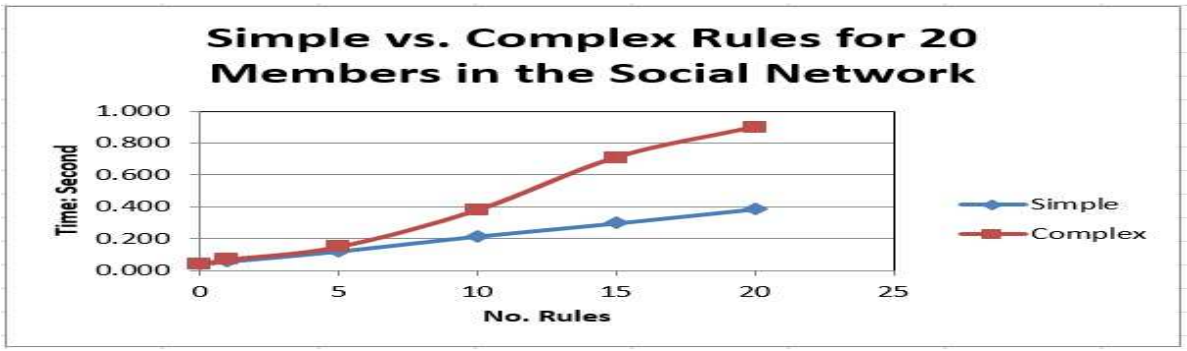

Figure 30 – Simple vs. Complex Rules for 20 Members

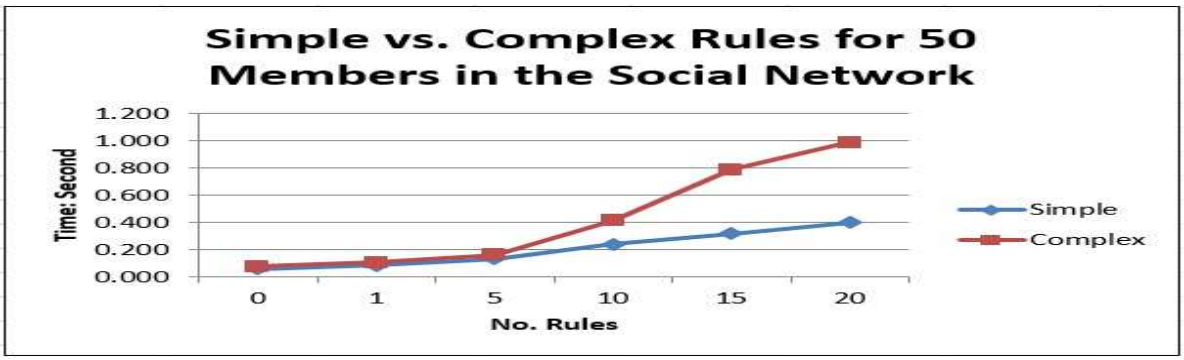

Figure 31 – Simple vs. Complex Rules for 50 Members

Figure 32 – Simple vs. Complex Rules for 20 vs. 50 Members

In summary, all graphs shown from Figures 21 to Figure 32 indicate that in order to have a better running time, resource owner should define simple rules including minimum number of attributes.

6. Policy-Based Attribute Access Control Functions and Algorithms

6.1 Functions

This section describes functions, which are used in our model particularly in the request engine process for traversing the graph of the model and extracting required information from the graph. The extracted information may be a specific user with attached attributes, a specific group of users, a specific relationship type, a particular resource, a group of resources, or a set of rules. The following describes these functions:

1) MemberOf: this function returns all users of the social network (SN). So this function will be defined as follows:

   MemberOf(SN) = □ {user | user ϵ SN}.

   For instance, $user_1$, $user_2$, … , and $user_n$ signed up into the $social\_network\_site_1$ so they are members of the social network site.

   MemberOf ($social\_network\_site_1$) = {$user_1$, $user_2$, …… $user_n$}

2) Assets: this function returns all resources existing in a social network. So for a social network site (SN), this function will be defined like:

   Assets (SN) = □ {resource | resource ϵ SN}.

   Resources can be in various types like files, devices, and etc. There is no doubt that every resource has an owner who defines access control policy for protecting that resource. So access to a resource is accomplished under the resource owner's supervision. A resource owner is a subject and can be either a controlling user or a supervisor.

   For instance, a set of assets existing in the $social\_network\_site_1$ is reached as follows:

   Assets ($social\_network\_site_1$) = {res_1, res_2, ……, res_n}

3) BelongingTo: this function returns all resources belonging to a particular controlling user or supervisor on the social network. As mentioned earlier, controlling users and supervisors can have one or more resources and they are the only members of the social network site who are allowed to supervise their resources. Hence, for a social network SN, a user (u) may have several resources. The following represents the definition:

BelongingTo (SN,u) = □ {resource | (resource € u) □ (u ϵ MemberOf(SN)) □ (resource ϵ Assets (SN)) }

For instance, $user_1$ shares associated resources such as book ($book_1$), and photos ($phot_1$) on the $social\_network\_site_1$, so BelongingTo returns all resources of $user_1$:

BelongingTo ($social\_network\_site_1$, $user_1$) = {$book_1$, $phot_1$}.

4) ParticipantOf: this function returns all components in the enterprise model. Participants may be either social network sites or organizations. Previously, we described that the information of all participants are stored in the enterprise supervisor's repository. So for the enterprise model (ESN) consisting multiple social network sites such as SN_1, SN_2,…… SN_n, this function will be defined as follows:

ParticipantOf (ESN) = {SN_i | SN_i ϵ ESN}.

For instance, enterprise_PBAAC encompasses various social network sites such as $social\_network\_site_1$, $social\_network\_site_2$, ……, $social\_network\_site_n$ so we have:

ParticipantOf(enterprise_PBAAC) = {$social\_network\_site_1$, $social\_network\_site_2$, ……, $social\_network\_site_n$}

5) RelationshipTypes: this function returns all relationship types among subjects (i.e. users) existing on a social network. It is obvious this function does not return relations between

users and their resources. Hence, for a social network site SN, this function will be defined as follows:

RelationshipTypes (SN) = □ {relationship | (relationship ϵ SN) □ ((user□user) □ (user ϵ MemberOf(SN))) }.

For instance, if "*friend*" and "*parent*" are the only relationship types in the social_network_site1, So:

RelationshipTypes (social_network_site1) = {"*frnd*", "*frnd-1*", "*prnt* "}

6) OwnershipType: this function returns the type of connection between subject and associated resources. Again controlling users and supervisors are subjects who have resources. Therefore, for a controlling user u and his resource (r) on the social networking site SN, this function is defined as follows:

OwnershipType (SN, u, r) = □ {Ownership | (r Є user) □ (u ϵ MemberOf(SN)) □ (r ϵ Asset(SN)) □ (u□r) □ (Ownership ϵ SN) }.

As mentioned earlier in Chapter 3, three types of ownerships (e.g. FullOwner, PartialOwner, and CollectedBy) are determined and it would be possible to define more in terms of requirements. For instance, $user_1$ shared the associated book ($book_1$) on the $social\_network\_site_1$, and $user_1$ owns the $book_1$ fully, so this function is as follows:

OwnershipType ($social\_network\_site_1$, $user_1$, $book_1$) = *"FullOwner"*

7) AllianceWith: this function returns relationship type between two particular entities. As mentioned earlier connection can connect subjects, objects, and subject and object. So the function will be defined as follows: if in social network site SN, u_1 and u_2 are connected by a relationship type, it can be shown like:

AllianceWith (SN, u_1, u_2) = □ {relationship | (relationship ϵ RelationshipType(t)) □ (u_1□u_2) □ (u_1 ϵ MemberOf(SN) □ u_2 ϵ MemberOf(SN)) }

For instance, if Alice is Bob's friend, this function returns the relationship type between these two entities as follows:

AllianceWith (social_network_site$_1$, *Alice*, *Bob*) = {"*frnd*", "*frnd*$^{-1}$"}

8) ComposedOf: this function returns all resources which all together generate another resource. As mentioned earlier, a resource may be composed of other simple resources. Hence, for a social network SN, a resource (r) may include several simple resources like r_1,r_2, …., r_k. The following represents the definition:

ComposedOf (SN,r) = □ {r | (r € u, r_1 € u_1 ,r_2 € u_2, …., r_k € u_p) □ (u, u_1 , u_2, ….,u_p ϵ MemberOf(SN)) □ (r, r_1,r_2, …., r_k ϵ Assets (SN)) □ (r C (r_1,r_2, …., r_k)) }

For instance, a book (book$_1$) includes several chapters, these chapters belong to different users in the social_network_site$_1$, so ComposedOf returns all chapters existing in book$_1$:

ComposedOf (social_network_site$_1$, book $_1$) = {chapt$_1$, chapt$_2$, chapt$_3$, ….., chapt$_k$ }.

9) ResourceToResource: this function returns relationship type between an including resource and an included resource. So the function will be defined as follows: if in a social network site SN, a resource r includes several simple resources like r_1,r_2, …., r_k, it can be shown like:

ResourceToResource (SN, r, r_i) = □ {relationship | (relationship ϵ RelationshipType(t)) □ (r□r_i) □ (r,r_i ϵ Asset(SN) □ (r C (r_i)) 1<=i<=k) }

For instance, if "*chapt$_1$*" is a part of "*book $_1$*" and has a connection as "*Revaggregation*" in the social_network_site$_1$, this function returns the relationship type between these two entities as follows:

ResourceToResource (social_network_site$_1$, *book $_1$*, *chapt$_1$*) = {"*Revaggregation*"}

10) Path_From__S_userToD_user: this function returns all paths between a source and a destination user. One path includes identical relationship type and represents a list of users existing in the path. Hence, for two users who S_user is a source user and D_user is destination user and they are members of a social network, this function will be defined as follows:

Path_From__S_userToD_user(S_user, D_user) = ∪ {path | (path = ∑ $relationship_i$ ϵ SN, ∧ S_user⇒ $user_1$ ⇒…….. ⇒ $user_j$ ⇒ D_user) ∧ (S_user, D_user, $user_j$ ϵ MemberOf(SN)) ∧ ($relationship_i$ ϵ RelationshipTypes(SN)) 1<=i<=n, 1<=j<=m}.

11) Labels: this function returns all labels on the social network. Labels will be defined for grouping entities in the social networking site. Labels' name must be started with *"lbl"*. If social network site SN has various labels such as l_1, l_2…… l_n. So for all users and all resources in the social network SN, we can define this function as follows:

Labels (SN) = □ {l | □ (□ (user ϵ MemberOf(SN)) □ (user ϵ l)) □ □ (□ (resource ϵ user) □ (resource ϵ Asset(SN)) □ (resource ϵ l)) }. For simplicity, a label is identified as an attribute of an entity in our model.

For instance, if "*person*" is defined to classify many users and this label is the only one in the social network site, so we have:

Label($social\_network\_site_1$) = {"*person*"}

12) GroupedBy: this function returns all entities grouped by a particular label. Under the social network site SN, if label (l) groups users, the function will be defined as follows:

GroupedBy (SN, l) = {users | (user ϵ MemberOf(SN) □ (l ϵ Label (SN)) }

If label l_r groups resources, the function will be defined as follows:

GroupedBy(SN,l_r) = □ {resource | (resource ϵ Asset(SN)) □ (l_r ϵ Labels (SN)) }

One entity can belong to more than one group or have several different labels. In this case, this label must group only either subjects or objects. This means that a particular label cannot be used for labeling a resource and a user simultaneously. For more clarification, in social_network_site$_1$, a label "*person*" has been defined to group many users such as user$_1$, user$_2$, user$_3$, user$_4$. So the result of this function may be as follows:

GroupedBy (social_network_site$_1$, "*person*") = {user$_1$, user$_2$, user$_3$, user$_4$}

If there is a label *"document"* for grouping multiple resources associated with different users in this social network site, this means these resources grouped by label *"document"* can be book$_1$, phot$_1$. So the function returns the resources grouped by "*document*" as follows:

GroupedBy (social_network_site$_1$, "*document*") = {book$_1$, phot$_1$}

13) AssociateWith: this function returns all labels grouping a particular user or a particular resource. If social network site SN has a member (u) who may be in one or more group labeled by l_1, l_2, ….. l_n, so this function will be defined as follows:

AssociateWith (SN, u) = □ {l_i | (l_i ϵ Labels(SN)) □ (u ϵ GroupedBy (SN, l_i)), 1<=i<=n}

If social network SN has a resource r which may be in one or more group labeled by l_1, l_2, …., l_n, so this function will be defined as follows:

AssociateWith (SN, r) = □ {l_i | (l_i ϵ Label(SN)) □ (r ϵ GroupedBy (SN,l_i)) , 1<=i<=n }

For more clarification, in social_network_site$_1$ there is a member such as *Alice* who is in two groups such as "*manager*" and "*employee*". So this function returns a list of groups of which Alice is member:

AssociateWith(social_network_site$_1$, "*Alice*") = {"*manager*", "*employee*"}

If social_network_site$_1$ consists resource *photo$_1$* which is in a group "image" the function returns the name of the group as follows:

AssociateWith(social_network_site$_1$, "*photo$_1$*") = {"*image*"}

14) InquiryToRequest: Since the social network receives an inquiry for accessing to a particular resource, this function will be performed. By performing several processes, which are part of the transformation engine, this function converts an inquiry to a request.

15) RunRequest: this function receives a request and makes a decision about whether or not grant permission to access a particular resource. This function implements the request engine.

16) RunInquiry: Since a user feeds an inquiry in social network to get access to a resource, this function will be performed. This function calls InquiryToRequest to generate the query, and RunRequest in order to either accept or reject the inquiry.

17) PolicyToRule: when a user defines a policy to govern associated resources, this function will be performed. Running several processes, which are part of the transformation engine, this function converts a policy to a rule and saves the rule into the repository associated with the user who generates the policy.

18) AdjacentCompanion: this function checks whether a user is adjacent to a particular user or not. By using Path_From__S_userToD_user, this function finds all paths between the source user and the destination user, and then by considering the adjacent user definition, this function reveals the destination user is either adjacent to the source user or not.

19) TrustworthyUser: this function determines whether a user is trustworthy by a particular user or not. By using Path_From__S_userToD_user, this function finds all paths between the source user and the destination user, and for each path, generates a result or value of trustworthiness. The value of trustworthiness has been generated by multiplying weights attached to each relation in the path. If the result meets the trusted user definition, then the

destination user is a trustworthy user for the source user, otherwise the destination user is not a trustworthy user for the source user.

20) GetAccessingUser: this function which is a part of the request engine, extracts accessing user from the request.

21) GetTargetUser: this function which is a part of the request engine, extracts target user from the request.

22) GetTargetResource: this function which is a part of the request engine, extracts target resource from the request.

23) GetAction: this function which is a part of the request engine, extracts action from the request.

24) RulesOf: this function returns rules stored in a particular entity's repository. The entity may be target user, target resource, or supervisor. Hence, for a social network site SN, a subject S, an object O, and an entity y, we have:

RulesOf(SN,y) = □ {rule | □ y: ((y ϵ S) □ (y ϵ O)) □ (rules € y) }

For instance, $user_1$ has several rules, namely $rule_1$, $rule_2$, $rule_3$ on the $social\_network\_site_1$, so RulesOf returns rules as follows:

RulesOf ($social\_network\_site_1$, $user_1$) = {$rule_1$, $rule_2$, $rule_3$}

25) RulesOfAction: this function returns rules that consider a specific action and are stored in an entity's repository. The entity may be a target user, a target resource, or the supervisor. Hence, for a social network site SN, an entity y, a subject S, an object O, and an action a, we have:

RulesOfAction (SN,y, a) = □ {rule | □ y: ((y ϵ S) □ (y ϵ O)) □ (rule € y) □ (a ⊗ rule)}

26) RulesOfEnvironmentCondition: this function returns rules that consider a specific environmental condition and are stored in an entity's repository. The entity may be the supervisor. Hence, for a social network site SN, an entity y, and an environment condition ec, we have:

RulesOfEnvironmentCondition (SN, y, ec) = □ {rule | □ y: (rule € y) □ (ec ϵ rule)}

27) AttributesOf: this function empowers us access to all attributes associated with a particular entity on a social networking site. As mentioned earlier, supervisor, accessing users, controlling users, target users, target resources, relationships, environmental condition, and actions are allowed to have attributes. Hence, for a social network site SN, a subject S, an object O, a relationship R, an action A, an environmental condition E, and an entity y, we have:

AttributesOf(SN,y) = □ {(attributeName:attributeValue) | □ y: ((y ϵ S) □ (y ϵ O) □ (y ϵ R) □ (y ϵ A) □ (y ϵ E)) □ (attribute ϵ Attr(S)) □ (attribute ϵ Attr(O)) □ (attribute ϵ Attr(R)) □ (attribute ϵ Attr(A)) □ (attribute ϵ Attr(E)) }

For instance, $user_1$ has several attributes such as userID, name, and age in the $social\_network\_site_1$, so AttributesOf returns as follows:

AttributesOf ($social\_network\_site_1$, $user_1$) = {(userID:$value_1$), (name:$value_2$), (age:$value_3$)}.

28) ValueOfAttribute: this function provides access to the value of the specific attribute associated with a particular entity in the social network site. Entities, namely accessing users, controlling users, target users, target resources, relationships, environment condition, and actions are allowed to have attributes. Hence, for a social network site SN, an entity y, a subject S, an object O, a relationship R, an action A, an environmental condition E, and an attribute a we have:

ValueOfAttribute(SN,y,a) = {attributeValue | (□attributeValue ϵ (attributeName:-)) □ (□ y: ((y ϵ S) □ (y ϵ O) □ (y ϵ R) □ (y ϵ A) □ (y ϵ E))) □ ((attribute ϵ Attr(S)) □ (attribute ϵ Attr(O)) □ (attribute ϵ Attr(R)) □ (attribute ϵ Attr(A)) □ (attribute ϵ Attr(E))) }

For instance, $user_1$ has an attribute $attr_1$ such as userID in the $social\_network\_site_1$, so ValueOfAttribute returns a list as follows:

ValueOfAttribute ($social\_network\_site_1$, $user_1$, $attr_1$) = {userID: 'user 001'}.

29) RemoveRuleClashes: this function collects all rules, which conflict, and based on the policy defined in the removing confrontation table, chooses one or more criteria. Based on the criteria, this function compares the rules and then selects one of them, which fulfills the request. For simulation, in order to remove conflicted rules, timestamp was chosen. Algorithm *result_inference* provides a comprehensive description.

6.2 Algorithms

This section presents algorithms, which have been defined as analyzing the proposed model.

Decision engine

Decision engine is the main process of our approach. After receiving an input which may be a policy or an inquiry, decision engine converts the input into a simpler format by running the *transformation engine* process. If the input is a policy, the converted format will be a rule which will be stored in the repository associated with an entity generating the policy. If the input is an inquiry, the converted format will be a request and will be sent to *request engine* for either accepting or rejecting.

Algorithm *Decision_engine*

Input: *S*: sentence

Output: *answer*: answer may be either "*accepted*" or "*rejected*"

1) Call *transformation_engine*

2) Call *request_engine*
3) Return *answer*

Transformation Engine

Transformation engine process receives a sentence which may be a policy or an inquiry. The goal of defining this engine is converting input, which is a sentence in natural language, into a format which is more recognizable and understandable by the rest of processes. Transformation engine runs several processes.

The first step of this process creates a list of words in the input. Each word has been attached to some grammatical points.

The next step defines a simple format to present the critical information encompassed in the input. After performing various functions mentioned in this algorithm, the input is transformed to the simpler format. This format looks like (subject, action, object, condition). A policy will be converted to a rule and an inquiry will be converted to a request. The rule is saved on the repository of the initiator or repository of the resource associated with the initiator. The request is sent to the request engine for further processing.

Algorithm *transformation_engine*

Input: *S*: sentence

Output: *Rule, Request*: these are a list of subject, action, object, and conditions

1) Call *NLP*
2) Call *knowledge_extraction*
3) If input is policy
   3.1) Call *Rule_maker*
   3.2) Store *Rule* on the repository
4) Else
   4.1) call *Request_maker*
   4.2) return *Request*

Natural Language Process

NLP algorithm runs *keyword_determination* and *keyword_recognition* to generate a set. Each item of the set presents a list including a word of the sentence *S*, and grammatical information related to the word.

Algorithm *NLP*

Input *S*: sentence

Output *W_info*: list of elements in *S*, followed by parts of speech to each element

1) Call *keyword_determination* (input: sentence S, output: list of words existing in S *W_list*)
2) Call *keyword_recognition* (input: a list of words included in the sentence *W_list*, output: list of elements in *W_list*, followed by parts of speech to each element *W_info*)
3) Return *W_info*

Keyword Determination Process

Keyword_determination extracts words from input, generates a list from these words. If sentence S includes $w_1, w_2, \ldots, w_n$, so the format of output or *W_list* may look like: $< w_1, w_2, \ldots, w_n >$

Algorithm keyw*ord_determination*

Input *S*: sentence

Output *W_list*: list of words in S

1) Read a word from *S*, store into *w*
2) Do loop until w is not End-of-Line {
   2.1) add w to *W_list*, separating with comma
   2.2) read a word from *S*, store into *w*}
3) Return *W_list*

Keyword Recognition Process

Keyword_recognition attaches grammatical points to the word of the list, and creates a list of words accompanied by parts of speech. There is a table, which has been stored on the supervisor repository, describing syntactic and semantic features for a word. Each word will be attached two grammatical definitions, major and minor parts of speech. The major parts of speech in English

are noun, pronoun, verb, adverb, adjective, determiner, preposition, conjunction, article, and interjection. The minor parts of speech may describe the detail of major parts of speech: Noun may be abstract, collective, common, and concrete. Pronoun may be possessive, objective, subjective, reflexive, intensive, demonstrative, interrogative, and indefinitive. Adjectives may be possessive. Verb can be auxiliary and main. Moreover, knowing basic grammatical terms such as subject, object, and action helps to understand some of the principle words in the sentence. Hence, this algorithm looks for words in the table and finds corresponding parts of speech, major and minor. In addition, identifying negative and positive meaning for each word, helps us to determine that this sentence will refuse or accept a request. Generally, all words have been ordered in terms of basic grammatical terms. This means that each item of the list starts with a subject, followed by an action and finally ended with an object of the sentence and perhaps one or more conditions. Providing *keyword_recognition* receives *W_list* as input, the output or *W_new* looks like:

{< w_1,(parts of speech(major), parts of speech(minor), neg/pos meaning)>, <w_2, (parts of speech(major), parts of speech(minor), neg/pos meaning)>, ..., <w_n, (parts of speech(major), parts of speech(minor), neg/pos meaning)> }

Algorithm *keyword_recognition*

Input *W_list*: a list of words included in the sentence

Output *W_info*: list of elements in *W_list*, followed by parts of speech to each element

1) Read an element from *W_list*, store into *w*
2) Do loop until *w* is not End-of-List
   2.1) look up *w* in *keyword* table
      2.1.1) If found: fetch parts of speech (e.g. principle, major, minor, neg/pos meaning), and attached to the word. Afterwards, add to the *W_info* as an element of the list
      2.1.2) Else: Error "wrong keyword"
3) Endloop

4) Return *W_info*

Knowledge Extraction

Knowledge extraction process uses the list generated by NLP process and consolidates some elements of the list in order to generate a short and simple formatted list. Due to the order of presenting elements on the list which is based on the principle grammatical terms of each word, this process retrieves elements. Some elements placed side by side provides a common concept. As a matter of fact, they complement one another. Hence, the *knowledge extraction* process distinguishes these elements and integrates them. The *knowledge extraction* process compares the parts of speech of elements. If both elements are subject, object, or action, then the process compares the major information of parts of speech. Some elements such as preposition may be deleted. Others like determiner may be used to distinguish that element is either singular or plural. If one element presents an auxiliary verb and the next one presents the main verb, the first element may be purged. Because this process uses natural language processing concepts to generate the final list and it is out of the scope of this research, we don't provide detail explanation of this process. Occasionally, subject or object carries one or more conditions. These conditions may be recognized by relative pronouns like who or whom in the original sentence. In this case the condition part will be added to the element defined as subject or object. Finally, the final list format looks like:

{(T,X), < w_1,(entity type, ∀ / ∃ / □/ - , entity, neg/pos meaning): condition>, <w_2, (entity type, - , entity, neg/pos meaning)>, ..., <w_n, (entity type, ∀ / ∃ / □/ -, entity, neg/pos meaning): condition> }

Conditions have the following format:

(EnvCon: (time: date&time, cond:value))

Algorithm *knowledge_extraction*

Input: *W_info*: list of words and grammatical terms attached to each word.

Output: *W_final*: list of critical words and related grammatical terms

1) Read elements from W_info
2) Purge some elements like prepositions
3) Group elements which have been correlated based on the principle grammatical terms and parts of speech, and select the element conveying the most valued information.
4) Collect metadata of correlated elements and choose the most appropriate metadata for the selected element
5) Replace subject with accessing user, or controlling user
6) If there are auxiliary verb and main verb, delete the auxiliary verb. Replace verb with action attached metadata to the selected element and store as one element.
7) Replace object with target user or target resource
8) Replace determiner with quantifier symbol like ∀ and ∃
9) Attach metadata information to the selected element and add to final list

Rule Maker

Rule maker process uses a predefined format to generate a rule. A rule may have the following format:

{< ∀ / ∃ / □/ - Ua [(att:value) … (att:value)] >, <action, [(att:valur), …… (att:value)] >, ….., < ∀ / ∃ / □/ - Tu/Tr [(att:value) … (att:value)] ∀ / ∃ / □/ - Uc [(att:value) … (att:value)] >, [<environment condition: (att:value) [(att:value), ….., (att:value)]] }

Algorithm *rule_maker*

Input: *W_final*: list of critical words and related grammatical terms

Output: *Rule*: list of subject, action and object

1) Read elements from *W_info*
2) Replace accessing user with Ua, and controlling user with Uc
3) Replace target user with Tu, and target resource with Tr
4) Rewrite elements with particular format
5) Return *Rule*

Request Maker

Request maker process uses a predefined format to generate a request.

Algorithm *request_maker*

Input: *W_final*: list of critical words and related grammatical terms

Output: *Rule*: list of subject, action and object

1) Read elements from *W_info*
2) Replace accessing user with Ua and controlling user with Uc
3) Replace target user with Tu and target resource with Tr
4) Rewrite elements with particular format
5) Return *Request*

Request engine

Receiving request, request engine calls several processes to collect all rules in the social network site, find the one, which fulfills the request perfectly, and ultimately either accept or deny the request.

Algorithm *request_engine*

Input: *Request:* a formatted request

Output: *answer*: a sequence of character that may be "*accepted*" or "*rejected*"

1) Call *selecting_rules*
2) Call *selecting_attributes*
3) Call *deriving_result*
4) Return *answer*

Selecting Rules

This process collects rules from several repositories. Some rules are extracted from the repository of the target user or target resource. Target user and target resource are included in the request. Some rules are extracted from the repository associated with the supervisor. These selected rules must be related to the action mentioned in the request. Moreover, if the request includes environmental conditions, these rules must meet that condition also.

Algorithm *selecting_rules*

Input: *Request:* a formatted request

Output: *rules*: set of rules, *UA*: accessing user

1) Read *Request*
2) Extract target user or target resource
3) Extract action
4) Extract accessing user
5) Read all rules stored in the repository associated to target user or target resource
6) Select rules which are related to the specific action and environmental condition meets the current condition of social network environment
7) Read all rules which are defined by supervisor and provided to protect the specific target resource or target user in terms of the specific action and particular environmental condition
8) Return *rules, UA*

Selecting Attributes

After passing rules collected by the *selecting rules* process into the selecting attributes process, selecting attributes searches attributes associated with the accessing user of rules and attributes associated with the accessing user of the request. Accessing users' properties mentioned in the rules may provide value for a name, relationship type, or label. In this case, selection attributes process determines that the accessing user in the request has a particular relationship with the target user or controlling user, or the accessing user in the request is a member of the label mentioned in the rules or not.

Algorithm *selecting_attributes*

Input: *rules:* set of rules, *UA*: user initiated an inquiry

Output: *yesrules*: set of rules supported the request, *norules*: set of rules rejected the request

1) Extract attributes of *UA*
2) Loop read item from *rules* {
   2.1) Extract accessing user from *rules*
   2.2) Extract properties of accessing user from *rules*
   2.3) Extract attributes of accessing user from *rules*

2.4) If UA.attributes = access user. Attributes under the defined environment condition

2.4.1) *yesrules*.add(*rules*)

2.5) Else

2.5.1) *norules*.add(*rules*) }

3) Return *yesrules*

4) Return *norules*

Deriving Results

The main responsibility of result inference is accomplishing the final answer to a request generated by an accessing user. This answer may accept or reject the inquiry. In order to achieve the result, value of the two received lists (e.g. *yesrules*, *norules*) must be checked. The social network site will not able to provide a subtle answer, if two lists are empty. If one list has value the other does not, the answer could be *accepted* or *rejected*. Otherwise, there are several rules which are not collaborated and are not able to derive a unique result. In this case, the algorithm must be able to define a solution in order to generate a unique result. The algorithm would sort the two received lists in terms of the time when the inquiry has been fed into social network site. The items of both lists are sorted in a descending way. The first item of the two lists will be compared. This means that recently added inquiry has the most priority for making a final result. Hence, the algorithm chooses the greater time. If the time is equal, the algorithm checks which rule dominates another rule. If the rule is chosen from *yesrules*, this means the rule from *yesrules* dominates the rule extracted from *norules*. So the accessing user has given permission to run the action against the resource which has been requested. Otherwise, the accessing user is not able to do the particular action.

Algorithm *deriving_result*

Input: *yesrules:* set of rules, *norules:* set of rules

Output: *Answer*: a sequence of character that may be "*accepted*" or "*rejected*"

1) If *yesrules* is empty and *norules* is empty {
   1.1) print "social network site is not able to make decision precisely, so the request will be rejected"
   1.2) Return *answer* = '*rejected'*}
2) If *yesrules* is not empty and *norules* is empty

   2.1) Return *answer* = '*accepted'*

3) Else if *yesrules* is empty and *norules* is not empty
   3.1) Return *answer* = '*accepted'*
4) Sort *yesrules* descending in terms of time when the rules have been inserted into social network site.
5) Select the rules from *yesrules* which were the most recently added to the social network site
6) Sort *norules* descending in terms of time when the rules have been inserted into social network site.
7) Select the rules from *norules* which were the most recently added to the social network site
8) Loop read (*yesrules)* is not End_of_list and read (*norules)* is not End_of_list {
   8.1) If *yesrules.Time* > *norules. Time*
        answer = '*accepted'*
   8.2) Else if *yesrules.Time* < *norules.Time*
        answer = '*rejected'*
   8.3) Else if *yesrules* dominates *norules*
        answer = '*accepted'*
   8.4) Else if *norules* dominates *yesrules*
        answer = '*rejected'* }
9) If *answer* != '*rejected' and answer* != *'accepted'*{
   9.1) print "social network site is not able to make decision precisely, so the request will be rejected"
   9.2) *answer* = '*rejected'*}

10)Return *answer*

7. Conclusions and Future Works

Conclusion

This work presents a new information security model to secure access to a resource or individual data in a social network, from unauthorized users. This model, which is the PBAAC, encompasses the ABAC model and the PBAC model. The PBAAC model enforces effective access control in a single and distributed environment.

As mentioned before, the PBAAC fundamental model includes either one social network site or one organization, and the PBAAC enterprise model (or distributed) includes either several diverse social network sites or organizations. Under both environments, users might have various resources and wish to share them with other legal users but not illegal users.

In the PBAAC method, every user can secure belongings by generating several policies in natural language. Furthermore, users who intend to access or do an operation on a resource are able to generate the inquiry in natural language. By defining policies or inquiries in natural language, users are able to determine guidelines for controlling belongings in an efficient way, and requestors are able to describe requirements with more clarification as well.

The PBAAC method offers a well-formed expression for storing policies and inquiries in the social network. After receiving policies or inquiries, the PBAAC method converts them into a new format. A policy is transformed to a rule and stored on the repository associated with a user who created the policy. An inquiry is transformed to a query, which will be evaluated by the proposed method.

Under this method, users are responsible to preserve their resources that could have one or several different parts. Because a simple resource has only one owner, determining access control to a simple resource is accomplished by a set of rules defined by only that owner. While

for a composed resource, enforcing access control is accomplished by several sets of rules; each set of rules is defined by the owner of a particular part of the composed resource, and each set of rules determines access to a part of the resource. This feature gives users the power to manage resources in a highly secure way.

Also, a requestor is able to feed an inquiry into a social network by using natural language. Using natural language provides users more flexibility and empowers users to protect assets in a remarkably secure way.

A social network site or organization is governed by a super user who defines general policies to enforce access control to all assets and resources existing in the environment. Hence, under the PBAAC method, there are two classes of rules, the first is defined by ordinary users and the second is defined by the super user, although it is assumed that a super user is able to determine who is able to dominate others, and which rule overrides other rules.

When a request is generated to access or do an operation on a particular object, the request will be evaluated, and then will be either accepted or rejected by using the access control decision engine. The access control decision engine needs to collect several rules, several attributes and environment condition. The several rules are defined to enforce access control of the resource. The several attributes are attached to entities participating in the rules and query. Afterwards, the access control decision engine evaluates them and derives a result. If the access control decision engine finds conflicts among rules, it requires choosing one, which fulfills the query perfectly.

The enterprise environment is designed for a homogeneous collection of social network sites. In addition to the collection of social network sites, this model includes a supervision section to make connections between other social network sites. The PBAAC provides an effective way for

sharing resources and enforcing access control, and it employs several independent graphs. The supervision graph collaborates with the other graphs, each of which is a social network.

The PBAAC model utilizes rules and attributes to accept or reject a request. This model is a compromise among having more basic rules including one attribute, fewer basic rules including several attributes, and nested rules with or without attributes. The basic rule with one attribute means that a rule addresses the entity directly by using the name of that entity. The basic rule with several attributes means that a rule addresses the entity by using several attributes of that entity.

The nested rule means that a basic rule may contain a subject which itself is a basic rule, or an object which itself is a basic rule, or both simultaneously. The nesting of the subjects and objects can continue indefinitely. The higher the number of basic rules, the better the performance of the proposed model.

This PBAAC model or its subtypes, the fundamental model and the enterprise model, make the following contributions:

- encompasses of the ABAC model and the PBAC model, enables users to define various policies to govern their resources and to feed inquiries to social network in order to access or perform an operation on a resource in the most effective way.
- enforces finer-grained access control in the basic and the enterprise social networks.
  - The basic or fundamental model, which is designed for one social network, shares resources and enforces access control. This model uses a specific data model, which captures data from a directed graph format. This format allows an efficient way to extract information through a graph search.

- The distributed or enterprise model, which is designed for a homogeneous collection of social network sites, includes a supervision to make a connection between other social network sites. Having an effective way for sharing resources and enforcing access control, this model employs several independent graphs in which one graph connects to the other graphs by the supervision graph.

- utilizes a language specification convertor to transform a policy or an inquiry to a well-formed expression. This expression follows the format (subject, action, object, condition) to present a rule or a request. By this definition, rules accept nested expressions, which means the subject and the object could be a well-formed expression, as well.
- offers a technique to enforce access control to an object when it is composed of other objects. Each composing object has its own rules to manage the access.
- determines a solution providing several rules that meet a received request, but all determined rules cannot exist together and not all are true together. The proposed model uses three criteria to choose a rule which fulfills the request perfectly.
- uses rules and attributes to accept or reject a request. Data analysis proves that higher numbers of basic rules improve or speed up the performance of the system.

Future works

As a future plan, the aforementioned model should be extended to cover several cases as listed:

- Define access user as a non-person, so the proposed model is able to utilize a social network and the non-person generates an inquiry on behalf of a person.
- The supervisor generates an inquiry to access an object. Occasionally, the supervisor should be able to generate various inquiries to ensure that the information security implemented for a social network blocks all unmanaged requests.

- Under the enterprise model, the supervisor should be able to define policies to manage all inquiries.
- Store all resources under one umbrella in the enterprise model. By implementing this feature, we hope the performance of social networks will be improved. Moreover, saving all users' rules and all supervisors' rules in one location decreases the time of graph traversal in order to reach the goal.
- Requesters should not be forced to be registered in the social network. Since the attributes belonging to the requester meet the criteria for achieving entry, access will be granted. This approach is definitely useful for social networks wherein resource owners wish unregistered users to be able to have access if they have attributes that meet certain criteria. This method empowers social networks to be free to have a predefined list of users who are approved for accessing resources; this is critical for enterprise social networks where people might join or leave the social network arbitrarily.

8. Bibliography

[1] Committee on National Security Systems: National Information Assurance (IA) Glossary, CNSS Instruction No. 4009, 26 April 2010.

[2] A. Hickey, "Wireless security's broken skeleton in the closet," http://searchnetworking.techtarget.com/news/1252992/WEP-Wireless-securitys-brokenskeleton-in-the-closet, April 26, 2007.

[3] Jason Andress, "The Basic of Information Security-Understanding the Fundamental of InfoSec in Theory and Practice," Amsterdam: Synger Press, 2011.

[4] http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA, November 2014.

[5] Ravi Sandhu, Pierangela Samarati, "Authentication, Access Control, and Audit,". ACM Computing Surveys, Vol. 28, No. 1, March 1996.

[6] Vincent C. Hu, D. Richard Kuhn, David F. Ferraiolo, "Attribute-Based Access Control," National Institute of Standard and Technology, Gaithersburg, MD, USA, February 2015.

[7] Ed Coyne, timothy R. Weil, "ABAC and RBAC: Scalable, Flexible, and Auditable Access Management,". IEEE computer society, IT Pro May/June 2013

[8] Wolfgang Emmerich, "Authentication, Access Control, Auditing, and Non-Repudiation," December 2015.

[9] Mike Chapple, Debra Littlejohn Shinder, Ed Tittel, "TICSA Certification: Information Security Basics," Pearson IT Certification, Nov 2002.

[10] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST Special Publication 800-162, National Institute of Standard Technology, January 2014. http://dx.doi.org/10.6028/NIST.SP.800-162

[11] L. Wang, D. Wijesekera, and S. Jajodia, "A Logic-based Framework for Attribute Based Access Control," in Proceedings of the 2004 ACM workshop on Formal Methods in Security Engineering, FMSE '04, ACM, New York (October 2004) 45-55. http://dx.doi.org/10.1145/1029133.1029140.

[12] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini, "A Constraint and Attribute Based Security Framework for Dynamic Role Assignment in Collaborative Environments," in Collaborative Computing: Networking, Applications and Worksharing, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 10, 322-339 (2009). http://dx.doi.org/10.1007/978-3-642-03354-4_24.

[13] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for Web Services," in Proceedings of the 2005 IEEE International Conference on Web Services, ICWS 2005,

IEEE Computer Society, Los Alamitos, California (2005) 561 - 569. http://dx.doi.org/10.1109/ICWS.2005.25.

[14] Yuan Cheng, Jaehong Park, and Ravi Sandhu, "A User-to-User Relationship-based Access Control Model for Online Social Networks," In the proceedings of the 26th IFIP, Conference on Data and Application Security and Privacy (DBSec '12), 2012.

[15] Yuan Cheng, Jaehong Park, and Ravi Sandhu, "Attribute-aware Relationship-based Access Control for Online Social Networks," Institute for Cyber Security, University of Texas at San Antonio. August 2014.

[16] Jennifer Golbeck, James Hendler, "Interfering Binary Trust Relationships in Web-Based Social Networks,", University of Meryland, College Park, 2006.

[17] Marsh, S, "Formalizing Trust as a Computational Concept,", PhD thesis. Department of Mathematics and Computer Science, University of Sterling, 1994.

[18] Castelfranchi, C, and Falcon, R, "Principle of trust for MAS: Cognitive anatomy, social importance, and quantification,". In Proceedings of the 3rd International Conference on Multi Agent System. 1998.

[19] Castelfranchi, C, and Falcon, R, "Social trust: A cognitive approach. In trust and Deception in Virtual Societies," C. Castelfranchi and Y. Tan, Eds. Kluwer Academic Publishers.2002.

[20] Eric Yuan, Jin Tong, "Attributed Based Access Control (ABAC) for Web Services," Booz Allen Hamilton, Inc., McLean, Virginia, 2005.

[21] Yanjun Zuo, Brajendra Panda, "A trust-based Model for Information Integrity in Open Systems," Department of Information Systems and Business Education, University of North Dakota, Grand Forks, ND,USA; Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR,USA, 2006.

[22] Shen Hai-bo, Hong Fan,"An Attribute-Based Access Control Model for Web Services," School of Computer, Huazhong University of Science and Technology, Wuhan 430074, China, 2006.

[23] Ting Yu, "Negotiating trust in the Web," IEEE Internet Computing· December 2002.

[24] Vladimir Kolovski, "Logic-based access control policy specification and management," Department of Computer Science, University of Maryland, College Park, USA, 2007.

[25] Thomas Y. C. Woo and Simon S. Lam, "Authorization in distributed systems: a new approach," Journal of Computer Security, 1993.

[26] David E. Bell, Leonard J. Lapadula, "secure computer system: unified exposition and MULTICS interpretation," Technical report MTR-2997, the MITRE Corporation, 1976.

[27] Nighui Li, Benjamin N. Grosof, and Joam Feigenbaum, “Delegation logic: a logic-based approach to distributed authorization,” ACM Trans. Inf. Syst. Secure., 6(1):128-171, 2003.

[28] Marianne Winslett, Ckarles C. Zhang, and Piero A. Bonatti, “Peeraccess: a logic for distribution authorization,” Proceeding of the 12th ACM conference on computer and communications security, New York, NY, USA, 2005.

[29] Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmendilla, Kent Seamons, and Marianne Winslett, “No registration needed: how to use declarative policies and negotiation to access sensitive resources on the semantic web,” In European Semantic Web Symposium, May 2004.

[30] Joseph Y. Halpern and Vicky Weissman. Using _rst-order logic to reason about policies. In In Proceedings of the Computer Security Foundations Workshop (CSFW'03), Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[31] Joseph Y. Halpern and Vicky Weissman. A formal foundation for xrml. In CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations, page 251, Washington, DC, USA, 2004. IEEE Computer Society.

[32] Mart__n Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. ACM Transactions on Programming Languages and Systems, 15(4):706{734, September 1993.

[33] C. A. R. Hoare. "Communicating sequential pr.ocesses," Commun. ACM, 21(8):666{677}, 1978.

[34] Lujo Bauer, Scott Garriss, Michael K. Reiter “Distributed proving in access control systems,” In SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy, pages 81{95, Washington, DC, USA, 2005. IEEE Computer Society.

[35] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. "Flexible support for multiple access control policies,". Database Systems, 26(2):214{260}, 2001.

[36] Michael A. Harrison, Walter L. Ruzzo, and Je_rey D. Ullman. "Protection in operating systems,". Commun. ACM, 19(8):461{471} 1976.

[37] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. Trbac: A temporal role-based access control model. ACM Trans. Inf. Syst. Secur., 4(3):191{233}, 2001.

[38] Carrie E. Gates. "Access control requirements for Web 2.0 security and privacy," In IEEE Web 2.0 privacy and security work ship *(*W2SP’07*)*, Oakland, California, USA, May 2007.

[39] Philip W. L. Fong, “relationship-based access control: protection model and policy language,” department of computer science, university of Calgary, Calgary, Alberta, Canada, 2011.

[40] Philip W. L. Fong, Ida Siahaan "Relationship-Based Access Control Policies and Their Policy Languages," In the proceedings of the first ACM conference on Data and application security and privacy, New York, USA, 2011.

[41] Fahad T. Alotaiby, J. X. Chen, "A Model for Team-based Access Control (TMAC 2004)," George Mason University, Fairfox, Virginia, USA, 2004.

[42] Roshan K. Thomas, "Team-based Access Control (TMAC): a primitive for applying role-based access controls in collaborative environments," Odyssey Research and Technology Park 33 Thornwood Drive, Ithaca, NY 14850-1350, 1997.

[43] Isabel F. Cruz, Rigel Gjomemo, Benjamin Lin, Mirko Orsini, "A Constraint and Attribute Based Security Framework for Dynamic Role Assignment in Collaborative Environments," ADVIS Lab – Department of Computer Science – University of Illinois at Chicago, 2009.

[44] Mohsen Saffarian, Qiang Tang, Willem Jonker, Pieter Hartel, "Dynamic User-Role Assignment in Remote Access Control, " Faculty of EWI, University of Twente, the Netherlands, Philips Research, the Netherlands, 2009.

[45] Ninghui Li John C. Mitchell, William H. Winsborough, "Design of a Role-based Trust-management Framework," Department of Computer Science, Stanford University, Gates 4B, Stanford, CA 94305-9045, and NAI Labs, Network Associates, Inc. 3060 Washington Road Glenwood, MD 21738, 2002.

[46] Ninghui Li and John C. Mitchell, "Datalog with Constraints: A Foundation for Trust Management Languages," Department of Computer Science, Stanford University Gates 4B, Stanford, CA 94305-9045, December 2002.

[47] Ravi Sandhu, Pierangela Samarati, "Access Control: Principles and Practices," IEEE Communication Magazine 1994.

[48] Jaehong Park, Ravi Sandhu, Yuan Cheng, "A User-Activity-Centric Framework for Access Control in Online Social Networks," University of Texas at San Antonio, September 2011.

[49] Moritz Becker, Cedric Fournet, and Andrew Gordon. Design and semantics of a decentralized authorization language. Computer Security Foundations Symposium, 2007. CSF '07. 20th IEEE, pages 3{15, 6-8 July 2007.

[50] Amirreza Masoumzadeh, James Joshi, "OSNAC: An Ontology-Based Access Control Model for Social Networking System," School of Information Science, University of Pittsburgh, 2010.

[51] Vincent C. Hu, Karen Scarfone, "Guidelines for Access Control System Evaluation Metrics," http://dx.doi.org/10.6028/NIST.IR.7874, September 2012.

[52] Vincent C. Hu, David F. Ferraiolo, D. Rick Kuhn, "Assessment of Access Control Systems," Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, Interagency Report 7316 September 2006.

[53] Summers R. C., "Secure Computing Threats and Safeguard," McGraw-Hill, 1997.

[54] Badger, L., Sterne, D. F., Sherman, D. L., Walker, K. M., Haghighat, S. A., "Practical Domain and Type Enforcement for UNIX," IEEE Symposium on Security and Privacy, 1995.

[55] Atluri V., Huang W., "An Authorization Model for Workflows," Proceedings of the Fifth European Symposium on Research in Computer Security in Lecture Notes in Computer Science, No 1146, 1996.

[56] National Computer Security Center (NCSC), "A Guide to Understanding Discretionary Access Control in Trusted System," Report NSCD-TG-003 Version1, 30 September 1987.

[57] Workflow Management Coalition, "Workflow Management Coalition Terminology & Glossary", http://www.wfmc.org/ Documentation number WFMC-TC-1011, February 1999.

[58] Pfleeger C. P., "Security In Computing," Second Edition, Prentice-Hall PTR, 1997.

[59] Mohammad Al-Kahtani, Ravi Sandhu, "Induced Role Hierarchies with Attributes-Based RBAC," Goerge Mason University, USA, 2003.

[60] Zhong, Y., Bhargava, B., and Mahoui, M., "Trustworthiness based authorization on WWW," In IEEE workshop on Security in Distributed Data Warehousing, New Orleans, October 2001.

[61] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. Rolebased, "access control model," IEEE Computer, 29(2), February 1996.

[62] Park, J., Sandhu, R., and Ahn, G. "Role-based access control on the web,". ACM Transactions on Information and System Security, Vol. 4, No 1, 2001.

[63] Herzberg, A., Mass, Y., and Mihaeli, J. "Access control meets public key infrastructure, or: assigning roles to strangers," Proceedings of the IEEE Symposium on Security and Privacy, 2000.

[64] Yao, W., Moody, K., Bacon, J., "A model of OASIS role-based access control and its support for active security," SACMAT'01, Chantilly VA, May 2001.

[65] Lightweight Directory Access Protocol (v3), RFC2251, December 1997.

[66] Dynamic Groups for LDAPV3 draft-haripriya-dynamicgroup-00.txt, October 2001.

[67] Al-Kahtani, M., and Sandhu, R., "A model for attribute-based user-role assignment," Proceedings of the18th Annual Computer Security Applications Conference, Las Vegas NV, December 2002.

[68] Miao Liu, He-Qing Guo, Jin-Dian Su. "An Attribute and Role Based Access Control Model for Web Services," College of Computer Science and Engineering, South China University of Technology, Guangzhou, China, August 2005.

[69] G. Saunders, M. Hitchens, V. Varadharajan. "Role-Based Access Control and the Access Control Matrix," Basser Department of Computer Science, University of Sydney, Australia, and School of Computing & Information Technology, University of Western Sydney (Nepan), Australia, July 2003.

[70] Xin Jin, Ravi Sandhu, Ram Krishnan, "RABAC: Role-Centric Attribute-Based Access Control," Institute for Cyber Security & Department of Computer Science, Institute for Cyber Security & Dept. of Elect. And Computer Eng. University of Texas at San Antonio. December 2012.

[71] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, "J. Karat, and A. Trombeta. Privacy-aware role-based access control,". *ACM Trans. Inf. Syst. Secur.*, 13(3):24:1–24:31, July 2010.

[72] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo. "Privacy-aware role based access control,". In Proceedings of the 12th ACM symposium on Access control models and technologies, SACMAT '07, pages 41–50, New York, NY, USA, 2007. ACM.

[73] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli. "Proposed NIST standard for role-based access control,". *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, August 2001.

[74] Sanjeev Arora, Eunjee Song, Yoonjeong Kim, "Modified Hierarchical Privacy-aware Role Based Access Control Model,". Dept. of Computer Science, Baylor University, Waco, Texas, USA, Dept. of Information Security Seoul Women's University Seoul, Rep. of Korea. October 2012.

[75] Barbara Carminati, Elena Ferrari, and Andrea Perego, "Rule-Based Access Control for Social Networks," DICOM, Universit`a degli Studi dell'Insubria, Varese, Italy, 2006.

[76] Barbara Carminati, Elena Ferrari, and Andrea Perego, "Enforcing Access Control in Web-Based Social Networks," DICOM, Universit 'a degli Studi dell.Insubria, October 2009.

[77] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao, "A Privacy Preservation Model for Facebook-Style Social Network Systems," Department of Computer Science, University of Calgary, Alberta, Canada, Department of Computer Science, University of Regina, Saskatchewan, Canada, 2009.

[78] Philip W. L. Fong, "Relationship-Based Access Control: Protection Model and Policy Language," Department of Computer Science, University of Calgary, Alberta, Canada, 2011.

[79] Philip W. L. Fong, Ida Siahaan, "Relationship-Based Access Control Policies and Their Policy Languages," Department of Computer Science, University of Calgary, Alberta, Canada, 2011.

[80] Hong Fan, Shen Hai-bo, "An Attribute-Based Access Control Model for Web Services," School of Computer, Huazhong University of Science and Technology, Wuhan 430074, China, 2006.

[81] Mike Chapple, Debra Littlejohn Shinder, Ed Tittel, "TICSA Certification: Information Security Basics," November 2002.

[82] Jonathon Tidswell, John Potter "An Approach to Dynamic Domain and Type Enforcement," Microsoft Research Institute, Department of Computing, Macquarie University, NSW 2109, June 2005.

[83] Yuan Cheng, Jaehong Park, Ravi Sandhu, "Attribute-aware relationship-based access control for online social networks," Institute for Cyber Security, University of Texas at San Antonio, USA, 2014.

[84] Moritz Becker and Peter Sewell. "Cassandra: Distributed access control policies with tunable expressiveness,". In POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04), page 159, Washington, DC, USA, 2004. IEEE Computer Society.

[85] https://en.wikipedia.org/wiki/Forward_chaining

[86] https://en.wikipedia.org/wiki/Backward_chaining

[87] https://en.wikipedia.org/wiki/Quantifier_linguistics

[88] https://en.wikipedia.org/wiki/Class_diagram

[89] https://en.wikipedia.org/wiki/Well-formed_element

[90] https://en.wikipedia.org/wiki/Well-formed_document

[91] Vincent C.Hu, Karen Scarfone, "Guidelines for Access Control System Evaluation Metrics," National Institute of Standards and Technology. Interagency Report 7874.

[92] Vincent C.Hu, David F. Ferraiolo, D. Rick Kuhn, “Assessment of Access Control Systems,” National Institute of Standards and Technology. Interagency Report 7316.

[93] John DeTreville. “Binder, a logic-based security language,”. In SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, page 105, Washington, DC, USA, 2002. IEEE Computer Society.

[94] Stuart J Russel, Peter Norvig, Ernest Davis “Artificial intelligence: a modern approach,” Prentice Hall 2010.

[95] Namarta Kapoor, Nischay Bahl, “Comparative Study of Forward and Backward Chaining in Artificial Intelligence,” Post Graduate Department of Computer Science, D.A.V. College, Jalandhar. International Journal of Engineering and Computer Science, April 2016.